Grant Agreement N° 215483

Title:	*Analysis on how to exploit codified HCI and codified context knowledge for SBA engineering – Confidential Annex*
Authors	*CITY, LERO, POLIMI, VUA*
Editor:	*Andreas Gehlert (UniDue)*
Reviewers:	*Branimir Wetzstein (UStutt)*
	Mohand-Said Hacid (UCBL)
Identifier:	*Deliverable CD JRA 1.1.5*
Type:	*Deliverable*
Version:	*1.0*
Date:	*15/03/2010*
Status:	*Final*
Class:	*Internal*

Management Summary

Deliverable CD-JRA-1.1.5 is a paper-based deliverable comprised of seven research papers. All papers deal with the exploitation of codified context knowledge. It can clearly be seen that on the one hand the different research communities interpret “context” differently ranging from HCI aspects to engineering aspects relevant for SBAs. On the other hand our analysis of the relation of the approaches to the life cycle model shows that the approaches try to integrate different phases of the life-cycle and, thus, S-Cube is now focussing more on integrated research, which covers more than one phase of the life-cycle of SBAs. This part of the deliverable contains the ten papers contained in CD-JRA-1.1.5

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 215483 (S-Cube).

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator)	Germany
Tilburg University	Netherlands
City University London	U.K.
Consiglio Nazionale delle Ricerche	Italy
Center for Scientific and Technological Research	Italy
The French National Institute for Research in Computer Science and Control	France
Lero - The Irish Software Engineering Research Centre	Ireland
Politecnico di Milano	Italy
MTA SZTAKI – Computer and Automation Research Institute	Hungary
Vienna University of Technology	Austria
Université Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politécnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
VU Amsterdam	Netherlands

Published S-Cube documents

All public S-Cube deliverables are available from the S-Cube Web Portal at the following URL:

http://www.s-cube-network.eu/

1 Kounkou, A.; Zachos, K.; Maiden, N.: Exploiting user error knowledge for service discovery. – not yet submitted.

Exploiting user error knowledge for service discovery

Angela Kounkou, Konstantinos Zachos, Neil Maiden
Centre for HCI Design, City University, UK
{sbbc775, kzachos, N.A.M.Maiden}@soi.city.ac.uk

Abstract

Although the interoperability and reuse capabilities associated with services are distinct advantages for dynamically building composed applications, these qualities can also introduce robustness issues in Service-Based Applications (SBA) due to the challenges of integrating and appropriately executing disparate error-handling methods. It is a well-documented fact in the field of Human Computer Interaction (HCI) that most system faults find their roots in errors committed by the human stakeholders at the various stages of their involvement with the system; little research however seems to consider the role of these stakeholders in SBAs' fault-tolerance. In this paper, we investigate whether HCI knowledge about user error can inform the engineering of SBAs - more specifically here, their discovery - and contribute to improving their robustness.

1 Introduction

Software services are increasingly being used in enterprise settings as well as in mainstream, day-to-day computing; visions of an upcoming Internet of services (e.g. [1]) and the predictions of IT analyst firms [2] do not show signs of this trend faltering. We believe that as services' application ranges and end user base augment, so will the need for robustness in Service Based Applications (SBAs) so that they continue to provide good service in anomalous situations, hence preventing possible productivity loss, reduced user satisfaction, and even accidents in safety-critical systems [3]. Consider for instance a journeyPlanner SBA mapping and delivering possible routes to a user-specified destination address: the disrupted provision of such services to a delivery company could amount to a backlog of scheduled deliveries as well as potential losses of dissatisfied clients.

The interoperability and reuse capabilities associated with services are distinct advantages for dynamically building heterogeneous, loosely-coupled SBAs [4]; they do however introduce robustness issues [5]. SBAs integrate services that are often developed using differing technologies that may implement different approaches for error-handling. For instance, some may implement a "throw and catch" approach where an error is internally handled while others might stop at outputting runtime errors for handling by an external human or automated component. Further, the granularity, format and content of these errors might differ from service to service, with some error codes being specific to a particular service and others being natural language, human-readable messages [5]. Consider for instance that the journeyPlanner SBA includes an *addressLookup* service that geographically locates an address provided by the user. Assume that an erroneous address input prevents the address location from being performed. Depending on the error handling strategy implemented in the *addressLookup* service and in the SBA, the application might either display an error code to the user, display a plain language error message, or "do nothing". We can imagine that for a wider audience, any other output than an understandable and actionable error message would negatively impact the fulfillment of the route-finding task and the end user's experience.

To the best of our knowledge, current research on fault-tolerance focuses on these interoperability issues and on mechanisms addressing fault detection, handling and recovery at various levels of the SBAs' technical infrastructure (e.g [3, 6-8]). We are not however aware of research focusing more closely on the human

sources of faults occurring in SBAs. It is a well-documented fact in the field of Human Computer Interaction (HCI) that most system faults find their roots in errors committed by the human stakeholders at the various stages of their involvement with the system: as end users, but also as designers and programmers introducing architectural errors and bugs in the software (e.g. [9]). As reported in [10], a study carried on large-scale Internet services' failure reports confirmed these findings by identifying operator errors as one of the main causes of user-visible service failures. In this paper, we investigate whether HCI knowledge about human error can inform the engineering of SBAs and contribute to improving their robustness. Drawing upon existing taxonomies of user error and on models of human interaction with interactive systems, we derived heuristics applicable to the engineering of SBAs preventing or handling common user errors, and proposed approaches for the codification of such heuristics into tools for design time and runtime service discovery.

2 User Error for SBA engineering

Errors have been defined as terms encompassing *"…all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and when these failures cannot be attributed to the intervention of some chance agency."* [11]. User errors are common feature of interactions between users and computer-based systems and have been extensively studied in the field of HCI; a number of taxonomies (e.g. [11-13]) exist to identify and classify user errors. Although they were initially considered as originating from people's erroneous assessments and decisions, user errors are now considered as the symptoms of possible trouble inside a system indicating a re-design need; specific guidelines and recommendations for their consideration and handling were developed for the design of interactive systems (e.g. [14, 15]). Such recommendations however, which often come in the form of design principles, are challenging to apply to SBA engineering. Consider for instance one such guideline aimed at improving a user's understanding of a system's operation to prevent the occurrence of errors: *"Make the action more perceptible – improve the match between actions and their outcomes"* (from [15]). Considering that SBAs are composed of existing, reusable software components using approaches that are often business-process-centric (e.g. [16, 17], it is challenging to think of ways such a generic principle could be transformed into concrete requirements or operations usable for services discovery and assembly. However, in view of the evidence of the utility of user error knowledge for software engineering, we explored possible novel ways to apply this knowledge to the specific case of service-oriented software engineering.

In order to frame our consideration of user error during user interactions with SBAs, two areas of HCI study were drawn upon: user error taxonomies and user interaction models. First, taxonomies of human error provided a reference to the different, broad categories of errors users make. We primarily drew on Rasmussen's and Norman's error types classifications. Rasmussen described three types of user errors: *skill-based performance errors* which are largely errors of execution; *rule-based performance errors* resulting from a human inability to correctly assess a situation; and *knowledge-based performance errors*, which he identified as the product of shortcomings in the user knowledge or in his abilities to apply existing knowledge to new situations [18]. Norman categorised user errors into *slips*, errors entailing the incorrect execution of a correct action sequence, and *mistakes*, correct executions of incorrect action sequences [14]. Interaction models describing users' interaction with a system were then examined for possible mappings between the user error types described in the taxonomies and specific stages of a user's interaction with an SBA. Norman's model in particular (Figure 1 below) broadly describes user's interaction with systems. It defines 7 stages of interaction starting with the user's goal or desired state of the world to be achieved. The following 3 stages (intention to act, formation of an action sequence, and execution of the action sequence) describe the user's perception and actions on the system in order to bring about the desired change in the state of the world. The last 3 interaction stages (world state perception and its subsequent interpretation and evaluation) describe the processes the user goes through when assessing the results of his actions on the system and the changes they brought on the world, including whether they brought him closer to his goal – the desired world state.

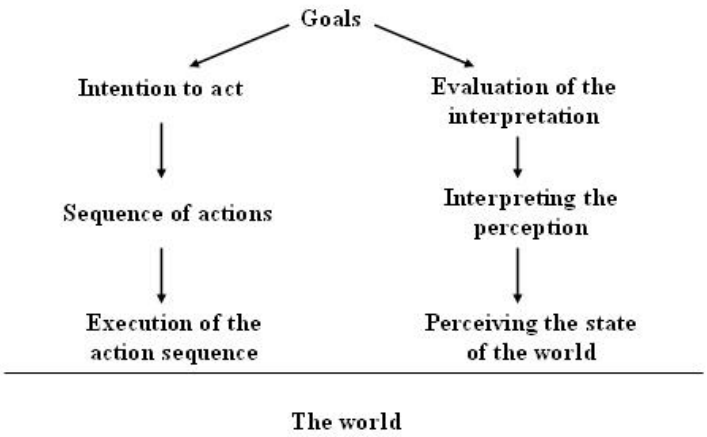


Figure 1: Norman's model of interaction

Considering user's interactions with SBAs, errors could solely be apprehended as the erroneous actions users perform on the system using a provided interface. The interaction model however helped to highlight the stages anterior to physical action, where errors in how the user comprehends the system and forms an intention to use it are formed. It also provided insight into the stages posterior to the user's actions on the interface, where the user's assessment of his actions' effects reinforce his understanding of the SBA's "workings", which influences any further actions to be made.

Following assessment and as illustrated in Figure 2 below, common errors drawn from established taxonomies were mapped to likely interaction stages for their occurrence. Possible causes for error were derived, and coarse-grained recommendations for the prevention, mitigation or handling of these errors were proposed for application at the service discovery phase of SBA engineering. Although external factors such as the context of use and the user's own characteristics can impact on user errors, they were not included in this exercise.

Interaction stage	Error	Possible causes	Selection recommendations
Goal formation	Mismatch between the goal and the SBA functionality	Incorrect service discovery query Incorrectly specified service discovery query No service description Incomplete service description Unclear service description Unclear user interface	Select fully specified services Select services with plain language description Select services with standards compliant user interfaces
Intention formation	Incorrect intention formed	Missing information Unclear information Incorrect heuristics/rules applied Incorrectly applied heuristics/rules Information overload	Select fully specified services Select services with a help function
Action sequence formation	Incorrect action sequence formed	Incorrect mental model of the service Misunderstanding of controls Bad visibility of controls	Select services with a help function Select services with standards compliant user interfaces
Action sequence execution	Incorrect action sequence execution	Bad visibility of controls Poor accessibility of controls Incorrect data input	Select services accessible to the user Select services with standards compliant user interfaces Select services validating data input
Perception of the SBA state	No perception of SBA state Incorrect perception of SBA state	No system feedback Bad visibility of feedback Poor accessibility of feedback Information overload	Select services providing feedback on action Select services providing multi-sensory feedback Select services providing feedback accessible to the user Select services compliant with usability standards
Interpretation of perceived SBA state	Incorrect interpretation of SBA state	Incorrect feedback Ambiguous feedback Incorrect mental model	Select services compliant with usability standards
Evaluation of SBA state	Incorrect evaluation of SBA state interpretation	Incorrect mental model	Select services compliant with usability standards

Figure 2: Possible user error types at various stages of interaction with SBA, inferred causes, and remedial recommendations

While some possible causes for user error could not be expected to be fully addressed during the engineering of the SBA (for instance, the user applying inappropriate heuristics when forming an intention to use the system, which could likely stem from internal causes e.g. inaccurate knowledge or poor IT skills), Figure 2 revealed potential areas for the application of user error knowledge during the SBA lifecycle.

At the design stage for instance, a mismatch between the user's goal and the functionalities offered by the SBA could be at the root of user errors (see also Figure 3 below). This mismatch may occur for various reasons (e.g. issuing an incorrect query discovering services that do not match the user or the SBA assembler's requirements), some of which could be handled at the discovery stage of the SBA lifecycle. For instance, a cause such as an ambiguous user interface (i.e. which does not permit the user to form a correct opinion of the SBA's overall function) may be avoided by selecting only services implementing user interfaces that are compliant with relevant HCI design standards.

Interaction stage	Possible Error	Possible causes	Selection recommendations
Goal formation	Mismatch between the goal and the SBA functionality	Incorrect service discovery query	Select fully specified services
		Incorrectly specified service discovery query	Select services with plain language description
		No service description	Select services with standards compliant user interfaces
		Incomplete service description	
		Unclear service description	
		Unclear user interface	

Figure 3: user error at the Goal formation stage

During physical, actionable interaction with the service at run-time (Figure 4), the end-user may commit mistakes triggered by a poor accessibility of the SBA controls (which may be solved by selecting services compliant with relevant accessibility standards) or by incorrect data entry (in which case SBAs requiring data input could also be required to perform data validation checks). Consider for instance that the journeyPlanner application is part of an eGovernment suite of software made available by a country's official authorities. The SBA is expected to be used by a wide array of users encompassing various ages, nationalities, IT skills, and abilities; it further has to be accessible from a host of computing platforms including mobile platforms. User input errors can reasonably be foreseen to be a likely occurrence under these conditions. Generally speaking, the improved robustness for such an SBA would among other require the use of services whose operations are accessible to the wider audience, whose implemented interfaces are compliant with the relevant guidelines and standards in effect, and which provide data input validation to address possible errors committed by the user when entering start or destination addresses.

Interaction stage	Error	Possible causes	Selection recommendations
Action sequence execution	Incorrect action sequence execution	Bad visibility of controls Poor accessibility of controls Incorrect data input	Select services accessible to the user Select services with standards compliant user interfaces Select services validating data input

Figure 4: user error at the Action execution stage

As a last example, at runtime a user's incorrect perception of an SBA's state (Figure 5 below), which would possibly cause further user errors to occur when interacting with the SBA, may result from an inappropriate system feedback – in which case it may be recommended that the relevant services composing the SBA provide human-readable feedback for instance.

Interaction stage	Error	Possible causes	Selection recommendations
Perception of SBA state	No perception of SBA state Incorrect perception of SBA state	No system feedback Bad visibility of feedback Poor accessibility of feedback Information overload	Select services providing feedback on action Select services providing multi-sensory feedback Select services providing feedback accessible to the user Select services compliant with usability standards

Figure 5: user error during perception of the SBA state

Overall, the insights gained at this stage indicated that user error knowledge may inform derived, simple heuristics (e.g. Figure 2 – discovery heuristics) for the engineering of SBAs that avoid or mitigate common user errors. These heuristics can be envisaged to span several stages of the SBA lifecycle, for instance service discovery for the specification of selection heuristics and rules for services implementing specific error-handling functions (e.g. help function, input validation); or service specification for richer descriptions of services' error-handling capabilities. The following section reports on a proposed codification of user error knowledge using these heuristics for design time and runtime service discovery. It refers to the service discovery approach outlined in the SeCSE[1] project, which made available development platform and tools permitting an implementation and future empirical validations of our proposed solutions.

2.1 User error knowledge for service discovery

Design time service discovery entails finding services for the composition of an SBA matching specific requirements. Service discovery in SeCSE calls upon a service discovery algorithm (EDDiE) which formulates service requests from use case and requirements specifications expressed in structured natural language. Overall the service query is divided into sentences, tokenised, tagged, and expanded with each token's morphological root (e.g. *drive* for *driving*). The algorithm then applies procedures to disambiguate each term by defining its correct sense and tagging it with that sense (e.g. defining a *driver* to be a *vehicle* rather than a *type of golf club*). The algorithm then expands each term with other terms that have similar meaning according to the tagged sense to increase the likelihood of a match with a service description (e.g. since *driver* is

[1] http://www.secse-project.eu/

synonymous with *motorist* it is then included in the query). Finally the algorithm matches all expanded and sense-tagged query terms to terms found in the service description *facet*s in the service registry. *Facets* are projections over service properties that describe these properties in *Facet Specifications* using *Languages* that may be natural (e.g. English) or XML based (hence permitting both human and computerized interpretation of the specification) as depicted in the conceptual model in Figure 6 below

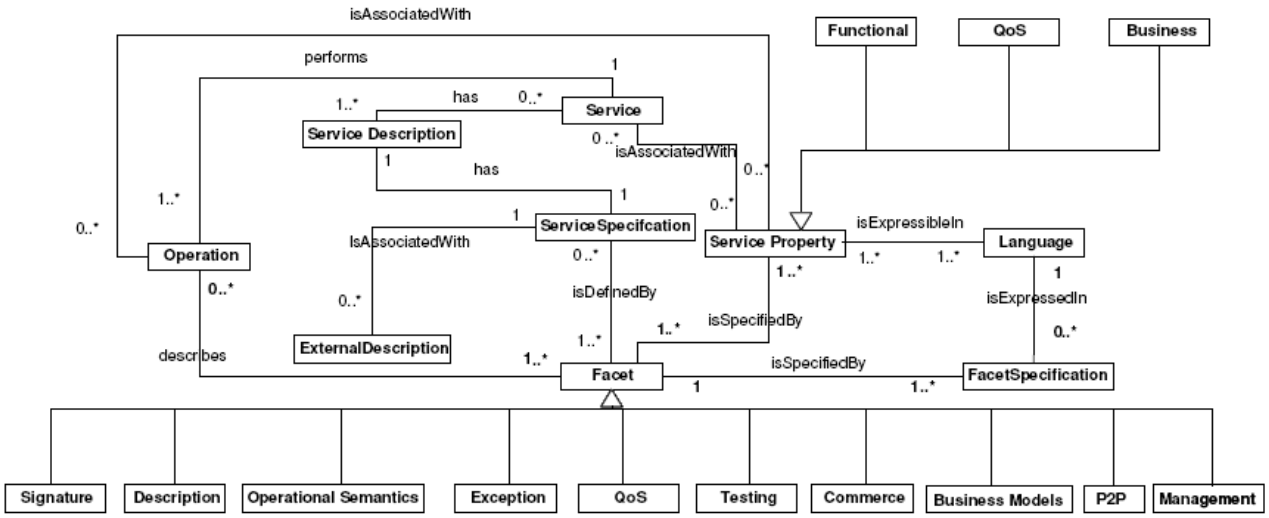


Figure 6: Conceptual model of faceted service description

Using the initial coarse-grained selection recommendations outlined in Figure 2, we derived an initial set of service selection heuristics (below) linking a possible error handling or mitigation action to the service description facet encapsulating the relevant information, this for matching during discovery. For each recommendation, suggested selection heuristics were proposed - for instance, as per "Recommendation 1" below, an SBA such as the journeyPlanner might mitigate the impact of user input errors by: providing users with feedback understandable to non-IT literate users; providing a spell-checker facility; providing predictive data input; and/or suggesting similar correct addresses.

Recommendation 1: Select services validating data input

Heuristic name: Availability of spelling validation
IF End user data input is performed
THEN select all services where DescriptionFacet/DescriptionScheme/ServiceOperations/Operation = "spell checker"

Heuristic name: Availability of predictive data entry
IF End user data input is performed
THEN select all services where DescriptionFacet/DescriptionScheme/ServiceOperations/Operation = "predictive text"

Heuristic name: Recovery from invalid input
IF End user data input is performed
THEN select all services where TestingFacet/TestSuite/TestCase/TestName= "GUI data input test"

Heuristic name: Recovery from erroneous data entry order
IF End user data input is performed
THEN select all services where TestingFacet/TestSuite/TestCase/TestName= "informal entry order test"

Recommendation 2: Select services with a help function

Heuristic name: Availability of help function

IF the SBA is intended for end users' use
THEN select all services where DescriptionFacet/DescriptionScheme/ServiceOperations/Operation = "Help"

Heuristic name: Availability of undo function
IF the SBA is intended for end users' use
THEN select all services where DescriptionFacet/DescriptionScheme/ServiceOperations/Operation = "Undo"

Heuristic name: Availability of helpline
IF the SBA is intended for end users' use
THEN select all services where CommerceFacet/BusinessEntity/Contacts = true

Recommendation 3: Select services providing feedback on action

Heuristic name: Availability of end user error recovery support
IF the SBA is intended for end users' use
THEN select all services where OperationalSemanticsFacet/OCLOperationalSemanticsScheme/OperationalSemantics/PostCondition = "plain language action feedback"

Heuristic name: Adherence to HCI design standards
IF the SBA is intended for end users' use
THEN select all services where DescriptionFacet/DescriptionScheme/Miscellaneous = "ISO 9241 compliant"

Heuristic name: Availability of user-accessible feedback
IF the SBA is intended for end users' use
THEN select all services where OperationalSemanticsFacet/OCLOperationalSemanticsScheme/OperationalSemantics/PostCondition = "multi-sensory feedback"

Recommendation 4: Select services accessible to the user

Heuristic name: Accessibility of service
IF the service is intended for end users' use
THEN select all services where DescriptionFacet/DescriptionScheme/Miscellaneous = "WCAG compliant"

Recommendation 5: Select fully specified services

Heuristic name: Availability of full service description
IF SBA assembler requires manual check of service's match to requirements
THEN select all services where
DescriptionFacet/DescriptionScheme/ServiceOperations/Operation = true **And**
SignatureFacet/OperationalFootprint = true **And**
SignatureFacet/BindingInformation = true

Recommendation 6: Select services compliant with usability standards

Heuristic name: Usability standards compliance
IF the SBA is intended for use by end users
THEN select all services where DescriptionFacet/DescriptionScheme/Miscellaneous = "ISO 9241 compliant"

The sets of initial heuristics presented above are currently being implemented as discovery rules in the SeCSE service discovery platform in order to empirically evaluate their impact on service discovery using the EDDiE algorithm; it is expected that they will be refined as an outcome of the validation activities performed.

2.2 User error knowledge for SBA adaptation

As defined in the S-Cube knowledge model[2], adaptation is a process of modifying SBAs in order to satisfy new requirements and demands dictated by the environment. Adaptation can be necessary to optimise SBAs, to react to changes in context, or as part of a recovery strategy to effect fault-recovery. We suggest that user error knowledge can be used for the latter case, i.e. to specify rules for the handling of user error occurring at run-time where appropriate. Adaptation can among others be realised through Run-time Service Discovery (RSD). RSD is concerned with the discovery of alternative services to replace services already integrated in an SBA that have become unavailable or unsuitable (e.g. they do not meet functional requirements or required QoS anymore). As demonstrated in section 2.1., user error knowledge can be exploited for the development of discovery rules, a principle that may be used for RSD.

An open source tool performing RSD was described in detail in [19]. As an overview, the tool's graphical user interface allows a human assembler to request the automatic construction of a query for discovering a service matching the signature and behaviour of a specified service to be replaced. The assembler can then choose to select and load pre-supplied *constraint queries*. Constraint queries are logical expressions used in RSD to refine the sets of discovered services; they specify additional constraints that a candidate service's specification must match on top of matching the structural and behavioural behaviour of the service to be replaced. These constraint queries are implemented in a Constraint Service Query Language (Constraint-SQL) and have the following four attributes:

- *Weight*: a number between 0 and 1 for the prioritisation of the parameters in a service discovery query.
- *Type*: its value is either "hard" or "soft", indicating respectively that the constraint has to be satisfied by all the services discovered, or would preferably but not necessarily satisfied.
- *Name*: a description of the constraint
- *Optional*: either "true" or "false" to indicate respectively whether the constraint query is optional or compulsory

As a solution idea for the application of user error knowledge to RSD, we developed a set of generic constraint queries informed by user error knowledge to extend the RSD tool by adding this set to its repository. More specifically, the heuristics presented in section 2.1. were operationalised and organised in sets (e.g. accessibility, data input validation, feedback on action), with the aim to make a coherent collection of rules available to the SBA assembler for loading pre-defined, common user error constraint queries when required (Figure 7).

[2] http://www.s-cube-network.eu/km/terms/a/adaptation/?searchterm=adaptation

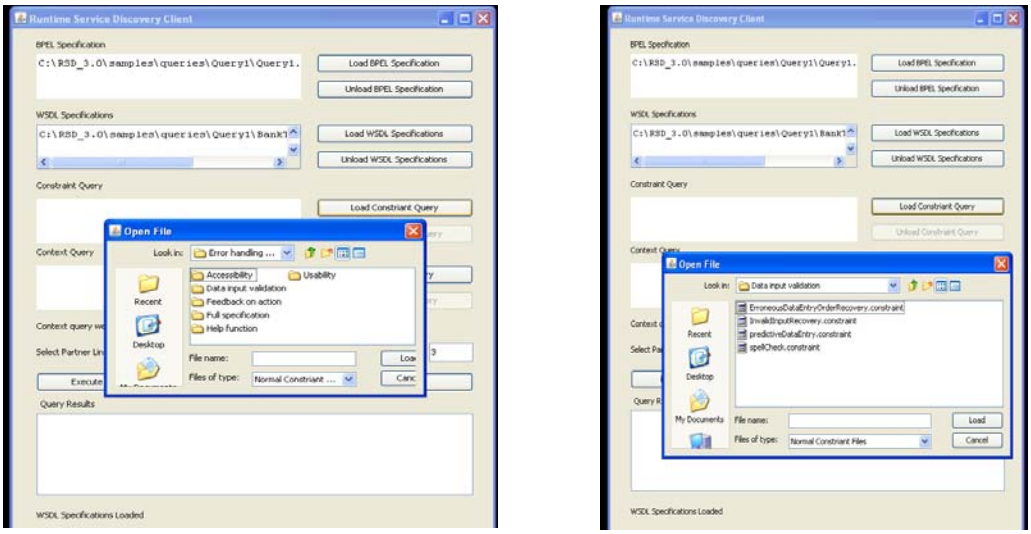
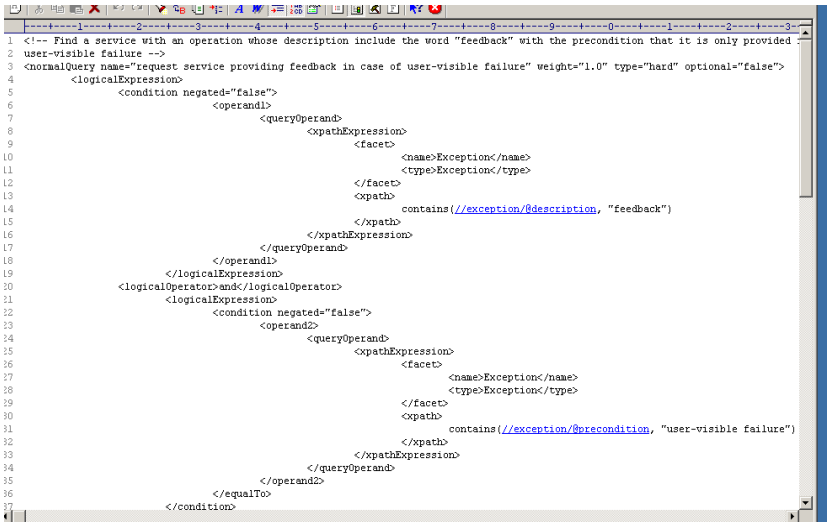


Figure 7: Error handling constraint queries sets (left) and subset (right)

Empirical evaluation of the constraint queries for RSD is ongoing at the time of writing, and individual queries (e.g. Figure 8) are iteratively being modified and refined as the testing occurs.



```
<!-- Find a service with an operation whose description include the word "feedback" with the precondition that it is only provided
user-visible failure -->
<normalQuery name="request service providing feedback in case of user-visible failure" weight="1.0" type="hard" optional="false">
        <logicalExpression>
                <condition negated="false">
                        <operand1>
                                <queryOperand>
                                        <xpathExpression>
                                                <facet>
                                                        <name>Exception</name>
                                                        <type>Exception</type>
                                                </facet>
                                                <xpath>
                                                        contains(//exception/@description, "feedback")
                                                </xpath>
                                        </xpathExpression>
                                </queryOperand>
                        </operand1>
                </logicalExpression>
        <logicalOperator>and</logicalOperator>
                <logicalExpression>
                        <condition negated="false">
                                <operand2>
                                        <queryOperand>
                                                <xpathExpression>
                                                        <facet>
                                                                <name>Exception</name>
                                                                <type>Exception</type>
                                                        </facet>
                                                        <xpath>
                                                                contains(//exception/@precondition, "user-visible failure")
                                                        </xpath>
                                                </xpathExpression>
                                        </queryOperand>
                                </operand2>
                        </equalTo>
                </condition>
```

Figure 8: Extract of a sample, single-service discovery query with the constraint: services' *exception* facet must contain the terms "feedback" in their description, and "user-visible failure" as a precondition.

3 Summary and future work

The increasing diversity of SBAs users and services' increasing use in mission-critical systems motivate the research into services fault-tolerance and robustness, this to ensure that the design and execution of SBAs strives towards minimizing the occurrence and impact of errors. This paper reported on proposed approaches for the application of codified knowledge about user errors to SBA engineering:

- For service discovery at design time, to inform the service discovery process and specifically seek to select services handling the common or likely errors a user is susceptible to make.

- For service discovery at runtime, to enact adaptation as a reaction to SBA faults possibly introduced by user errors.

User error taxonomies and interaction models were drawn upon to frame the common user error types liable to occur at various stages of a user interaction with SBAs. These were then used to derive service selection rules for the discovery of services with functionalities addressing these error types. Constraint queries were further derived for performing runtime adaptation via RSD, which we expect will be refined after the first round of empirical testing is complete. Following implementation and evaluation in the UCaRE tool, the refinement of the selection rules and possibly the creation of new rules tailored to particular domain applications is also expected. Several research directions will further research into the application of user errors for SBA engineering. The exploration and integration of factors affecting user errors, possibly by interlinking discovery rules, models about the users and their context, and the tasks performed is another promising research direction, possibly leading to the development of task models' associated common user errors for instance. Finally, novel ways for services to describe their error handling capabilities and the development of possible "user error service design patterns" encapsulating HCI recommendations for user-appropriate handling of errors are being considered.

4 Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

5 References

1. Lizcano, D., Jimenez, M., et. al.: *Leveraging the Upcoming Internet of Services through an Open User-Service Front-End Framework*, ServiceWave, Madrid, Spain, December 2008
2. Gartner research, Gartner Inc. *Hype cycle for software as a service*. Accessed January 2010 http://sharepoint.microsoft.com/sharepoint/worldwide/cn/south/SaaS/Hype%20Cycle%20for%20Software%20as%20a%20Service.pdf, 2006.
3. Chan, P.P.W., M.R. Lyu, and M. Malek, *Making services fault tolerant.* Lecture notes in Computer Science, 2006. **4328**: p. 43.
4. Papazoglou, M.P., *Web services: principles and technology*. 2008: Pearson Prentice Hall.
5. Bean, J., *SOA and Web Services Interface Design: Principles, Techniques, and Standards*: Morgan Kaufmann.
6. Aghdaie, N. and Y. Tamir, *CoRAL: A transparent fault-tolerant web service.* The Journal of Systems & Software, 2009. **82**(1): p. 131-143.
7. Pallemulle, S.L., H.D. Thorvaldsson, and K.J. Goldman. *Byzantine fault-tolerant Web services for n-tier and service oriented architectures*. 2008.
8. Liu, L., et al., *A Fault-Tolerant Web Services Architecture.* Lecture notes in Computer Science, 2006. **3842**: p. 664.
9. Gaitros, D.A., *Common errors in large software development projects.* The Journal of Defense Software Engineering, 2004. **12**(6): p. 21-25.
10. S-Cube CD JRA 1.1.2. *Separate design knowledge models for software engineering and service based computing*, 2009.
11. Reason, J., *Human error*. 1998: Cambridge University Press.
12. Cacciabue, P.C., *A methodology of human factors analysis for systems engineering: theory and applications.* IEEE Transactions on Systems, Man and Cybernetics, Part A, 1997. **27**(3): p. 325-339.
13. Sutcliffe, A. and G. Rugg, *A taxonomy of error types for failure analysis and risk assessment.* International Journal of Human-Computer Interaction, 1998. **10**(4): p. 381-405.

14. Norman, D.A. *Steps toward a cognitive engineering: Design rules based on analyses of human error*. 1982: ACM New York, NY, USA.
15. Rizzo, A., D. Ferrante, and S. Bagnara, *Handling human error*. Expertise and technology. Cognition and humancomputer interaction. Hillsdale, NJ, Lawrence Erlbaum, 1995.
16. Trainotti, M., et al., *Astro: Supporting composition and execution of web services.* Lecture Notes in Computer Science, 2005. **3826**: p. 495.
17. Arsanjani, A., *Service-oriented modeling and architecture.* IBM developer works, 2004.
18. Isaac, A., et al., *The human error in ATM technique (HERA-JANUS)*. 2003, Eurocontrol, report HRS/HSP-002-REP-03.
19. SeCSE: *Platform for Runtime Service Discovery – v3.0* http://www.secse-project.eu/wp-content/uploads/2007/10/a2d13-platform-for-runtime-service-discovery-v30.pdf

2 Kounkou, A.; Zachos, K.; Maiden, N.: *Exploiting user model information for service discovery*. – not yet submitted

Exploiting user model information for service discovery

Angela Kounkou, Konstantinos Zachos, Neil Maiden
Centre for HCI Design, City University, UK
{sbbc775, kzachos, N.A.M.Maiden}@soi.city.ac.uk

Abstract

Service-based application users are shifting from a small, often technically-minded user base to a much more inclusive, broader audience whose varied characteristics and needs must be accommodated. Although extensive research exists on the topic of user-adaptive software systems, the particular case of SBAs has not as widely been investigated as of yet. In this paper, we propose to exploit user model information to develop additional criteria and filters for selecting services based on user's characteristics, this to permit the discovery of services whose specifications match user requirements better.

1. Introduction

Software services are increasingly being used in diverse domains and settings that span business and mission-critical applications, but also personal computing and entertainment mashups. Their uptake in varied domains as well as the drive towards an Internet of Services [1] means that Service-Based Applications' (SBAs) end users are shifting from a relatively small, technical and professional user base to encompass a wider audience of users whose various needs, preferences and abilities need to be accommodated.

Consider for instance a government-provided journeyPlanner SBA that maps and displays routes to user-specified destinations. Users of such an application, although sharing a similar core goal, can reasonably be expected to have varied needs, requiring for instance a choice of output modalities (e.g. audio rather than visual output in case of sight impairment), or different annex functionalities (e.g. data validation strategies such as spell-checking for dyslexic users; multilingual outputs for users with varying language skills).

Building software that satisfies and appropriately supports end-users in their tasks is a long-standing challenge in software engineering, and it is widely acknowledged that user-tailored interactive systems support users in their tasks better than more generic, "one size fits all" systems [e.g. 2, 3]. Extensive research in the fields of personalisation and adaptation of software systems investigates the delivery of computing services customized to the characteristics of individual end–users, and various software engineering approaches leverage HCI knowledge to engineer systems in a user-centric manner, notably through the integration of User-Centred Design (UCD) techniques. SBAs' engineering and architectural particularities however put a different slant on this challenge.

SBAs are composed of services which are developed to be reusable rather than for exclusive use in specific, well-scoped applications. Further, their range of operations and end users are potentially too diverse to be foreseen: "*the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider*" [4]. Consequently, individual services' design often cannot take into

account anything but the most generic considerations about their hypothetical end user population where appropriate. Additionally, approaches to the design and implementation of SBAs are often business-driven and seldom leave space to the integration of knowledge about the end users in the engineering process [5, 6]. As a result, SBAs cannot yet deliver their functionalities in a truly user-centric way, which we argue is obstructive to a good user experience, to SBAs' effectiveness in performing their functions, and ultimately to their adoption by the wider audience.

Integrating knowledge about end users in the engineering of SBAs will help configuring systems that better fit and accommodate their user needs, hence improving the efficiency and quality of the interaction and affording such SBAs a competitive advantage compared to other services offering similar core functionalities, but a lesser user experience.
This paper proposes User Models - knowledge structures encapsulating information about users – as a means to integrate end user knowledge in the SBA engineering process, more specifically at the stage of service discovery. The remainder of the paper is as follows: section 2 describes existing HCI approaches to user modelling and reports on our choice of one such approach for SBA engineering. Section 3 reports on the codification of a user model into a form amenable to automated use for service discovery. Finally, section 4 concludes this paper and outlines planned future work.

2. User Models for SBA engineering

User models can be defined as "models that systems have of users that reside inside a computational environment" [7]; they are a system's representation of a user's properties such as personal characteristics and preferences, and have also been referred to as a "knowledge structure that represents the profile of a single (registered) user" [8]. They are classically used to impart end user information to either human designers or to computational tools depending on their structure and on the phase of software engineering they are used at. User models are most commonly used during the implementation of interactive system to focus some of the engineering considerations on the end users, and after the implementation to inform software alterations for a better fit with the users (especially in the case of adaptive and adaptable systems).

Current approaches to SBA engineering place an emphasis on business process models and notations to integrate information about the process-oriented context relevant to the SBA; little information about the human actors involved in the process is weaved into the engineering practices. Initiatives such as BPEL4People [9] attempt to address this gap by incorporating human considerations into the specification of business processes; however they stop at defining human tasks and describing them as activities, and do not impart information about their characteristics. User models can contribute to integrating more information about human actors of business processes that are end users of the SBAs.

Numerous approaches to modeling users exist in the field of Human Computer Interaction (HCI), and depending on their intended use, user models can widely differ in their representation and focus. In order to identify those that were amenable for use in an automated environment for the engineering of SBAs, we investigated the user characteristics found to be informative of system design in the HCI literature (e.g. [10, 11, 12, 7]), and retained characteristics which we found fell in 5 broad areas of affinity:

1 *Personal* – basic demographic and identification data about an individual user (e.g. name, d.o.b., address)

2 *Skills* – information about academic and professional skills (e.g. qualifications obtained, employment sector and job role, languages spoken)

3 *eActivity* – information about users' previous service use if any (e.g. services used, frequency, trusted providers) and online activities (e.g. social network accounts, browsing topics).

4 *Preferences* – information about the user's preferences regarding security and privacy settings, localisation options, and interaction modes with software systems

5 *Accessibility* – information about the user's abilities, including eventual mental and physical impairments (age-related or otherwise e.g. sight impairment, dyslexia).

These were examined against existing approaches to modelling users identified from the literature:

1 *User stereotypes*, descriptive enumerations of a set of traits that often occur together [13].

2 *User profiles*, which usually comprise a brief, mostly demographic description of users. [14]

3 *User roles*, focused collections of users' characteristic needs, interests, expectations, and behaviours in relation to a particular system [15].

4 *Cognitive user models*, specifications of the cognitive processes occurring during the execution of computer-based tasks that help designers understand how the human mind works [16].

5 *Personas*, rich descriptions of fictitious individuals, their goals and behaviour that are based on patterns of use of real users and written in natural language [17].

We identified user profiles as able to encapsulate the needed information while still being amenable to use in automated environments, this without drastic changes in their structure (as would be the case with personas). Unlike user roles and cognitive user models, they also avoided a too narrow focus on specific aspects of the user. Finally, they permitted the expression of richer nuances and evolutions in the users' characteristics than stereotypes.

3. User model codification

In order to model the relevant categories of information identified (Section 2), we developed a user model as a simple, faceted representation of aspects of the user as illustrated below.

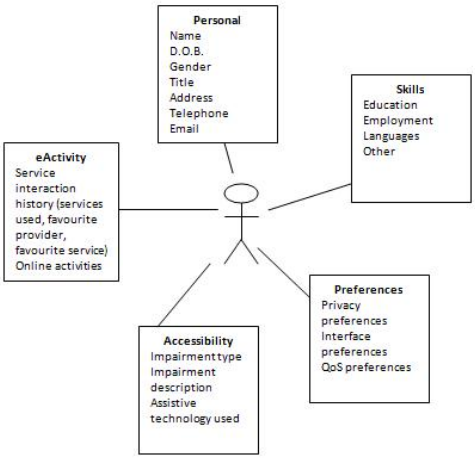


Figure 1: faceted user profile

The broad categories identified (Personal, Skills etc) were expanded with attributes detailing user knowledge for each fragment as shown in Figure 9. For instance, the "*skills*" category was expanded to encompass information about a user's academic or professional education and qualifications, his employment history, and his language skills; the "*other*" attribute provided an amount of flexibility to allow the specification of relevant information not already covered.

As the profile information attributes were established, sets of plain-text heuristics were developed that reflected likely impact areas of user model information on service selection rules (Figure 2). For instance, the *"skills"* facet provided information about the end-user employment including job title and organisation, which were amenable to influencing the choice of registries to search for relevant services for the user (e.g. use of specialised registries as appropriate depending on the job title), or additional requirements for the services depending on available information on the user's organisation (industry, preferred providers, organisational values and culture).

UM facet	Personal	Skills	eActivity	Preferences	Accessibility
Selection rules	Gender-related (e.g. gender differences for spatial visualisation services); Age-related (e.g. age related impairments, QoS prioritisation); Localisation -related (e.g. required national standards' compliance)	Role-related (e.g. specialised service registries to be used) Organisation-related (e.g. QoS prioritization by organisation culture) Abilities-related (e.g. service operation language)	Preferences-related (e.g. trusted services; trusted providers)	Privacy-related (e.g. prohibited data sharing) Security-related (e.g. required standards' compliance; QoS prioritisation) QoS-related (e.g. QoS prioritisation)	Accessibility-related (e.g. standards compliance)

Figure 2: user model facets' areas of relevance for service selection rule

For each attribute and user facet, these simple heuristic rules – e.g.: *"If a user is above 65, then select services with a help function, that are compliant with accessibility standards, and that have a high learnability. Else if a user is below 18, then select services with appropriate age rating"* expressed the alterations or qualities an SBA would ideally make or have (respectively) to best match a user's requirements. For instance, the age-related heuristics specified that users whose birth date indicated were over 65 years old should be suggested services that: offered a help functionality, were compliant with relevant accessibility standards, and had a high learnability as these were valued and helpful qualities and requirements of such users [18]. Users under 18 on the other hand, ought to be restricted in the services available to them (no access to gambling services for underage individuals for instance).

The profile information had to be "machine readable" to allow for use in appropriate phases of SBA engineering. Similarly, the simple, plain-language heuristic rules developed had to be encoded in service tools to allow for their automated use during the engineering process. The FP6 SeCSE Integrated Project produced a range of open-source service engineering tool suites, and we opted to use its SeCSE service discovery environment for SBAs due to the availability and accessibility of the platform, which we built upon and extended to allow for the codification and investigation of user models in service discovery.

3.1 User model information for service discovery

This section explores applications of user models information to design-time service discovery using the UCaRE tool, which it briefly introduces.

Service Discovery is the process of finding services that match the requirements of the service requestor. Service discovery in the SeCSE development environment can be performed using UCaRE, a module that generates service request queries from use cases and requirements specifications. The tool is detailed in [19]; as an overview, EDDiE, the algorithm for service discovery used to formulate these query requests, performs the following 4 key steps: 1) *Natural language processing*; 2) *Word sense disambiguation* - the algorithm attempts to define the correct meaning for each term; 3) *query expansions* - each term is expanded with other terms having similar meaning; and 4) *query matching* for the discovery of services matching the expressed constraints. The EDDiE algorithm was extended to permit the inclusion of user model information in the SeCSE discovery process.

First, a set of user model facets codifying our user models using XML-based templates (e.g. see the "personal" facet's implementation in Figure 3 below) was developed and added to the UCaRE repository using an implementation of an eXist XML database. During *query matching* (step 4), this permitted the EDDiE discovery algorithm access to user model information in order to perform the matching of terms in the service request to terms in the user model facets.

```
<LanguageSpecificSpecification>
  <FacetType>Personal</FacetType>
  <ReferencedOntology />
  <ReferencedSIM />
  <FacetSpecificationLanguage>XML</FacetSpecificationLanguage>
  <FacetSpecificationOwner>City</FacetSpecificationOwner>
  <FacetSpecificationLastEdited>03 Dec 2009</FacetSpecificationLastEdited>

  <FacetSpecificationData>
        <PersonalSpecification>
              <PersonalOntologyReferences>
                    <Reference />
              </PersonalOntologyReferences>
              <PersonalCharacteristics>
                    <Name>
                          <FirstName />
                          <MiddleName />
                          <lastName />
                    </Name>
                    <DoB />
                    <Gender />
                    <Title />
                  - <Address>
                          <HouseNumber />
                          <StreetName />
                          <Town />
                          <Postcode />
                          <Region />
                          <State />
                          <Country />
                      </Address>
                    <Telephone />
                    <Email />
                    <Other />
              </PersonalCharacteristics >
        </PersonalSpecification>
    </FacetSpecificationData>
</LanguageSpecificSpecification>
```

Figure 3: codified user model facet

User model rules - new discovery rules expressing our heuristics were then developed and added to the repository of SeCSE service queries (Figure 4 below).

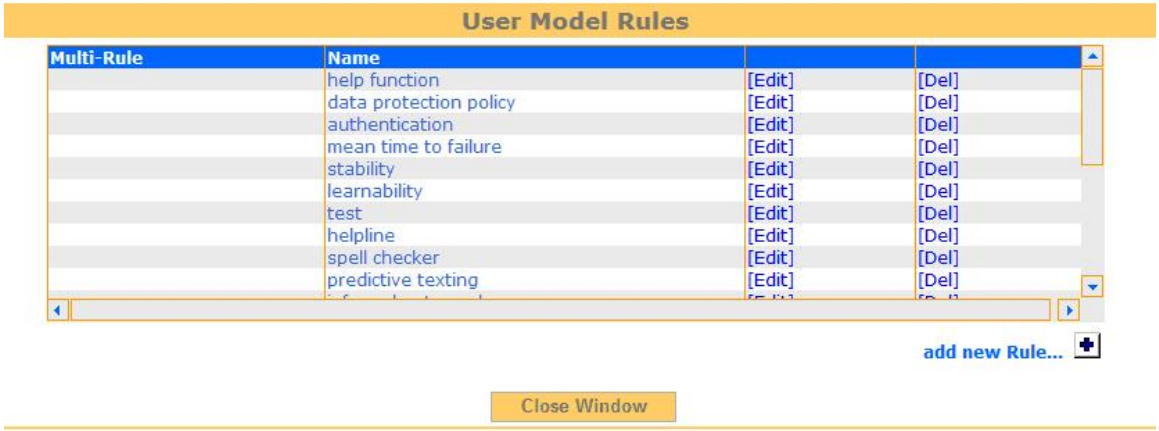


Figure 4: User model rules implemented in UCaRE

The plain text heuristics were formalised into these user model rules by: specifying the relevant user model facet and providing the path to its relevant node; linking to the relevant service description facet [20] by providing its path node for the verification of the rule; specifying the type of verification to be performed (e.g. existence of a criterion, ranges the criterion had to be within); and suggesting additional keywords for query expansion, this for each of these rules (e.g. of one rule Figure 5 *a* and *b* below).

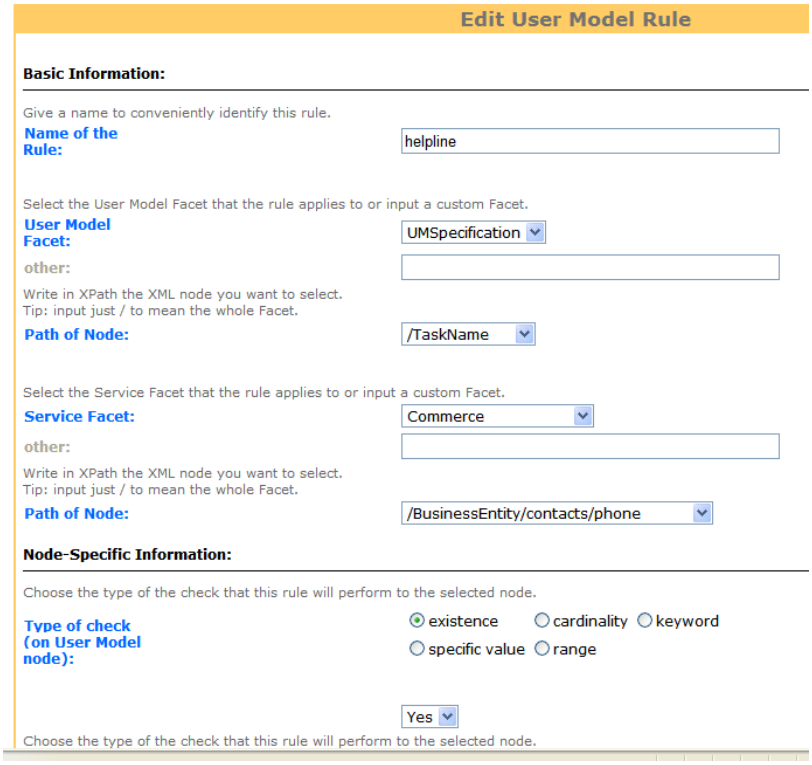


Figure 5a: example of a user model rule in UCaRE

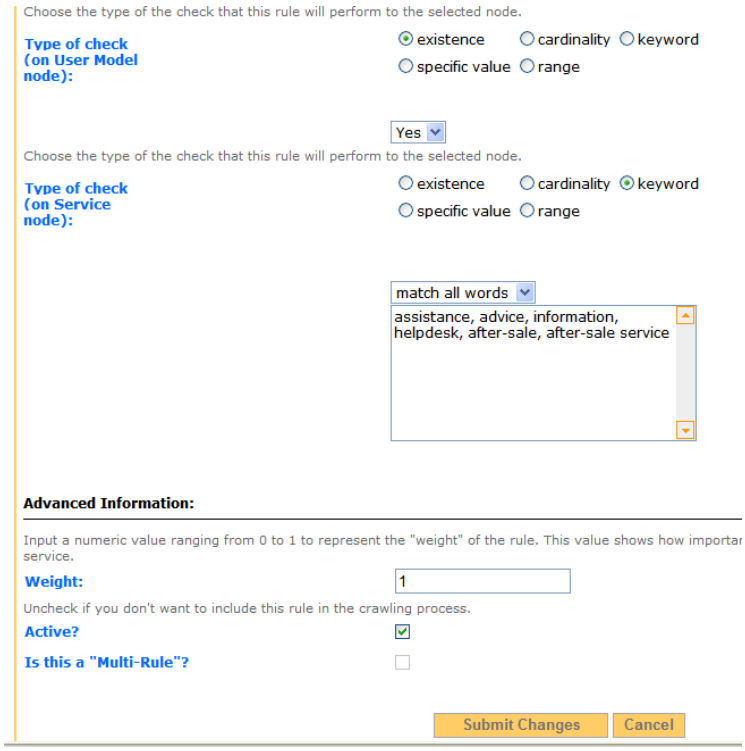


Figure 5b: example of a user model rule in UCaRE (continued)

The rules specified additional constraints on selection, the required terms and their locations in both the stored user model facets and in the service description facets for matching. In Figure 5 for instance, if the user model specified the need for a helpline to be available for the service, the existence of such function or similar ones (e.g. assistance, helpdesk) could be ascertained in the Commerce facet of the services under the path *BusinessEntity/Contacts/Phone* to verify the existence of such contact details. The rule could further be specified to be *active*, i.e. executed during the service discovery process, and its weight set between the range 0 – 1 to specify it relative importance. Overall, the codified user model heuristics and the availability of the user model information provide additional criteria and filters for the selection of services based on user's characteristics, this to permit the discovery of candidate services in the registry whose specifications match the user model requirements.

The empirical evaluation of this approach to using user model information is currently ongoing; a preliminary appraisal however suggests it will be proven to have a beneficial impact on service discovery, this by permitting a better match between the user requirements and the discovered services' functions.

4. Summary and Future Work

This paper reported on our suggested use of user models for the discovery of services matching better their end users requirements. Our proposed approach is being tested using the SeCSE service discovery module UCaRE, which has been extended with user model facets encapsulating user model information, and new user information-based rules for service discovery.

Along with empirical evaluation, further lines of research are either ongoing or being considered. Current HCI-aware approaches for SBAs adaptation monitor users' interactions with the application: the actions taken by the users ("user event streams") are the inputs for an analysis to infer the users' intentions, then correspondingly adapt the SBA as appropriate. We suggest that user model information is another valuable source of requirements for SBAs' implicit adaptation, and could trigger adaptation or personalisation of the SBA upon changes in the user model. In this case of figure, part of the monitoring focus would be on a user's user model rather than on services only. We are currently investigating mechanisms for 1) monitoring and detecting changes in user models 2) mapping them to relevant user requirements heuristics and 3) linking them to relevant services or service operations within an SBA for assessment of whether the user's needs and requirements are still being supported after the change, triggering the adaptation if that is not the case. Another area being explored is the application of user model discovery rules for runtime service discovery, rather than only at design time as is the case with our suggested approach.

5. Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

6. References

[1] Lizcano, D., Jimenez, M., et. al.: "Leveraging the Upcoming Internet of Services through an Open User-Service Front-End Framework", ServiceWave, Madrid, Spain, December 2008

[2] B. Mehta, "Learning from What Others Know: Privacy Preserving Cross System Personalization," Proceedings of the 11th international conference on User Modeling, Corfu, Greece: Springer-Verlag, 2007, pp. 57-66.

[3] Ford, G., Gelderblom, H.: "The Effects of Culture on Performance Achieved through the use of Human Computer Interaction". Proceedings of SAICSIT, 2003, pp. 218-230.

[4] C. MacKenzie et al.: Reference Model for Service Oriented Architecture 1.0 – OASIS Standard. [Online], 2006 Available at: http://docs.oasis-open.org/soa-rm/v1.0/ [Accessed December 2009]

[5] BEA: SOA Practitioners' Guide Part 3, Introduction to Services Lifecycle. [Online]. Available at: http://dev2dev.bea.com/2006/09/SOAPGPart3.pdf [Accessed July 2009]

[6] SOMA: Service Oriented Modelling and Architecture. [Online]. Available at: http://www.ibm.com/developerworks/library/ws-soa-design1/ [Accessed July 2009]

[7] Fischer, G.: User Modeling in Human-Computer Interaction, Journal of User Modeling and User-Adapted Interaction (UMUAI) 2001, Vol. 11, No.1/2, pp.65-86.

[8] C. Stary, "User Diversity and Design Representation: Towards Increased Effectiveness in Design for All", Universal Access in the Information Society, vol. 1, no. 1, pp. 16-30, June 2001.

[9] Adobe Developer Connection, “BPEL4People overview”. [Online]. Available at: http://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html [Accessed December 2009]

[10] Bellekens, P.A.E., Houben, G.J.P.M., Aroyo, L.M., Schaap, K., Kaptein, A. (2009). “User model elicitation and enrichment for context-sensitive personalization in a multiplatform TV environment”. Proceedings of the 7th European Interactive Television Conference (EuroITV'09, Leuven, Belgium, June 3-5, 2009). (pp. 119-128). New York NY: ACM.

[11] Abbattista, F., N. Fanizzi, S. Ferilli, P. Lops and Semeraro, G.: “User Profiling in an Application of Electronic Commerce”. In F. Esposito (Ed.), AI*IA 2001: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence 2175, 87-98, Springer: Berlin, 2001.

[12] Carmagnola, F., Cena, F., Cortassa, O., Gena, C., and Torre, I.: “Towards a tag-based user model: How can user model benefit from tags”, 11th International Conference on User Modeling, 4511/2007, (2007).

[13] Rich, E. (1999). Users are individuals:- individualizing user models. /International Journal of Human-Computer Studies, 51, 323-338.

[14] Teixeira, C., Sousa Pinto, J., & Martins, J. A. “User profiles in organizational environments” (2008). In Campus-Wide Information Systems, 25, pp. 128 – 144

[15] Constantine, L. (2005). “Users, roles, and personas” [Online]. Available at: http://www.foruse.com/articles/rolespersonas.pdf [Accessed April 2009]

[16] Cairns, P. and Cox, A.L. “Research Methods for Human-Computer Interaction”. (2008). Cambridge: Cambridge University Press

[17] Cooper, A., Reimann, R. & Cronin, D.: “About Face 3.0: The Essentials of Interaction Design”. (2007). Indianapolis: Wiley

[18] Zaphiris,P., Kurniawan, S., and Ghiawadwala, M.: “A Systematic Approach to the Development of Research-Based Web Design Guidelines for Older People”. (2007). In Universal Access in the Information Society, 6(1), pp. 59–75

[19] Zachos, K., Maiden, N.A.M., Jones, S., and Zhu, X., “Discovering Web Services To Specify More Complete System Re-quirements” Proc. 19th Conference on Advanced Information System Engineering (CAiSE'07), 2007. pp.142-157.

[20] Walkerdine, J., Hutchinson, J., Sawyer, P., Dobson, G., and Onditi, V.: "A Faceted Approach to Service Specification", Proc. 2nd Int’l Conf. on Internet and Web Applications and Services (ICIW 07), Mauritius, 2007.

3 Zachos, K.; Kounkou, A.; Maiden, N.: *Codified Knowledge about User Task Modelling Applied to Service Discovery*. – not yet submitted.

Codified Knowledge about User Task Modelling Applied to Service Discovery

KonstantinosZachos, Angela Kounkou, Neil Maiden
Centre for HCI Design,
City University, UK
{kzachos,sbbc775,N.A.M.Maiden}soi.city.ac.uk

***Abstract*—*Approaches to the engineering of Service-based Applications (SBAs) comprise phases for the analysis of applications functionalities and business processes in order to inform their implementation as services. The human element however is not specifically taken into account, and HCI task modeling techniques may contribute to the output of these phases by focusing on the tasks and the human users rather than on the system in place and the processes to be. This paperreports the application of user task models as part of a new requirements-based service discovery approach to integrate human actors in the discovery and selection of SBAs. It describes how user task models will enhance the service discovery process based on a task knowledge base that is currently being populated, then demonstrates these enhancements with an example from web services development in the E-Government domain.*

1. Introduction

Service-Oriented Computing (SOC) has been tipped as a major transformation to software development [1] but it has yet to fulfil its potential.Some of the obstacles it will have to overcome in order to reach widespread adoption point at human and organizational issues rather than technological ones [2].As well as being accessed and used by other services or applications, services and Service-based Applications (SBAs) can require interaction with human users taking part in the business process enabled by the service, or imparting human intelligence to the relevant services (e.g. the Amazon Mechanical Turk web service [3]).Until now Human-Computer Interaction (HCI) knowledge was treated in isolation from service engineering techniques for developing SBAs. Humans are present in SOC as end users and consumers, but also as service providers [cf. Human-provided services] and designers. However, human specificities, diversity and tasks characteristics – properties that could be powerful drivers for SBAs configuration and personalization – are currently not taken into account in SBA design and delivery.Thus, new ways of thinking are needed toexplicitly relate users with service discovery and service composition techniques.

HCI task modeling techniques can contribute to the output of these changes by focusing on the tasks and the human users rather than on the system in place and the processes to be. They canalso support the definition of appropriate level of granularity, and functional service cohesion for SBAs by helping to clearly scope and define tasks for their later implementation as service operations.Tasks models enable designers to represent and manipulate a formal abstraction of the activities that ought to be performed in order to reach user goals. Each task can be decomposed into sub-tasks.Typical characteristics include their "hierarchical structure, the task decomposition and sometimes the temporal relationships between elements" [4]. In SBAsuser tasks should be supported by services as demonstratedin the following simple user task example taken from an E-Government case studythat pertains to citizens accessing government services online [5]. An end user requeststo use a governmental journey planning service.The underlying task model describes the context in which candidate solution services will be needed to fulfil the user goal, i.e.to receive personalised suggestions of available routes to a particular destination. The task model decomposes the journey planning user goal into sub-tasks and identifies the sub-tasks that solution services might deliver, for example sending journey planning request, receiving route suggestions, and following the directions. For this to happen these services need to be discovered. Service discovery is a critical challenge in SBAs. During the use of SBAs new services need to be discovered if these services become available or currently invoked services need to be replaced by other services with improved qualities such as performance and reliability. Processes and techniques for service discovery have been researched extensively in previous projects [e.g. 6, 7]. However, none of these processes and techniques explicitly use knowledge about user tasks to refine the discovery and selection of services appropriate to the user task.

This paper describes how user task models can further contribute to the integration of human actors in the discovery and selection of SBAs by providing richer, more contextual descriptions and models than those currently available. The remainder of this paper is in 4sections. Section 2 describesexisting user task models and their use in SBA engineering. Section 3 describes the task-based service discovery solution and demonstrates the it with a simple example. The paper ends with a review of progress so far and future work.

2. User Task Models for SBA Engineering

A task model describes structured sets of activities [8] -

often in interaction with a system influenced by its contextual environment - that the user has to perform to attain goals [9]. In cognitive psychology, task models are a means for formally describing human problem solving behavior [10].

Task models have been used in various approaches to support different phases of the software development life cycle, including requirement analysis and usability evaluation [11]. Task models help understanding how people currently work [12] so they can function as a requirements elicitation device and indicate how activities should be performed in an application being designed. They allow for the software design to be described more formally, analysed in terms of usability, and be better communicated to people other than the analysts [13, 12].

Current service-oriented approaches rely primarily on business process models and notations such as BPEL to indicate the process-oriented context in which a service needs to be invoked; however process models normally lack other important information about the actor who is performing the process and his actions.This fact has been recognised by initiatives such as BPEL4People [14], which attempts to incorporate human considerations into the specification of Business Processes. It does so by defining human tasks and describing them as activities, but stops at rendering human activities as simple processes without addressing their context. Task models have semantically richer models and notations and may provide more context-specific information to inform SBAs' modeling: their specific semantics permit different possible uses of task models in service-centric systems; for example tasks and operations in HTA [15] can map to specific service capabilities, thus enabling finer-grain service discovery and selection. Likewise, the distinction between abstract, interaction, application and user tasks in CTT [16] can be mapped to different service types, thus informing more effective service composition, especially if finer-grain services are available.

One goal of the presented research is to create task ontologies for modeling real world user activities. To avoid the ontology-modeling bottleneck that often inhibits ontology-based solutions we are seeking to extract task knowledge that can be reused. Our approach is to identify task knowledge that is domain-specific, then extract the domain-independent task knowledge that can be reused, similar to the KADS approach to knowledge modeling. For instance, domain-independent task knowledge that describes how to "go to *somewhere*", describes a general process model to perform the activity of moving from a starting point to a destination, is common knowledge among specific task knowledge regarding going to specific places from specific places. Such domain-independent task knowledge can be used to describe more specific task knowledge in a new domain that decreases the cost for expanding the coverage of task knowledge.This paper describes how user task models can further contribute to the integration of human actors in the discovery and selection of SBAs by providing richer, more contextual descriptions and models than those currently available.

3. Codifying User Task Models during Query Reformulation

In this section we describe the association between user task models, software services and service-based applications in more detail, and demonstrate the use of the CTT task modelling formalism to represent knowledge about user tasks and introduce this knowledge to inform service discovery and selection. We opted to model user tasks using CTT due to the engineered approach to task models, thus overcoming the limitations of other approaches, which either had no or very limieted tool support or used unsuitable notations with limited operators. We selected the SeCSE service discovery environment for service-based systems upon which to design and implement codified HCI knowledge. The FP6 SeCSE Integrated Project [6] is one of the cornerstone research development projects in service-centric systems funded by the European Commission. It has produced substantial research, development and evaluation results, as well as tool suites available to be extended in S-CUBE. We extended SeCSE service queries specified using XQuery and selection filters with types of knowledge about user task knowledge.

3.1 Service Discovery in SeCSE

Before outlining the approach, we briefly describe the current SeCSE algorithm for service discovery, called EDDiE, which is used to develop the task-oriented service discovery extension. EDDiE formulates service requests from use case and requirements specifications expressed in structured natural language [17]. This section summarizes the algorithm's description. A full description is provided in [18]. The algorithm has 4 key components;the *Natural Language Processing*, *Word Sense Disambiguation*, *Query Expansion* and the *Matching Engine*. In the first the service request is divided into sentences, then tokenized and part-of-speech tagged and modified to include each term's morphological root (e.g. *driving* to *drive*, and *drivers* to *driver*). Secondly, the algorithm applies procedures to disambiguate each term by defining its correct sense and tagging it with that sense (e.g. defining a *driver* to be a *vehicle* rather than a *type of golf club*). Thirdly, the algorithm expands each term with other terms that have similar meaning according to the tagged sense, to increase the likelihood of a match with a service description (e.g. the term *driver* is synonymous with the term *motorist* which is also then included in the query). In the fourth component the algorithm matches all expanded and sense-tagged query terms to a similar set of terms that describe each candidate service, expressed using the *ser-*

vice description facet, in the SeCSE service registry. Query matching is in 2 steps: (i) XQuery text-searching functions to discover an initial set of services descriptions that satisfy global search constraints; (ii) traditional vector-space model information retrieval, enhanced with WordNet, to further refine and assess the quality of the candidate service set. This two-step approach overcomes XQuery's limited text-based search capabilities.

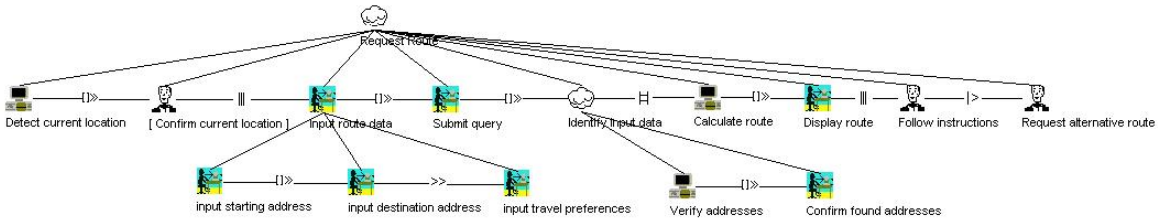


Figure 1. Request Route example expressed as a CTT Task Model

3.2 Task-based Extension to Service Discovery

In the SeCSE's current service discovery environment, query expansion alone cannot resolve the semantic mismatch problem that arises because the problem request and solution service are inevitably expressed using different ontologies. To overcome this ontological mismatch, we are extending the algorithm with user task model libraries that encapsulate knowledge about classes of proven service solution to classes of user tasks. As such, task-oriented service discovery supports the user in finding appropriate services by querying a rich task knowledge base (Task KB) that represents common sense knowledge about typical complex tasks. Throughout the paper we demonstrate key elements of the approach using the request route example scenario taken from the E-Government case study [5].The actors involved are an end user requesting and using a governmental journey planning service, and a public body providing the service. The user sends a journey planning request with details about the start, end point and travel preferences for his journey. She receives personalised suggestions of routes to follow thatshe can query for additional details; relevant travel alerts; and dynamic re-mapping of his route as needed.

3.2.1 Structure of a User Task Model

A user task model defines a reusable and generic task structure that encapsulates a well-defined functionality for a recurrent design problem in task modeling [11]. In S-Cube we employ this definition to describe: (i) classes of tasks that re-occur during the design of SBAs, and: (ii) classes of candidate service solutions proven to solve these tasks.Figure 2describes the structure of each user task model in the Task KB that we introduced to support task-based extensions to service discovery.

Each user task model specifies classes of service that transform the service request and are matches to discover instances of new software services in service registries. User task models(1) contain descriptions for abstract as well as concrete tasks and their interrelations as semantic descriptions that have the potential to be compliant with requirements that are instantiated as service queries during early service discovery; (2) define task-specific categories that are compliant with classes of software service.

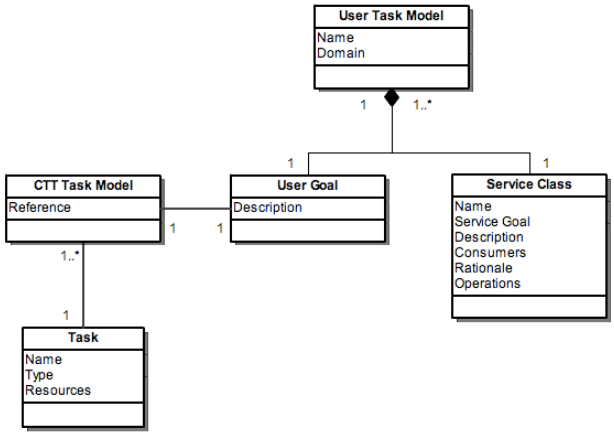


Figure 2. Outline task model schema expressed as a UML class diagram

More specifically, each user task model is an aggregation of 2 parts – the user goal part and the service class part. The user goalpart includes a structured natural language description of a generic task in context and the task structure expressed in CTT [11]. For each single task, it is possible to directly specify a number of attributes and related information as shown in Figure 2. Apart from the task name, we distinguish between four different task types: abstract tasks, which are further decomposed, user tasks – tasks to be carried out by the user –, interaction tasks – tasks to be carried out by the interaction of a user with a software system – and application tasks representing those tasks, which are fully supported by software system [16]. The sequence of different task is describedby operators[1].Finally, each task may contain one or more resources that are needed for fulfilling the single task. Figure 1 shows an excerpt of the CTT task model of the

[1] For a full list of operators refer to [11]

request route example scenario.

The service class part of the user task model describes the behaviour of candidate classes of software serviceinstructured natural language descriptions that might be invoked to deliver the behaviour specified in the user goal part. That means that the service class part contains one or more typical classes of services that are proven solutions to the generic task and previously associated with the task. During request reformulation, EDDiE can reformulate one or many service requests for each service class specified for a matched user task model and selected by the analyst. Table ?describes one such class – *MobileNetworkProvider*–that a SBA might invoke service instances of to deliver the behaviour specified inthe request route user task model.

Service	MobileNetworkProvider
Service Description	The service is able to identify/detect/discover a wireless/mobile network provider which can establish a mobile point-to-point connection/link.
Service Goal	Given the requested country and preferences, the service returns the provider which operates a mobile network that satisfies the cost, availability and channel type requirements for the connection.
Service Rationale	Due to the often complex pricing of network providers it is challenging to always select the suitable provider, especially in terms of costs. Furthermore, cross-border travelling requires knowledge of cooperating roaming partners and their technical compatibility.
Service Consumers	- Server - Administrator
Service Operations	- Identify all available mobile network providers. - Compare available network providers in terms of price, coverage, availability and compatibility. - Suggest alternative mobile network providers.

Table 1. Description of the *MobileNetworkProvider* service class

During task matching EDDiE matches terms in the service request's use case attributes to terms in the user goal part's description attribute. The service class part specifies 6 attributes that correspond to 6 attributes of a service's description facet in SeCSE registries [19]. During request reformulation EDDiE replaces or extends terms in a service request with corresponding terms in a service class. For example, service class terms for the *service goal* replace or extend *use case name* terms in the service request, and *service consumer* terms replace or extend *actor* terms in the service request.The next section describes the conceptual model of the task-based service discovery approach.

3.2.2 Conceptual Model of the Task Model Extension

As Figure 3shows we extend the current SeCSE's service discovery algorithm by adding a Task KB and two new components – the task navigator and query reformulator – to EDDiE.

The Task KB is stored using eXist, an Open Source native XML database featuring index-based XQuery processing, automatic indexing, and tight integration with XML development tools. Each user task model has a unique identifier and name that are used to reference the user task model parts. The database was implemented to 'mimic' SeCSE service registries also implemented in eXist, thus enabling the catalogue to be prototyped without changing the implementation of EDDiE's service discovery algorithm.

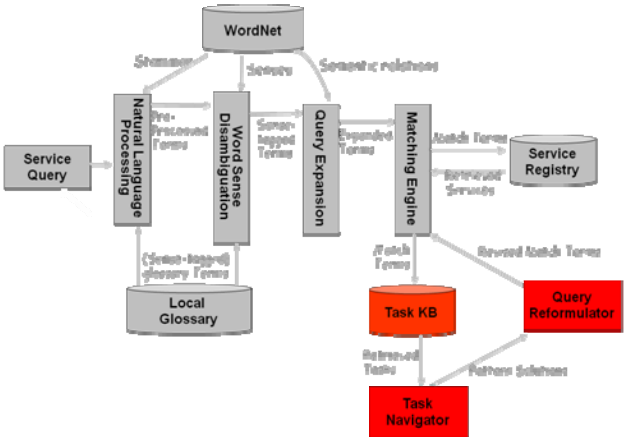


Figure 3. SeCSE'sservice discovery algorithm enhanced with task knowledge

Inputs are one more expanded and disambiguated terms in a service request, and output is one or more new service queries that have been reformulated using retrieved user task models. Task-based service discovery is in 3 stages:

1. *User Task model match*: EDDiE uses its existing algorithm to match the expanded and disambiguated service request to the description part of each user task model. The result is an ordered set of retrieved user task models that match the service request;
2. *Reformulate service request*: EDDiE uses the described classes of service in the solution part of each retrieved user task model to generate one or more new service requests that are expressed in terms of the service features rather than consumer requirements. The analyst can then modify these reformulated requests as needed;
3. *Service match*: the service discovery algorithm uses each revised service requests to discover candidate service specifications in service registries. The result is an ordered set of service specifications that match to the revised service request.

In the following sections we demonstrate this task-based extension to service discovery to reformulate service requests for the enhanced route planner application.

3.2.3 *The Request Reformulation Algorithm*

During the reformulation of service requests, if the terms in each query generated from the service request match terms in the user goal part of a user task model, the service request is extended or replaced with new text describing candidate service classes from the service class part. As well as overcoming the ontological mismatch problem, reformulation re-scopes each request to the granularity of each available class of service. Service consumers often express coarse-grain requirements that can only be solved by composing multiple service instances of multiple service classes that can only be discovered by more than one service request. Therefore each user task model specifies more than one possible class of solution service for each class of requirements problem.

The basic request reformulation algorithm is specified in Figure 4. The algorithm automatically replaces or extends terms in the original service request depending on a decision made by the analyst and recorded through the interactive dialogue. Each attribute of the service request is extended or replaced in turn, depending on the decision. Once the analyst selects the reformulation strategy, EDDiE implements the algorithm for the entire service request. The current version implements a coarse-grain version of the algorithm that we plan to refine through future evaluation studies.

```
If (OptionOfReplacingServiceRequest==true) then
Foreach (Element in Original Service Request)
      If (CorrespondedElementOfServiceClass != null) then
CorrespondedElementOfRevisedServiceRequest
          =CorrespondedElementOfServiceClass
else
CorrespondedElementOfRevisedServiceReques = none
 Else if(OptionOfExtendingServiceRequest==true) then
Foreach (Element in Original Service Request)
    If (CorrespondedElementOfServiceClass != null) then
CorrespondedElementOfRevisedServiceRequest
=ElementOfOriginalServiceRequest+CorrespondedElementOfService
Class
else
    CorrespondedElementOfRevisedServiceRequest
    = ElementOfOriginalServiceRequest
```

Figure 4. The request reformulation algorithm

The algorithm extends and replaces attributes of a service request with attributes in the service class according to attribute type mappings. These mappings provide a one-to-one correspondence between each attribute of a use case specification and a service description [19]. Indeed, during service publication in SeCSE, we propose that service providers document new services that are published on service registries using use cases. For example, an actor in a use case specification corresponds to an actor in a service request that corresponds to a service consumer in a service class. Likewise a use case précis in a specification corresponds to a use case précis in a service request that corresponds to a short service description in a service class.EDDiE presents each candidate service class to the analyst who can choose the classes to use to reformulate the request - one new service request is generated for each selected class.

3.2.4 *Dialogue for Selecting Task Models and Reformulating Requests*

Consider the following example service requestexpressed in XML (Figure 5)generated using UCaRE [20] from a use case that describes an enhanced route planning application.

```
<?xml version="1.0"?>
<ServiceRequest serviceRequestID="865">
    <UseCase useCaseID="105">
        <UseCaseName>Enhanced route planning
        </UseCaseName>
        <Precis>A chauffeur must take his passenger to an
        important meeting. The chauffeur enables the journey
        planner service from the onboard system. The
        chauffeur enters the start and destination
        coordinates as well as the required arrival time
        into the onboard system. The journey planner service
        displays the price and requests payment details. The
        chauffeur enters the company's credit card details.
        The onboard system generates the route plan and
        delivers it to the chauffeur. The chauffeur follows
        the directions to arrive at the requested
        destination.
        </Precis>
        <ProblemStmt>Current route planning systems do not
        use local, up-to-date knowledge about route
        conditions that inform the route planning process.
        </ProblemStmt>
        <Assumptions>Company's chauffeur will be prepared to
        use and pay for enhanced route planning service.
        </Assumptions>
        <FunctReq FcID="84">
            <FunctDescription>The chauffeur shall be able to
            select journey plans with different price plans.
            <FunctDescription>
        <FunctReq>
    </UseCase>
</ServiceRequest>
```

Figure 5. Part of the service request for use case *enhanced route planning*

EDDiE can match a service request to one or more user task modelsin a Task KB. Interaction between the analyst and the environment is neededto select which task is a correct abstraction of the specified problem. To implement this interaction we extended functionality in the original Service Explorer component of UCaRE [20] to support user task model comprehension and selection.

EDDiE presents matched user task models to the analyst in an ordered list through the Task Navigator. The list is ordered according the semantic similarity score for the match between the service request terms and user goal part. The analyst can select to view details of each listed task in order to understand it, then select or reject it as a correcttask behaviourof part of the fired service request. Figure 6shows the presentation of the Request Route user task model including the CTT task model and descriptions of classes of service that can be invoked to undertake the described task behaviour. If more than one service class is specified for the task – as in this case – the analyst can interact with the environment to select the most appropriate service class.

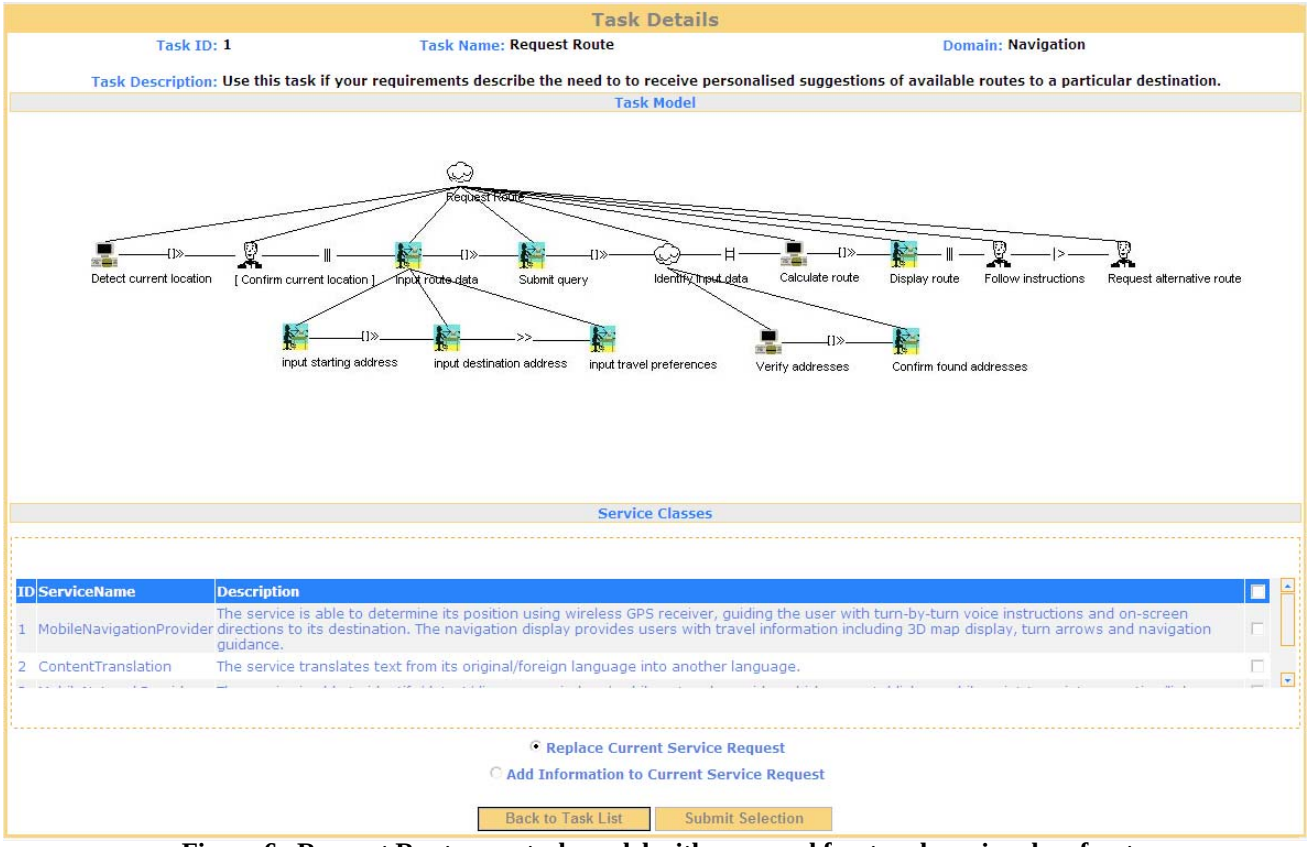


Figure 6. Request Route user task model with user goal facet and service class facet

Interaction is also needed to direct the automated service request reformulation algorithm. As mentioned earlier, the Task Navigator offers the analyst two choices – replace the current service request with terms from the selected service class, or extend the current request by adding terms from the selected service class to terms in the original request. This is handled through simple radio button controls that the analyst selects between – also shown in Figure 6. These two simple reformulation strategies were implemented to enable a first-cut validation of task-based service discovery. Clearly more sophisticated and potentially more effectivestrategies are possible. These are discussed at the end of the paper.

According to the selected reformulation strategy the revised service request warrants comparison with the original request shownin Figure 5. For example the original use case précis attribute includes nouns such as *chauffeur*, *passenger*and *meeting*as well as compound nouns such as *destinationco-ordinates*and *paymentdetails*. It also includes verbs such as *deliver*and *display*.The reformulated service request includes nouns such as *travel* and *navigation*, and verbs such as *guide*. These terms can contribute to the similarityvalue of the terms that describe relevantsoftware services that populate existing service registries, and therefore more likely to lead to successful service discovery from queries fired by EDDiE at these registries.

4. Discussion and Future Work

This paper described the association between user task models, software services and SBAs, and demonstrated the use of the CTT task modelling formalism to represent knowledge about user tasks and introduce this knowledge to inform service discovery.

We are currently building a prototype Task KB. In the first stage we are eliciting domain-specific knowledge that describes service-centric solutions for known tasks in thenavigation domain based on the S-CubeE-Government case study scenarios. We will then extract the domain-specific task knowledge to generate domain-independent task knowledge that can be reused, similar to the KADS approach to knowledge modeling. Such domain-independent task knowledge can be used to describe more specific task knowledge in a new domain that decreases the cost for expanding the coverage of task knowledge. Two important requirements on each task model are that: (i) each task is sufficiently general to be applied across domains and across designs within a domain, and; (ii) the descriptive part of each task model is rich enough to match to service requests using the SeCSE service discovery algorithm.

The validation of the presented task-based service discovery platform through future empirical studies is still in its infancy anda research open question. In this regard we

plan to collaborate with all S-Cube partnersto collaboratively buildthe above-mentioned Task KB.

In the introduction we mentioned that not only new ways of thinking are needed toexplicitly relate users with service discovery but also with service composition techniques.In S-CUBE we conjecture that codified HCI knowledge can be used to inform service composition during the architecture design for a SBA. Most existing business process and work flow modelling techniques model coarse-grain processes with little support for finer-grain user tasks of different types and interactions with the service-based applications. User task models from HCI naturally plug this gap, and introduce new concepts such as task goals from the user perspective not modelled using approaches such as BPEL. We plan to explore the proposal through extension of another SeCSE development tool – the Composition Designer [21]. In future work we will extend the Composition Designer to allow a service integrator to generate a service composition with user task models in order to inform more effective service composition.

5. Acknowledgements

The research leading to these results hasreceived funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

6. References

1. M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-Oriented Computing Research Roadmap.
2. SOA failures traced to people, process issues. [Online]. Available at: http://www.networkworld.com/news/2008/043008-interop-soa.html [Accessed 18th July 2008]
3. Amazon mechanical Turk homepage, http://www.mturk.com/mturk/welcome
4. P. Palanque and S. Basnyat, “Task Patterns for Taking into Account in an Efficient and Systematic Way Both Standard and Abnormal User Behavior” in IFIP 13.5 Working Conference on Human Error, Safety and Systems Development, pp. 109-130, 2004
5. S-CubeCD-IA-2.2.2
6. EU 511680 Integrated Project (SeCSE), http://secse.eng.it.
7. SODIUM, Service-Oriented Development In a Unified fraMework, IST-FP6-004559, http://www.atc.gr/sodium.
8. J. Preece, Y. Rogers, D. Benyon, S. Holland and T. Carey, Human-Computer Interaction,.Addison-Wesley, 1994
9. F. Paterno, C. Santoro, “Preventing user errors by systematic analysis of deviations from the system task model” in International Journal of Human-Computer Studies, Vol. 56, Issue 2, pp. 225-245, 2002
10. G.D. Abowd, “Using formal methods for the specification of user interfaces” in Proceedings of the Second Irvine Software Symposium, pp. 109-130, 1992
11. G. Mori, F. Paterno and C. Santoro, ‘CTTE: support for developing and analyzing task models for interactive system design’. IEEE Transactions on software engineering (2002) vol. 28 (8) pp. 797-813
12. D. Sinnig, M. Wurdel, P. Forbrig, P. Chalin and F. Khendek, “Practical Extensions for Task Models” in lecture Notes in Computer Science, Vol. 4849/2007, pp. 42-55, 2007
13. D. Diaper and N. Stanton, The Handbook of Task Analysis for Human-computer Interaction, Lawrence Erlbaum Associates, 2003
14. Adobe Developer Connection, http://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html
15. B. Kirwan and L.K. Ainsworth (eds), A Guide to Task Analysis, Taylor and Francis, 1992
16. F. Paterno, C. Mancini, S. Meniconi, “ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models” in Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, pp. 362-369, 1997
17. Jones S.V., Maiden N.A.M., Zachos K. & Zhu X., 2005, ‘How Service-Centric Systems Change the Requirements Process’, Proceedings REFSQ’2005 Workshop, in conjunction with CaiSE’2005, 13-14 2005, Porto, Portugal.
18. K. Zachos, N.A.M Maiden, S.Jones and X.Zhu, “Discovering Web Services To Specify More Complete System Requirements’” Proc. 19th Conference on Advanced Information System Engineering (CAiSE'07), 2007. pp.142-157.
19. J. Walkerdine, J., Hutchinson, P. Sawyer, G. Dobson, V. Onditi, "A Faceted Approach to Service Specification", Proc. 2nd Int’l Conf. on Internet and Web Applications and Services (ICIW 07), Mauritius, 2007.
20. Zachos K., Zhu X., Maiden N.A.M & Jones S., 2006 (b), “Seamlessly Integrating Service Discovery into UML Requirements Processes”, The 2006 International Workshop on Service Oriented Software Engineering (IW-SOSE '06), ICSE, Shanghai, China, 2006.
21. M. Colombo, E. Di Nitto, M. Mauri, “SCENE: a service composition execution environment supporting dynamic changes disciplined through rules”, The 4th International Conference on Service Oriented Computing (ICSOC 2006) Chicago, USA, December 4-7, 2006

4 Dhungana, D.; Seyff, N.; Graf, F.: *Using Codified Context Knowledge to Facilitate End-user Requirements Elicitation*. – not yet submitted.

Using Codified Context Knowledge to Facilitate End-user Requirements Elicitation

Deepak Dhungana[1], Norbert Seyff[2], Florian Graf[2]

[1]Lero- The Irish Software Engineering Research Centre
University of Limerick, Limerick, Ireland
deepak.dhungana@lero.ie

[2]Centre for HCI Design, City University London, London, UK
n.seyff@soi.city.ac.uk; florian.graf1@gmail.com

Abstract. End-users communicate their needs and wishes in form of natural language. Such statements are often used to start activities such as requirements elicitation and negotiation which result in the specification of agreed key stakeholder requirements. Requirements analysts usually facilitate these cost and time consuming tasks. Traditional requirements engineering does not focus on identifying individual user needs. However, novel software engineering paradigms such as service-oriented computing require approaches which allow the cost-effective identification of individual end-user needs. Our research investigates on how end-users can be empowered to document their individual needs themselves, which allow the provision of tailored service-based application. We propose a solution which facilitates end-users requirements elicitation by providing contextual information codified in models of reuse-based approaches such as software product line engineering. We present a "smart" tool for end-users allowing them to specify their needs and to customize a service-oriented software system based on given contextual information.

Keywords: End-user requirements elicitation; software product lines; automated customization of applications.

1 Introduction and Motivation

Software applications are increasingly being part of our everyday lives. Novel software engineering paradigms such as service-oriented computing promote reusing of available functionality and allow the cost-effective composition of tailored software systems [1]. With such developments, it is inevitable to adapt traditional Requirements Engineering (RE) approaches and to strengthen end-user involvement in early software system design. We foresee that in the near future end-users will be directly involved in customizing and tailoring applications to immediately get a software system fulfilling their needs. Such a vision has to consider several constraints and has significant implications for requirements engineering research and practice.

For example, end-users are usually not familiar with requirements engineering or software development and do not understand technical details of software configuration. This means that technical details need to be abstracted away from end-users and that systems' adaptability has to be communicated in non-technical terms. Furthermore, requirements analysts are not imminent in the daily lives of end-users and therefore cannot support them in specifying formalized requirements descriptions that can be automatically processed. Research is needed to explore end-user driven requirements elicitation and software customization.

In this paper, we present a tool-supported approach combining end-user driven requirements elicitation and end-user driven customization of service-based applications. We propose to use (contextual) knowledge codified in software product line variability models to inform end-user driven requirements elicitation. This approach shows the potential to enable end-users to specify their needs using natural language text while sophisticated product line modeling and configuration generation tools are used to provide further input for requirements gathering. Particularly, end-users are asked questions about missing details regarding their needs. The advantage of this approach is the possibility to automatically evaluate given answers for further processing and customization of service-based applications. Thereby it will be possible to provide a service based system fulfilling individual end-users' needs within a short amount of time.

Several attempts have been made in the past to introduce feature modeling as a means to involve end-users in service customization. For example, the authors in [10] classify web services features from the users' point of view and propose to use feature diagrams for modeling flexibility of the Web Services. In [12], authors introduce feature modeling and configuring techniques in domain engineering into service-oriented computing, and correspondingly propose a business-level service model and an end-user friendly service customization mechanism. The use of feature modeling is a promising way of customizing applications, however not very convenient for the end-users as they are strictly forced to think in terms of available features. Furthermore, end-users are not used to selecting features from a complex feature tree; it does not reflect the natural way of their thinking procedures. Hartman et al. [8] have already introduced the concept of a "Context Variability model", which contains the primary drivers for variation, e.g. different geographic regions. However, the motivation behind this research was not to support the end-users during requirements elicitation. The context variability model constrains the feature model, which makes it possible to model multiple product lines supporting several dimensions in the context space.

The rest of this paper is structured as follows: In Section II, we describe our goal and the research objectives underlying this work. Section III discusses a conceptual solution which is structured by the research objectives and ends with a service provider's and end-user's perspective on an application scenario. Section IV presents a tool prototype enabling end-user driven requirements elicitation and service based system configuration. In Section V we discuss an application example and Section VI rounds out the paper by summarizing the approach, and giving an overview on ongoing and future work; thereby underlining our vision of involving end-users in tailoring and customizing service oriented applications covering their needs.

2 Research Goal and Objectives

The goal of our research is to *explore how codified context knowledge can support end-users in specifying individual needs and self-customizing service-oriented solutions*. We aim to build tools and techniques that enable end-users in gathering requirements and consequently customizing a service-based application. In particular, we focus on exploring how end-user needs and contextual information can be discovered and automatically processed. In this paper, we aim to provide first solutions to the following objectives, which have guided our research.

RO 1: Codify context knowledge in the service-oriented domain. The first research objective focuses on providing a solution which allows modeling and codifying contextual information in order to inform early system design. The codified context knowledge should include architectural aspects of service based solutions as well as information on when a service needs to be part or a service-based solution in order to fulfill an end-user's need.

RO 2: Identify relevant context knowledge based on individual user needs. Our research is based on the idea that end-users are able to express individual needs with natural language text descriptions. Research objective two focuses on mapping codified context knowledge and user needs in order to present relevant context information to the end-user and to enable the systematic acquisition of context information.

RO 3: Use codified context knowledge to stimulate requirements elicitation. The context-specific provision of context related information is supposed to stimulate end-user driven requirements elicitation. Research objective three focuses on exploring and identifying different ways of how contextual information can be beneficial for end-users and stimulate the discovery of individual needs.

RO 4: Use codified context knowledge to customize service-based applications. The fourth research objective focuses on providing a solution allowing end-users to self-customize a service-oriented system with the help of codified context knowledge. This research will particularly focus on identifying a solution which allows representing architectural knowledge in form which is understandable for end-user.

RO 5: Use individual end-user needs to maintain and evolve codified context knowledge. The fifth research objective highlights the need for a solution, which allows identifying novel contextual knowledge described in end-user needs. While new relevant information can be mined from the users' needs the codified knowledge needs to continuously be maintain and evolve to ensure the correctness and completeness of the codified information.

3 Conceptual Solution

Our approach combines well-accepted methods for requirements elicitation (i.e. natural language descriptions of end-user needs) and well-established methods for software customization (i.e. product line models). We explore to how to use codified contextual knowledge in form of product line models as input for requirements

gathering and as an enabler for end-user driven software customization. Therefore we first define our notion of contextual knowledge.

3.1 Contextual Knowledge

The interpretation of user's needs requires contextual information which, for example, can be related to the identity of things named in the text: people, places, books, etc; which can be related to geographical locations, dates, weather conditions, units of measurement etc. Using the term "context" we also refer to information that is required by system experts or developers to view the requirements in a broader perspective. Furthermore, contextual information may be the source of data required to ascertain whether target levels for attributes have been met, or whether there are exceptions for particular cases. Contextual information covers profiles of participating or related systems, environment and stakeholders, e.g., natural environment profile; network profile; user profile; terminal device profile etc. Contextual information may be collected by the service providers or may be extracted from documentation of the existing services. In our work we define contextual information as any kind of information that characterizes or provides additional information regarding any feature or condition of the delivery and consumption environment of services.

Codifying context information is related to documenting all possible situations and purpose of use, focusing on the different alternatives for implementation, deployment or usage. A huge body of work exists in software product line engineering, which could be used to model the contextual information and use it for product customization. We have exploited this idea in our approach to create variability models for contextual information relevant to the end-users during requirements elicitation.

3.2 Approach Overview

An overview of the conceptual solution is presented in Figure 1. It depicts the key activities and the flow of information among the different participants. A service provider usually knows about the features of a service and how it can be adapted to different contextual situations. (1) In our approach this knowledge is codified in the form of product line variability models. (2) End-users document initial text-based requirements descriptions which are analyzed and allow to present codified contextual knowledge presenting form of questionnaires. To enable this, we map the entered text to the information in models using natural language processing. (3) providing contextual knowledge to end-users is supposed to stimulate requirements elicitation and allows end-users to provide more detailed information about her context by answering the questions. (4) Answers and the underlying product line model allow the generation of a service-based prototype application.

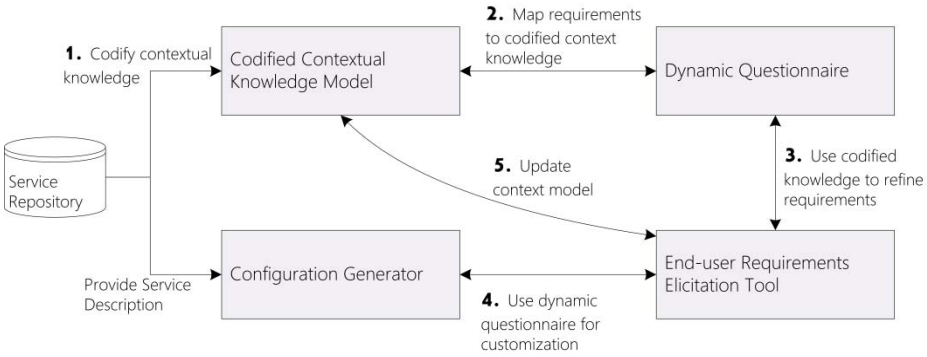


Figure 1 Overview of different activities associated with requirements elicitation using codified context knowledge models of service based applications.

In the following we discuss how our approach realizes the discussed research objectives:

RO 1. Codify context knowledge in the service-oriented domain

Product line models are primarily designed to document variability of reusable software artifacts. They contain the information on how reusable artifacts can be combined in order to generate a running system. On the other hand such models also document configuration options enabling the customization of software components such as services. Therefore, they contain a lot of codified knowledge about the envisioned work-context of future service-oriented systems. Some of this contextual information can be directly gathered from service repositories as they contain service descriptions documenting the purpose and the context of services. However the information on how reusable artifacts can be combined and configured can also provide information on the context of service centric systems.

Product line variability models are particularly useful, when it comes to documenting a set of available options, relationships between them and the implications of choosing an option. In that sense, contextual information is related to placing the users' needs and restrictions in the space of all available options. For example, when we need to elicit information about the weather conditions, under which the system is adopted, one can model that a variation point in a product line model with possible weather conditions as alternative to that variation point.

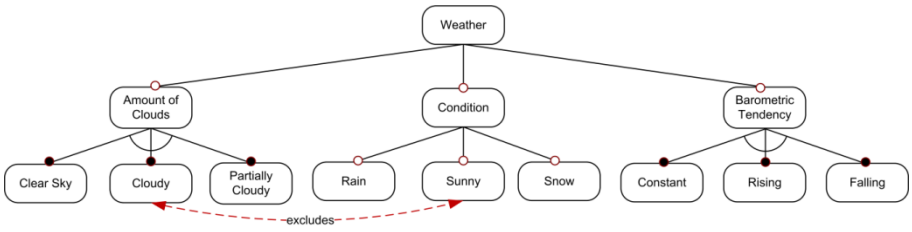


Figure 2 An example of a product line model depicting the variability of weather conditions identified to be relevant for the future system context.

RO 2. Identify relevant context knowledge based on individual user needs

To make codified context knowledge in variability models useful, it is important to identify relevant parts of such models in order to present the right information to the end-user at the right time. Therefore it is essential to present relevant information to the end-user while the end user continues writing her requirements. Given the users' input, we pick different parts of the model and present the information in the model to her to stimulate further requirements gathering, and encourage the end-user to refine already identified needs.

To achieve this goal, it is important to analyze, tokenize, and filter natural language text. One has to detect and extract metadata and structured text content from the users' needs, thereby attempting to extract keywords that are likely to be present in the codified context model. The mapping is established based on extracted keywords and their synonyms. By using advanced techniques like, proximity searches (finding related terms near a given term), one can significantly improve the mapping between the users' needs and codified contextual knowledge.

RO 3. Use codified context knowledge to stimulate requirements elicitation

Codified context knowledge is presented to end-users in form of questions and possible answers to questions. Although providing answers to these questions primarily focuses on product configuration and customization this information is also supposed to support end-users in requirements discovery. We foresee several benefits for end-user driven requirements elicitation by presenting context knowledge to end-users. As discussed in the previous section relevant context knowledge is selected based on initial end-user requirements. The presented contextual knowledge, in most cases, includes more detailed information than the actual requirement documented by the end-user. Reviewing this information might support the end-users in refining the initial requirement and help end-users to more precisely and more accurately describe a need.

Although the presented context knowledge is triggered by initial end-user needs, it might discuss functions which are is not covered by the end-user requirements description so far. Consequently the end-user might start to specify his individual needs related to these discovered functions. The gathering of new requirements based on presented contextual information might lead to more complete requirements. Presenting contextual information in general might support the user in getting a

clearer vision of the future system and guides end-users towards a solution which can be customized and generated based on existing service solutions. However, we highlight that the discussed approach does not limit end-users in describing individual needs. It is the end-user's choice when to stop documenting initial requirements and reviewing contextual information in order to simulate requirements election.

RO 4. Use codified context knowledge to customize service-based applications

Developing a customized system by composing reusable artifacts such as services is a non trivial task [1]. However, we intend to use codified context knowledge in order to enable end-users to build customized software (prototypes) without being facilitated by software engineers. As discussed in RO1 we use contextual knowledge codified in product line models. This contextual information enables us to ask the end-user about the future system context of an envisioned customized software system. Furthermore it can be used to map entered requirements to codified contextual knowledge in order to ask the end-user questions related to entered requirements. A list of questions regarding all possible configurations of services in a repository might contain several hundreds of entries. Therefore we intend to reduce the number of questions presented to the end-user to relevant ones. For example, if the end-user enters the requirement "I would like software support regarding a travel to London!", the codified context and the word "travel" can be mapped to present the question: "Which kind of transport would you like to use?". Such questions are presented dynamically to the end-user depending on entered natural language requirements descriptions. By answering the questions (e.g. by selecting the bus option) the end-user selects a customized solution without having to consider technical details or knowing about underlying services. For example the selection of the "bus" option could select the "Bus Ticket Booking" service. If the product line model which serves as basis for these configuration options is correct the end-user will always be able to select a valid configuration by answering the questions.

RO 5. Use end-user needs to maintain and evolve codified context knowledge

The discussed approach enables end-users to customize a solution to her needs by reusing existing services. However a solution composed of standard services might not cover all individual end-user needs. Therefore we propose to give service providers access to end-user documented end-user needs. By analyzing gathered end-user requirements and semi automatically mapping them to documented contextual knowledge and variability models, service providers can be supported in discovering requirements which could not be satisfied with standard services. Identified gaps between end-user requirements and provided solutions can trigger the development of new service based solutions or an update of the variability model and the codified contextual knowledge. An update of the variability model might be necessary, if an end-user's requirement could not be mapped to codified contextual information for reasons of incompleteness or incorrectness of the context knowledge model. Therefore gathered end-user requirements can provide essential input for further service development and to maintain and enhance the contextual knowledge codified in variability models. As our approach relies on valid variability models and correct

and complete contextual knowledge such a feedback mechanism is essential to ensure end-user support for service customization.

3.3 Service Provider's Perspective

In this section, we describe our approach from the perspective of a service provider, describing the steps required to create the infrastructure for supporting enabling end-user requirements elicitation and product customization.

Create and publish variability models. Our approach relies on codified context knowledge; so the first step is obviously to codify context knowledge. To do this, service providers can analyze available services and conditions of their adoption. Based on these conditions, service providers create decision models reflecting the variability of available services, as discussed in [3, 4].

Categorize variability model on the basis of domains. To enable support for end-users based on a large number of variability models these can be categorized based on domains (e.g. travel). The categorization can support end-users to narrow down the number of relevant services in an efficient way and therefore facilitates providing fast feedback to end-users via exploring relevant services for information relevant for end-users.

Present configuration options of services to end-users. Based on the natural text descriptions provided by end-users, it might be possible to identify relevant services and to initially configure them. However the end-user will not be aware about possible configuration options when she documents her needs. Therefore we intend to provide natural language questions to the end-user about his whishes regarding a configuration option.

Generate and present prototypic solution to the end-user. With the information that is available in a product line model of the available services, the service provider can adopt domain-specific configuration generators to create a prototypic solution to the users' needs. This is done by evaluating the answers given by the users to the presented questions related to their needs.

Discuss and negotiate requirements and system. Although the outcome of the requirements elicitation process with our approach is a working service based system, this system might not fulfill all requirements of the end-user. As our approach relies on reuse there might be end-user's requirements which cannot be covered by composing reusable services. However, applying our approach, results in a first functional prototype which can be used in further requirements negotiations and discussions between end-users and system providers. These discussions can lead to the development of new services and/or changes to the variability model.

3.4 End-users' Perspective

The following paragraph describes the approach from the perspective of an end-user.

End-user selects domain of interest. To narrow down the relevant variability models in an effective way the first task of the end-user is to select a domain of interest. These domains are intended to be very general and inclusive as end-users are familiar with selecting categories like 'travelling'. However, this information is highly important to identify relevant services.

End-users document their needs using natural language. End-users are familiar with expressing their needs, ideas and requirements in natural language text expressions. Furthermore end-users are familiar in documenting text with a text editor such as Microsoft Word. Therefore we enable end-users to blog their needs and ideas regarding a service oriented system via a simple to use text editor.

End-users answer questions presented to them. The text that is typed in by the end users is analyzed and selected questions from the variability model are presented to the end user. The end-user can then decide to answer these questions or to continue with typing. The list of questions changes dynamically based on the user's textual input and answers to previous questions.

Configure and deploy system (prototype). The end-users' natural language requirements descriptions and their answers to the configuration questions provide the necessary information to select one of many possible valid configurations of a system composed of reusable artifacts such as services. At the end-of the requirements elicitation process the chosen configuration can be processed automatically to compose a service based system and directly deploy it to the end-user's device.

4 Tool Support

We have implemented a tool prototype called EuReCuS, which enables end-users to write down requirements using natural language text and presents relevant questions, as the user writes down her needs. The tool is currently available as an Eclipse plug-in, utilizing the product line variability modeling capabilities of the DOPLER [5, 6] tool suite.

4.1 Tool Architecture

An overview of the EuReCuS tool architecture is presented in **Fehler! Verweisquelle konnte nicht gefunden werden.**We now discuss the different aspects of the tool as numbered in the Figure.

1. *Service Variability Model:* This is a formal documentation of the different contextual options available in service based software solutions. This model is created with the DOPLER tool suite. It consists of decisions, the users can take and rules that need to be considered when selecting services based on the users answers to the relevant questions.
2. *Variability Model Execution Engine:* DOPLER Variability models are executed using rule engines that are capable of mapping the user's decisions to available services and propagating the effects of such decisions in the configuration of the future system.
3. *End-user Tool:* This tool has been designed for end-users and is capable of gathering end-users' needs using a seemingly simple text editor. The text editor is however sensitive to what the user is typing. It is linked to the variability model execution engine to identify relevant questions and pass the end-users' answers on these questions.
4. *Natural Language Processor:* We are currently adopting Apache Lucene, as a natural language analyzer and tokenizer. It is a high-performance, full-featured

text search engine library written in Java. Lucene provides advanced features like stemming and synonym-based search.

5. *Questionnaire tool:* This part of the end-user tool, is linked with the text editor for requirements input. The questionnaire tool displays the relevant questions to the end-user and provides interactive UI elements to answer them. The answers are then passed to the variability modeling execution engine.
6. *Domain-specific service composer:* This tool makes use of the knowledge in the variability model to generate a running configuration of the selected services. The information about which services are selected is passed through by the model execution engine.
7. *Customized Application:* This is the actual prototype of the system the user envisions. The domain-specific service composer creates and deploys this application based on the end-users' answers. The end-users have a chance to review the application on the fly and decide, whether is what they wanted to have.

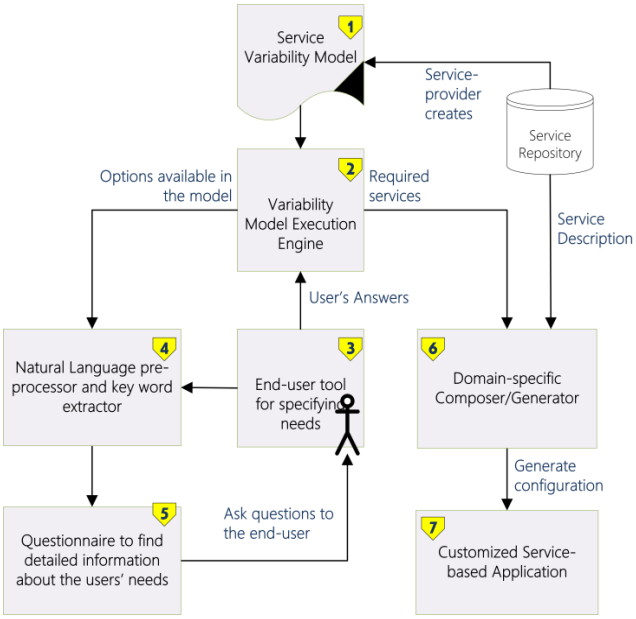


Figure 3 Architecture of the EuReCuS tool suite, depicting the position of the end-user and related tool-components.

5 Application Example

To highlight the application of our approach we prepared an example discussing how an end-user would use the developed tools in order to document requirements and customize a software solution.

We decided to use an example which discusses everyday needs of an end-user named Tom. His requirements describe how a future software system should support his daily commuting. With the help of our tool prototype Tom starts to document individual needs in the form of natural language text (see Figure 4).

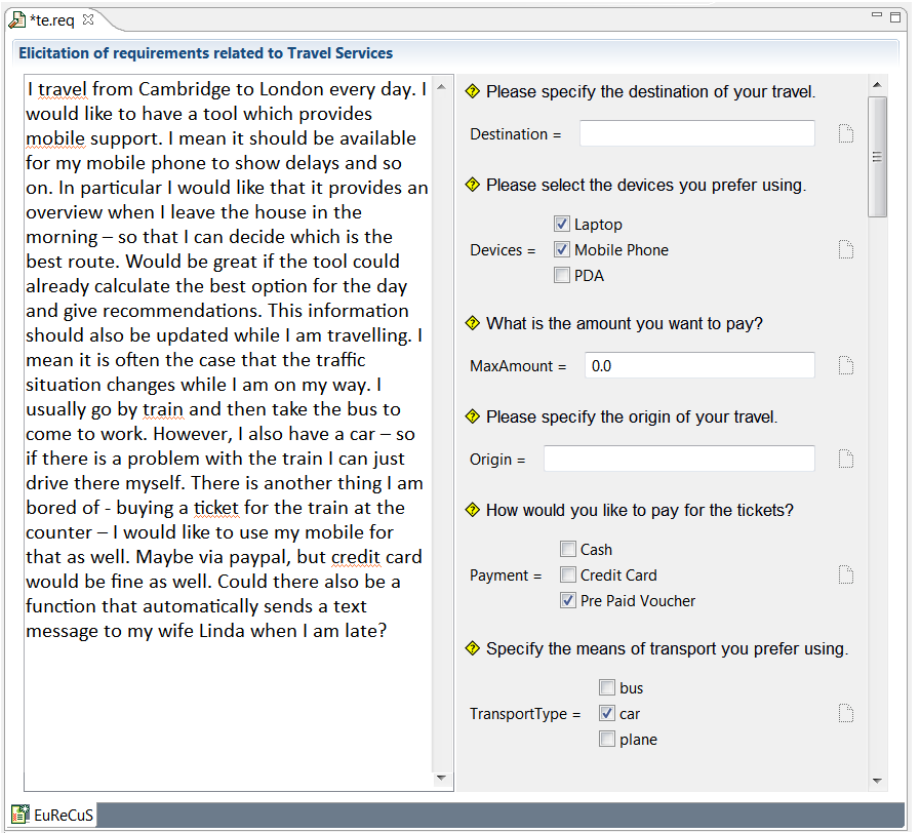


Figure 4 Screenshot of the EuReCuS tool, depicting the text editor on the left and the set of relevant contextual questions on the right.

Using the text-editor functionality of our tool Tom is able to describe individual needs using natural language text. However, Tom, not being an RE expert and unfamiliar with requirements documentation, will most likely not document fully specified requirements descriptions. We expect Tom to provide a mixture of needs, rationale descriptions, and uncertainties documented in a kind of user story. In general Tom's description is supposed to include a lot of contextual information. Tom, for example, could describe needs using statements such as: I would like to have a tool which provides mobile support and this (travel) information should also be updated while I am traveling. Using our tool Tom is not forced to describe his needs using a certain notation. Furthermore, the approach is not limiting Tom's creativity as he is allowed to document whatever comes into his mind.

While Tom is brainstorming his vision of the future system our tool uses this information to identify codified context knowledge which could support Tom in

refining and identifying needs. The codified context knowledge is presented in the form of domain specific questions together with possible answers. For example, analyzing Tom's description and using keyword matching our tool comes up with more detailed questions. This could include questions on the type of mobile device Tom is envisioning to use (e.g. Please specify the devices you prefer using). The system will provide possible answers, such as Laptop, Mobile Phone, and PDA which allows Tom to think about alternative options. Although he did not mention support for his Laptop in his initial description, he might discover that he wants to use the envisioned system on his laptop as well. Furthermore, the question about how much money Tom is willing to pay might support him to specify in more detail what kind of metrics the envisioned system is supposed to use to identity the best route. For example, based on this contextual trigger he could start to describe in more detail that the future system should automatically identify the cheapest way to get to a destination. However, regarding this example, we would assume that Tom requests that the system should identify the fastest option instead of the cheapest.

Stimulating Tom's brainstorming by providing recognition cues in the form of questions in only one important aspect of our solution. The questions represent codified context knowledge, which includes information about system variability. Tom is therefore able to customize existing software solutions with the help of codified context knowledge. This does not mean that Tom needs a technical background and has to understand the system architecture. Our solution is based on the idea that Tom is able to customize existing software solutions based on answering the provided questions. Depending on his individual needs Tom is able to select the answers reflecting what he expects from a future system. After answering all necessary questions this information is used to automatically generate a first prototype which is tailored to Tom's needs.

6 Conclusions and Future Work

Reuse-based approaches such as service-oriented computing provide efficient ways to compose software systems fulfilling end-users' individual needs. In the context of reuse-based systems, there is a change in how RE should be perceived. Instead of being a front-end activity in the software engineering process and focusing on defining requirements for the development of software systems, the focus shifts towards mapping the users' needs to already existing reusable artifacts. This implies that knowledge about already available functionality is vital and can guide requirements elicitation and system analysis. Furthermore, weaving this knowledge into requirements might stimulate end-user creativity and trigger new requirements.

Requirements engineering for service centric systems has to consider individual customer requirements and on the other hand to enable the reuse existing services. However current methods and techniques rely on requirements analysts to identify existing services covering end-user needs and to document services configuration; end-users are not likely to understand technical service interface descriptions and ways to configure existing services. Therefore, we propose to use codified context knowledge to support the end-users during requirements elicitation.

We consider product line variability models to be suitable for modeling and presenting contextual information to the end-users. The primary focus and contribution

of this paper lies on increasing the creativity and stimulating new requirements during requirements elicitation. However, applying this approach has several other benefits. In ideal cases, the end-users can construct tailored applications themselves (by utilizing domain-specific product generators and reusable artifacts documented in the product line model). In the normal run, the requirements elicited using our approach constitute of textual description of users' needs and a prototypic configuration of available services based on the users' interaction with the product line model.

Rigorous modeling methods, languages, and tools are needed to describe and manage the context of service-oriented applications and to implement effective means for configuring and tailoring them. More importantly, the involvement of end-users to make use of these models and tools is a key to any significant breakthrough in this area.

In the future, we aim to continue research in this area by carrying out user-studies to measure and validate the effectiveness of the requirements elicitation approach. Based on the feedback from users, we will work towards applying our approach in real-service repositories, empowering users to customize service-oriented applications themselves. This will provide us with further feedback for further improvements of the approach and the tools.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State-of-the-art and Research Challenges. In IEEE Computer, vol. 40 (11), November 2007.
2. Clements, P. and Northrop, L., Software Product Lines: Practices and Patterns: SEI Series in Software Engineering, Addison-Wesley, 2005.
3. Clotet, R., Dhungana, D., Franch, X., Grünbacher, P., Lopez, L., Marco, J., and Seyff, N., Dealing with changes in service-oriented computing through integrated goal and variability modeling. In Workshop on Variability Modelling of Software-intensive Systems (VAMOS 2008), pages 43–52, Essen, Germany, 2008. ICB-Research Report No. 22.
4. Dhungana, D., Heymans, P., Rabiser, R., A Formal Semantics for Decision-oriented Variability Modeling with DOPLER, In Workshop on Variability Modelling of Software-intensive Systems (VAMOS 2010), pages 37-45, Linz, Austria, 2010. ICB-Research Report. 37.
5. Dhungana, D., Rabiser, R., Grünbacher, P., Lehner, K., Federspiel, C.: DOPLER: An Adaptable Tool Suite for Product Line Engineering. 11th International Software Product Line Conference (SPLC 2007), Kyoto, Japan, September 10-14, 2007.
6. Dhungana, D., Grünbacher, P., and Rabiser, R., Domain-specific adaptations of product line variability modeling. In IFIP WG 8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences, Geneva, Switzerland, 2007.
7. Froschauer, R., Zoitl, A., and Grünbacher, P., Development and adaptation of iec 61499 automation and control applications with runtime variability models. In 7th IEEE Int'l Conference on Industrial Informatics, INDIN 2009,, Cardiff, UK, 2009.
8. Hartmann, H., Trew, T., "Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains," 12th International Software Product Line Conference, pp. 12-21, Limerick, Ireland
9. Pohl ,K., Böckle, G., and v. d. Linden, F. J., Software Product Line Engineering: Foundations, Principles, and Techniques: Springer, 2005.

10. Robak, S., Franczyk, B., Modeling Web Services Variability with Feature Diagrams, in Web, Web-Services, and Database Systems, Volume 2593/2009.
11. Stevens, R., Brook, P., Jackson, K. & Arnold, S. (1998). Systems Engineering: Coping with Complexity. Prentice Hall Europe.
12. Wang, J., Yu, J., A Business-Level Service Model Supporting End-User Customization, Service-Oriented Computing - ICSOC 2007 Workshops, pp 295-303, Vienna, Austria.
13. Wolfinger, R. Reiter, S., Dhungana, D., Grünbacher, P., and Prähofer, H. Supporting runtime system adaptation through product line engineering and plug-in techniques. In 7th IEEE International Conference on Composition-Based Software Systems (ICCBSS), Madrid, Spain, 2008. IEEE Computer Society.

5 Lucchese, C.; Orlando, S.; Perego, R.; Silvestri, F.; Tolomei, G.: *Detecting Task-based Query Sessions using Wiktionary*. – not yet submitted.

Detecting Task-based Query Sessions using Wiktionary

Claudio Lucchese
ISTI - CNR
56124 Pisa, ITALY
c.lucchese@isti.cnr.it

Salvatore Orlando
Ca' Foscari University
30172 Venice, ITALY
orlando@dsi.unive.it

Raffaele Perego
ISTI - CNR
56124 Pisa, ITALY
r.perego@isti.cnr.it

Fabrizio Silvestri
ISTI - CNR
56124 Pisa, ITALY
f.silvestri@isti.cnr.it

Gabriele Tolomei
ISTI - CNR
56124 Pisa, ITALY
g.tolomei@isti.cnr.it

ABSTRACT

Nowadays, people have been increasingly interested in exploiting the *World Wide Web* (Web) not only for having access to simple Web pages, but mainly for accomplishing even complex tasks in a simpler way. Our research challenge is to provide a mechanism to split into user sessions a very large, long-term log of queries submitted to a Web Search Engine (WSE). Our hypothesis is that query sessions entail the concept of user *task*. Hence, we present a novel query clustering technique aimed to identify these hidden tasks. We devise an extended *K-means*-like algorithm that uses a novel distance metrics, and combines query content features, inter-query temporal interval, and a new feature based on the collaborative knowledge base collected by *Wiktionary*. Basically, we exploit the Wiktionary data source for increasing the meaningfulness of each query, whose average number of terms is otherwise low.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query clustering, Collaborative knowledge, Search process*

General Terms

Algorithms, Design, Experimentation

Keywords

Query log segmentation, Session breaking, Query clustering Web-mediated Task

1. INTRODUCTION

According to Tim Berners-Lee, the World Wide Web has becoming to move from a simple "read-only" environment (i.e., Web 1.0) to a collaborative "read-write" platform in which end users may act both as content provider and consumer from time to time (i.e., Web 2.0). More recently, people have started to exploit the Web not only for having access to its huge document collection but also for accomplishing their everyday activities in a simpler way. This makes the Web to go one step over toward a "read-write-execute" platform, which is the so-called Web 3.0. This vision is also supported by authoritative people in the Web search domain. During the DEMOFall08 Conference, there was a discussion panel on "Where the Web is Going"[1], in which Peter Norvig from *Google* and Prabhakar Raghavan from *Yahoo!* basically agreed that, rather than supporting only one search at a time, Web Search Engines (WSEs) will soon focus on helping people get a bigger task done. A significative quote extracted from Raghavan's speech is the following: "*People intrinsically don't want to search. People don't come to work every day saying "I need to search"...* ***They want to run their lives!***".

So far, several works investigated how real search intent can be devised by looking at the queries users issued to WSEs [3, 34, 16, 13]. Following this research direction, we plan to go one step forward aiming to understand users' behaviors on a *task-based* perspective.

From our preliminary study of historical data stored on a very large and long-term WSE query log, we pointed out a significative set of sample activities that many users tried to achieve by issuing some independent query chains. Those Web-mediated processes occurred quite frequently in our WSE query log as the result of stream of queries issued by different users. To some extent, this enforce the vision provided by Ricardo Baeza-Yates from Yahoo!, who claimed that in the next so-called Web 3.0 "***People want to get tasks done***" [2].

So, we can claim an interesting shift here from a model where each search is independent, to one where Web search support systems may expect users to do multiple searches when trying to accomplish their activities. This novel way of searching the Web "by tasks to be executed", instead of "by documents to be retrieved", has to be enabled by new mechanisms that we aim to investigate.

[1] http://blogoscoped.com/archive/2008-11-06-n63.html
[2] http://www.cwr.cl/la-web2008/slides/Wisdom-of-Crowds-long.pdf

Typically, Web users are not easily identifiable by WSEs because IP addresses and cookies may be shared across multiple users. Recently WSEs have added login capabilities, so that users can perform their search activities while they are logged into the system. However, even in that case, detecting precisely a user search session is not trivial because users could stay logged in for a long time trying to perform different tasks that are possibly unrelated.

Thus, our aim is to detect *task-based* query sessions, that is user query sessions whose final goal is to perform a Web-mediated task, from the whole stream of queries issued by a user to a WSE and collected into a query log. For this purpose, we propose an *unsupervised learning* approach that leverages on a novel query clustering algorithm whose aim is to group together queries related to the same task.

We devise a *K-means*-like query clustering algorithm, which relies on a novel *distance metrics* that exploits also the collaborative knowledge provided by the *Wiktionary*[3] data source for enriching the meaning of each issued query. The obtained results show that our approach produces better quality clusters with respect to traditional solutions that take into account only similarity between query terms and/or clicked-documents.

The rest of this document is organized as follows: Section 2 describes related works about session identification and real search intent discovery from the information available on WSE query logs. Furthermore, Section 3 describes our concept of user query session and formalizes the problem of detecting this kind of session from long-term query logs. Section 4 outlines our proposed solution for detecting task-based sessions describing the query clustering algorithm and its novel distance metrics we have devised. Section 5 shows the results obtained from the experiments done so far. Finally, Section 6 provides a summary of our work and points out possible future research directions and challenges.

2. RELATED WORK

Analysis of query logs collected by most Web Search Engines (WSEs) has increasingly gained interest across Web mining research community. Roughly, query logs record information about the *search activities* of users and so they are a suitable source of information for understanding how people search the Web.

Silverstein et al. [28] present a broad analysis of a very large query log data set collected by the *AltaVista* search engine. In particular, their analysis involves statistics about the occurrences of single query term and the co-occurrences of query term pairs. Moreover, they define a concept of "session" as a 5-minutes fixed time window on user search activities. According to this definition, they found that the average number of queries per session are 2.02.

Jansen and Spink [13] make a comparison of nine WSE transaction logs from the perspectives of session length, query length, query complexity, and content viewed. Here, they define a "session" to be the period of time occurring from the first recorded time-stamp to the last recorded time-stamp on the WSE server from a particular user in a particular day.

Richardson [25] shows the value of long-term WSE query logs with respect to short-term (within-session) query information. He claims that long-term query logs can be used to better understand the world where we live, showing that query effects are long-lasting. Basically, in his work Richardson does not look at term co-occurrences just within a search session that he agreed to be a 30 minutes time-window, but rather across entire query histories.

Silvestri et al. [29] show a number of applications, i.e., caching, index partitioning, and document prioritization that can benefit from analysis performed on WSE query logs, and that have strong high performance requirements.

Most works concerning with mining of query logs aim at understanding the real intent behind queries issued by users.

Broder [3] claims that the "need behind the query" in Web context is not clearly *informational* like in classical Information Retrieval domain. Hence, he proposes a *taxonomy* of Web searches classifying the queries according to their intent into 3 classes: (i) *navigational*, whose intent is to reach a specific Web site, (ii) *informational*, which aims to acquire some information from one or more Web documents, and finally (iii) *transactional*, whose intent is to perform some Web-mediated tasks. Moreover, Rose and Levinson [34] propose their own user search goals classification by extending the taxonomy devised by Broder in [3] to more hierarchical levels. Lee et al. [16] describe *whether* and *how* search goal identification process behind a user query might be automatically performed on the basis of two features, that is *past user-click behavior* and *anchor-link distribution*.

Ozmutlu and Çavdur [21] describe a mechanism for identifying topic changes in user search behavior and tested its validity. The parameters for the topic identification task are determined by a genetic algorithm using topic shift and continuation probabilities of the dataset leveraging on query patterns and time intervals.

Besides, many other works deal with the identification of users' search sessions boundaries.

He and Göker [10] propose different timeouts to segment user sessions, and later extended their work [11] to consider other features such as the overlap between terms in two consecutive queries.

Radlinski and Joachims [23] observe that users often perform a sequence of queries with a similar information need, and they refer to this sequence of reformulated queries as *query chains*. Their paper presents a simple method for automatically detecting query chains in query and clickthrough logs, and showed how to learn better retrieval functions using evidence of query chains.

Spink et al [30] investigate *multitasking* behaviors while users interacting with a WSE. Multitasking during Web searches involves the "seek-and-switch" process among several topics within a single user session. Here, a user session is defined to be the entire series of queries submitted by a user during one interaction with the WSE, so that session length might vary from less than a minute to a few hours. The results of this analysis performed on an AltaVista query log show that multitasking is a growing element in Web searching.

Boldi et al. [2] introduce the *query-flow graph* as a model for representing data collected in WSE query logs. They exploit information provided by this model for segmenting the query stream into sets of related information-seeking queries, i.e., *logical sessions* leveraging on an instance of the Asymmetric Traveler Salesman Problem (ATSP).

Finally, Jones and Klinkner [15] show that timeouts, whatever their length, are of limited utility in identifying task session boundaries and propose and evaluate a *supervised*

[3] http://www.wiktionary.org

learning method for the automated segmentation of users' query streams into hierarchical units.

Suitable mechanisms for identifying session boundaries, thus for extracting search goals, are clustering algorithms and techniques applied to the whole query session of each user.

Query clustering is a quite recent research issue and it is based on the assumption that if two queries belong to the same cluster, then they are also likely to be topically-related. Beeferman and Berger [1] introduce a technique for mining a collection of user transactions with a WSE for discovering clusters of similar queries and similar URLs. The algorithm they propose makes no use of the actual content of the queries or the URLs, but only how they co-occur within the click-through data stored in the query log modeled as a bipartite graph.

Wen et al. [33] describe a query clustering method that makes use of user logs for identifying the Web documents users have selected for a certain query. The similarity between two queries may be deduced from the common documents the users selected for them. Fu et al. [8] introduces a hybrid method to cluster queries by utilizing both the query terms and the results returned to queries, showing that this combination performs better than using either method alone.

Wai-Ting Leung et al. [17] develop online techniques that extract concepts from the Web-snippets of the search results returned from a query, and use these concepts to identify related queries. Moreover, authors propose a new two-phase personalized agglomerative clustering algorithm that is able to generate personalized query clusters.

Finally, Yi and Maghoul [35] propose an algorithm for extracting all maximal bipartite cliques (i.e., bicliques) from a click-through graph and compute an equivalence set of queries (i.e., a query cluster) from the maximal bicliques. A cluster of queries is formed from the queries in a biclique.

3. TASK-BASED SESSION

In digital libraries environments, users' sessions are easily detectable because the interaction pattern between users and system is well defined. Here, in fact, a user logs into the system, performs one or more queries for achieving a specific task, and finally she logs out. Unfortunately, that pattern is not applicable when user interacts with a Web Search Engine (WSE) for having access to the Web, because anonymous users could be logged in for a long time, by submitting independent queries and trying to perform different tasks that are possibly unrelated.

Therefore, an important challenge dealing with the information collected in WSE query logs is to detect meaningful *user query sessions*. A few work has proposed some definitions of user query sessions and different heuristics to split query logs. However, timeout-based methods remain the most common ones for detecting session boundaries. Silverstein et al. [28] define a session as the stream of consecutive queries issued by the same user within a 5-minutes time window, while Jansen and Spink [13] simply identify a session with the period that occurs between the first and the last time-stamp recorded for a certain user to the WSE server. Other approaches have been proposed in between, devising timeout threshold values that range from 30 to 120 minutes [20, 11].

According to Jones et al. [15], we believe that timeout-based approaches are not fully suitable for detecting *real* user search sessions. On the basis of the shift in Web users behaviors that we previously pointed out, we claim that a real user session should entail the concept of *task*. Thus, we define a *task-based session* to be a sequence of topically-related queries aiming to reach the same search goal, i.e., accomplishing the same Web-mediated task. Queries within the same task-based session might not be necessarily consecutive, hence splitting a query log using only a timeout-based mechanism could identify multiple and interleaved task-based sessions.

By referring to the terminology proposed by Jones et al. [15], we can further subdivide user search activities in a hierarchical way, and distinguish between *search goals* and *search missions*. A search goal is an atomic information need of a certain user, represented by one or more queries, for achieving a single task, i.e., a Web-mediated task. On the other hand, a search mission is a set of topically-related information needs, aiming to perform a bigger task represented by the composition of one or more eventually interleaved search goals, i.e., Web-mediated process.

3.1 Problem Modeling and Statement

Query logs of a WSE contains a tremendous amount of information regarding the users' activities over time. This is stored in terms of queries submitted, user IDs, time-stamps, clicked results, etc.

We denote with $\mathcal{QL}$ the log of the queries submitted to the WSE by a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ during a given observation period. Moreover, let $q_i \in \mathcal{QL}$ be a generic query issued by user u_i, and $q_{i,j} \in \mathcal{QL}$ be the j-th query issued by user u_i.

The methods that extract user sessions from $\mathcal{QL}$ has to analyze all the queries issued by each user u_i, i.e. the so-called *long-term sessions*.

Definition 3.1 (Long-Term Session $\mathcal{S}_i$). *$\mathcal{S}_i$ is the sequence of* all *the queries $q_i \in \mathcal{QL}$ issued by user $u_i \in \mathcal{U}$, chronologically ordered during the period of observation recorded in the query log:*

$$\mathcal{S}_i = \langle q_{i,1}, q_{i,2}, \ldots, q_{i,K} \rangle$$

Therefore,

$$\mathcal{QL} = \bigcup_{i=1}^{N} \mathcal{S}_i$$

Any partitioning of $\mathcal{S}_i$ could be a valid sessioning of the activity of user u_i.

A simple method is timeout cutoff. Given the query $q_{i,j}$, all the successive queries $q_{i,j+l}$ submitted within a given time-threshold, i.e. 30 minutes, are added to the same partition of $q_{i,j}$. Such a simple approach fails in detecting *task-based sessions*, as shown in [15]. First, it is not possible to predict in advance what is the duration of a session. More importantly, user interleaves many different information needs, and the related queries, during her long-term session. For example, at a given time she starts looking for something related to a given topic/task/information need, and then repeats a related search some time later.

Definition 3.2 (Task-based Sessions Θ_i). *Let $\theta_i^j \subseteq \mathcal{S}_i$ be a generic task-based session, i.e., a set of (not necessarily consecutive) queries issued by user u_i for reaching*

the same search goal and performing a given Web-mediated task.

Then, let $\Theta_i = \{\theta_i^1, \theta_i^2, \ldots, \theta_i^k\}$ *be the set of all the task-based sessions that are present in* $\mathcal{S}_i$*, where* $\theta_i^j \cap \theta_i^h = \varnothing$ *for any* j, h*.*

Thus, our *Task-based Session Detection Problem* (TSDP) can be formulated as the problem of detecting the *query grouping strategy* that better approximates the true user tasks based Θ_i, for each user $u_i \in \mathcal{U}$. Let $\mathcal{C}_i = \{c_i^1, c_i^2, \ldots, c_i^h\}$ be the task-based sessions identified by a given grouping strategy σ applied over $\mathcal{S}_i$, i.e. $\sigma(\mathcal{S}_i) = \mathcal{C}_i = \{c_i^1, c_i^2, \ldots, c_i^h\}$. The TSDP can thus be defined as follows.

Definition 3.3 (TSDP). *Given a query log* $\mathcal{QL}$*, the TSDP requires to find the best partitioning* $\sigma(\mathcal{S}_i)$ *of every long-term session* $\mathcal{S}_i$*, that better approximates the true task-based sessions* $\Theta_i = \{\theta_i^1, \theta_i^2, \ldots, \theta_i^k\}$*:*

$$\underset{\sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \delta(\Theta_i, \sigma(\mathcal{S}_i))$$

where δ *is a given distance function that measures the error in the partitioning* $\sigma(\mathcal{S}_i) = \mathcal{C}_i$ *with respect to* Θ_i*.*

Several concepts of "distance" can be used to measure the accuracy of a task-based sessioning, and consequently, several δ functions can be devised. Indeed, δ can be defined in terms of metrics like *coverage*, *precision*, *recall*, *purity*, etc. Different applications that exploit sessioning, e.g. query suggestion, results clustering, may prefer a different metric in order to maximize their achievements.

4. TASK-BASED SESSION DETECTION THROUGH QUERY CLUSTERING

A problem similar to the Task-based Session Detection Problem (TSDP) has been already faced by Jones and Klinkner using a *supervised learning* approach [15]. Here, the authors try to learn how to automatically detect search *goals* and *missions* by training some classifiers from a sampling set of manually labeled data. Conversely, we decide to address the TSDP using an *unsupervised learning* approach that does not require any manual training phase.

In particular, we claim that the concept of *task* is just contained into the stream of queries issued by users of Web Search Engines (WSEs). Thus, clustering queries according to specific features could be a suitable mechanism for identifying search goals (i.e., clusters of task-related queries) from each user long-term session, hence detecting task-based sessions.

As for any other clustering problem, research has to be focused mainly on two aspects: (i) the clustering algorithm to be run on data, and (ii) the *distance metrics* used for computing the distance function between queries.

Regarding the first issue (i), there have been mainly two classes of clustering algorithms: *hierarchical* and *partitional*. Hierarchical agglomerative clustering (*HAC*) and *K-means* algorithms are representatives of the two classes respectively [12].

In our work, we propose a *K-means*-like algorithm where the number K of final clusters to be produced is not given as the input. A parameter t, which corresponds to the maximum radius of a centroid-based cluster is otherwise provided.

The termination condition of the algorithm is assured because at the end of the main loop over the queries contained in a long-term session, each query is either added to an existing cluster or it becomes the centroid of a brand new cluster.

The description of our TSDP query clustering algorithm is given in Algorithm 1.

Algorithm 1 The TSDP query clustering algorithm.

Input:
- A long-term user session $\mathcal{S}_i$.
- The threshold t.

Output:
- $\mathcal{C}_i = \{c_i^1, c_i^2, \ldots, c_i^h\}$ set of clusters representing a partitioning $\sigma(\mathcal{S}_i)$.

$\mathcal{C}_i = \varnothing$
$current\_centroid = select\_centroid(\mathcal{S}_i)$
$c_i^{start} = create\_new\_cluster(current\_centroid)$
$\mathcal{C}_i = \mathcal{C}_i \cup c_i^{start}$
$\mathcal{S}_i = \mathcal{S}_i \setminus current\_centroid$
for all $q \in \mathcal{S}_i$ **do**
  $min\_dist = \infty \quad c_i^{min} = \text{NIL}$
  for all $c_i^j \in \mathcal{C}_i$ **do**
    $dist[j] = compute\_distance\left(q, centroid(c_i^j)\right)$
    if $dist[j] < min\_dist$ **then**
      $min\_dist = dist[j]$
      $c_i^{min} = c_i^j$
    end if
  end for
  if $min\_dist \leq t$ **then**
    $c_i^{min} = c_i^{min} \cup q$
    $recompute\_centroid(c_{min})$
  else
    $c_i^{new} = create\_new\_cluster(q)$
    $\mathcal{C}_i = \mathcal{C}_i \cup c_i^{new}$
  end if
end for
return C_i

4.1 Distance Metrics

Several approaches for devising suitable distance metrics to be used in query clustering contexts have been proposed. Most of these approaches derive from traditional document clustering in Information Retrieval domain and are based on similarity between query content [27]. However, the precision of those approach results to be quite low due to the short length of queries and the lack of the contextual information in which queries are issued [33]. According to Silverstein et al. [28], more than 85% of queries contain less than 3 terms and the average number of query terms ranges between 2 and 3 words (exactly 2.35). Thus, query terms can neither convey much information nor help to detect the semantics behind them since the same term might represent different semantic meanings, while on the other hand, different terms might refer to the same semantic meaning [24]. As an example, let us consider a query composed only by the term "donkey", which can refer to an animal, to a character from the Shrek movies, to a steam engine, and even to a bad poker player. Moreover, the query "donkey" is very dissimilar from the query "fiona" from a content-based point of view, while they can both refer to the same semantic concept (i.e., the

Shrek movies).

In order to overcome these drawbacks, other proposed approaches exploit cross-reference between queries and user activities (i.e., relevance feedback) [26]. By cross-reference, we mean any relationship created between a query and a Web document. The intuition of using cross-references is that similarity between Web documents can be transferred to queries through these references, and vice versa.

Roughly, if two queries share some identical query result documents, then it is arguable that they are similar. Cross-references between two queries can be represented by the overlapping number of result documents (i.e., overlapping URLs) [9], as well as by the content similarity between resulting Web pages [24]. This last solution is time-consuming and can be relaxed by looking at the content similarity between Web-snippets that are present on the result documents, instead of considering the whole contents of the documents [17].

Anyway, all these feedback-based approaches are affected by the fact that queries representing different information needs might lead to the same results since one result can contain information about several topics. Thus, recently, hybrid solutions that combine both content- and feedback-based approaches have shown to be more effective [8].

In addition, inter-query temporal interval must be taken into account, especially when dealing with long-term historical data [4].

In our work, the TSDP query clustering algorithm uses a novel distance metrics μ, which results from the *convex combination* of 3 different kinds of metrics concerned with: (i) query content ($\mu_{content}$), (ii) inter-query temporal interval (μ_{time}), and (iii) the collaborative knowledge base collected by Wiktionary ($\mu_{wiktionary}$).

$$\mu = \alpha \cdot \mu_{content} + \beta \cdot \mu_{time} + \gamma \cdot \mu_{wiktionary} \qquad (1)$$

where $(\alpha + \beta + \gamma = 1)$.

So far, we have not considered any feedback-based metrics since they are generally expensive to compute, and also because they are not always available on WSE query logs.

In the following, we provide a description of each metrics by considering the distance between two queries q_1 and q_2.

Content-based ($\mu_{content}$). We propose a distance metrics $\mu_{content}$, which is computed by using an extended version of the *Jaccard* similarity coefficient.

Given two sample sets $\mathcal{A}$ and $\mathcal{B}$, the Jaccard index is defined as the size of the *intersection* divided by the size of the *union* of the two sample sets [31]:

$$Jaccard = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$$

Jaccard index is widely used in Information Retrieval and Natural Language Processing domains [14]. Typically, the sample sets correspond to the sets of words (i.e., terms) obtained from the two text strings we want to compare. Thus, the Jaccard index between two queries corresponds to the number of shared query terms divided by the union of query terms. However, according to Järvelin et al. [14] we decide to compute the Jaccard index on the basis of the n-grams instead of considering the two strings at a term-level. Basically, the two sample sets $\mathcal{A}$ and $\mathcal{B}$ are obtained extracting 3-grams from the two strings representing the two queries. Thus, $\mu_{content}$ can be written as follows:

$$\mu_{content}(q_1, q_2) = 1 - Jaccard_{3-grams}(q_1, q_2)$$

Other term-level metrics could be used including *cosine similarity*, *Dice similarity*, *Jaro-Winkler similarity* as well as character-level functions like *Levenstein distance* (i.e., edit distance) [6].

Time-based (μ_{time}). Time interleaving between issued queries plays a significative role, especially when dealing with long-term user sessions [25]. In fact, let us consider three queries q_i, q_j, and q_k that are similar from a content-based point of view. Suppose also that q_i and q_j were issued within a short-time interval while q_k was issued a long time after, then we must consider that q_i is more similar to q_j than q_k.

In order to represent this behavior, we introduce an exponential time decay over the difference between the timestamp values of the two queries we want to compare:

$$e^{-\lambda(|\tau(q_1) \ - \ \tau(q_2)|)}$$

Thus, μ_{time} can be written as follows:

$$\mu_{time}(q_1, q_2) = 1 - e^{-\lambda(|\tau(q_1) \ - \ \tau(q_2)|)}$$

Furthermore, the exponential decay metrics is also important to discriminate two re-submissions of the same query relative to two different user goals [32]. For instance, a user submitting the query "WSDM 2010" on August 13th, 2009, likely, has in mind to check the submission deadline of the query. The same query submitted a couple of months later by the same user is probably aimed at looking for news about the conference program. Even if the two queries are identical, the goal is different. The exponential decay will evaluate these two identical queries, completely different.

Wiktionary-based ($\mu_{wiktionary}$). Since average number of query terms is low, there is often a lack of semantic meaning associated with the queries that a WSE user issues [28]. In order to overcome this problem, we figure out that we could expand each query with its "wiktionarization". Basically, we exploit the Wiktionary data source for increasing the meaningfulness of each query, trying to overcome the lack of semantic information it usually carries on.

Wiktionary, as well as Wikipedia, is a knowledge base that is collaboratively constructed by mainly non-professional volunteers on the Web. According to [37], we call such collections *Collaborative Knowledge Bases*, in order to distinguish them from classical *Linguistic Knowledge Bases* like *WordNet* [7].

Collaborative knowledge bases like Wiktionary and Wikipedia have been recently used for increasing *semantic relatedness* between words in Natural Language Processing domain [37, 36, 18]. However, to the best of our knowledge, this feature has not yet been used for query clustering.

In order to wiktionarize the string associated with a query q, we have to first wiktionarize each of its terms. Let us suppose that $q = \{t_1, t_2, \ldots, t_l\}$ where each t_i is a term. Thus, we can define a function ω as follows.

DEFINITION 4.1 (SINGLE-TERM WIKTIONARIZATION ω). *Let t_i be a single-term string over that alphabet. The function ω takes the string t_i as input and produces a set $\mathcal{W}_i$ whose elements are set of strings obtained from the infor-*

mation collected by the Wiktionary data source:

$$\omega(t_i) = \mathcal{W}_i = \{\mathcal{G}_i, \mathcal{HYP}_i, \mathcal{SYN}_i, \mathcal{DT}_i, \mathcal{CAT}_i\}$$

In particular, the set $\mathcal{W}_i$ contains the following information associated with t_i and available on Wiktionary:

- a set $\mathcal{G}_i$ containing the first *gloss*;
- a set $\mathcal{HYP}_i$ containing the top-K *hyponyms*;
- a set $\mathcal{SYN}_i$ containing the top-K *synonyms*;
- a set $\mathcal{DT}_i$ containing the top-K *derived terms*;
- a set $\mathcal{CAT}_i$ containing the top-K *categories*.

Finally, we can define a function Ω, which is responsible of the wiktionarization of a multi-term string as follows.

DEFINITION 4.2 (MULTI-TERM WIKTIONARIZATION Ω). *Let T be a multi-term string over that alphabet such that $T = \{t_1, t_2, \ldots, t_l\}$. The function Ω takes the string T as input and produces a set $\mathcal{W}$ as follows:*

$$\Omega(T) = \mathcal{W} = \bigcup_{i=1}^{l} \omega(t_i) = \bigcup_{i=1}^{l} \mathcal{W}_i$$

where:

$$\bigcup_{i=1}^{l} \mathcal{W}_i = \bigcup_{i=1}^{l} \mathcal{G}_i \oplus \bigcup_{i=1}^{l} \mathcal{HYP}_i \oplus \bigcup_{i=1}^{l} \mathcal{SYN}_i \oplus \bigcup_{i=1}^{l} \mathcal{DT}_i \oplus \bigcup_{i=1}^{l} \mathcal{CAT}_i$$

Thus, the wiktionarization of a multi-term string T is a set $\mathcal{W}$ whose elements are sets obtained from the union of the corresponding elements contained in each W_i.

Given two queries q_1 and q_2, we first split the two queries in terms applying a splitting algorithm that also takes into account missing spaces, e.g. birthdaypartyideas.com is split into "birthday", "party", "ideas", and "com". Then, we apply the wiktionarization process described above for obtaining their "wiktionarized" representations $\Omega(q_1)$ and $\Omega(q_2)$, respectively.

Therefore, we build up the corresponding term-frequency vectors associated with $\Omega(q_1)$ and $\Omega(q_2)$, and we use a *cosine similarity* metrics for computing the distance between those vectors. According to the comparative analysis between different similarity metrics proposed by Cohen et al. [6], cosine similarity, on average, performs best with respect to other term-level meterics. Moreover, here we use a term-level distance metrics because the query strings are strongly enriched, resulting in a less sparse term-frequency vectors.

Finally, $\mu_{wiktionary}$ can be written as follows:

$$\mu_{wiktionary}(q_1, q_2) = 1 - cosine\_similarity(\Omega(q_1), \Omega(q_2))$$

5. EXPERIMENTS

5.1 Experimental Setup

This Section describes the setup phases for running, testing, and evaluating our TSDP query clustering algorithm.

We start choosing the 2006 *AOL* query log as the initial data set. This query log is a very large and long-term collection consisting of about 20 million of Web queries issued by more than 657 thousands users over 3 months (from 03/01/2006 to 05/31/2006). Moreover, the AOL query log is still publicly available throughout several Web mirror sites, despite the company decided to remove those data from its own servers on August 2006 after its release incident[4].

First of all, we preprocessed the query log for a data cleaning phase using a combination of *Python* and *AWK* scripts. In particular, we removed query log records containing both empty and "non-sense" query strings (i.e., query strings composed of only punctuation symbols). Moreover, we removed also all the *stop-words* from each query string. Then, we sorted the query log according to the length of its long-term user sessions in a decreasing order. Finally, we run the *Porter stemming algorithm* [22] for removing the commoner morphological and inflexional English endings from the terms of each query string.

We implemented the clustering algorithm presented in Section 4 totally in *Java*. That choice allowed us to rely on a suite of existing libraries for dealing with two main issues: (i) distance metrics, and (ii) Wiktionary data source.

In particular, concerning the first issue (i) we used *SecondString*[5], while for (ii) we relied on the *Java-based Wiktionary Library* (JWKTL)[6].

SecondString. SecondString is an open-source Java-based package of approximate string-matching techniques, which has been developed at the Carnegie Mellon University by Cohen et al. [5]. It supports a large number of non-adaptive distance functions like a wide range of metrics based on *edit distance*, including *Levenstein* distance, which assigns a unit cost to all edit operations, and the *Monge-Elkan* distance function [19]. Of course, it also implements a number of term-level distance metrics like the *Jaccard* similarity coefficient and the *cosine similarity*, as well as novel hybrid distance functions, which combine term-level and string-based matching schemes.

Java-based Wiktionary Library (JWKTL). JWKTL is a Java-based API that enables efficient programmatic access to the information contained in the English and German language editions of Wiktionary that has been developed at the Technische Universität of Darmstadt by Gurevych et al. [36].

JWKTL is based on freely available Wiktionary dumps[7] of different language editions in XML format. In order to provide a fast and easy access to the lexical semantic knowledge in Wiktionary, the output of the parser is stored using the *Berkeley DB* database library[8]. Then, for each Wiktionary entry, the API returns a Java object, namely a wrapper, which contains the extracted information.

Currently, the JWKTL API provides robust parsing of the English and the German Wiktionary editions and extracts structured information, including glosses, etymology, examples, quotations, translations, derived terms, characteristic word combinations, lexical relations, as well as links to other language editions of Wiktionary, Wikipedia articles, and external Web pages.

Figure 1 shows the architecture of JWKTL, and JWPL that is the analogous Java-based library for having programmatic access to the Wikipedia data source.

For our experiments, we downloaded the English Wiktionary XML dump file of 2009/04/29.

[4] http://sifaka.cs.uiuc.edu/xshen/aol_querylog.html
[5] http://secondstring.sourceforge.net/
[6] http://www.ukp.tu-darmstadt.de/software/jwktl/
[7] http://dumps.wikimedia.org/backup-index.html
[8] http://www.oracle.com/technology/products/berkeley-db/index.html

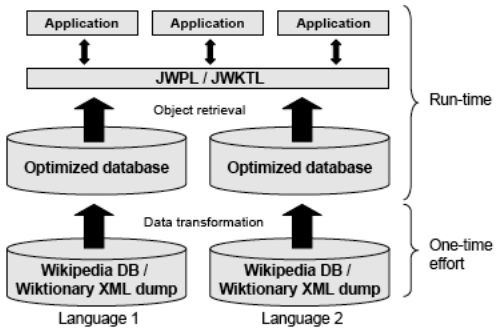


Figure 1: Architecture of JWPL and JWKTL [36].

First, we collect the *top*-1000 long-term user sessions extracted from the whole sorted query log as initial input data. The *top*-1000 long-term user sessions are the 1000 sessions containing the maximum number of issued queries.

Actually, we decided to drop the longest user session from that collection because it contains 198060 queries over 3 months, which means that user issued on average about 2200 queries per day. This is a huge number, especially if we consider that the second longest user session contains 8203 queries, which means an average of about 91 queries per day, that is a more reasonable number. Moreover, even the subsequent long-term sessions converge toward the order of thousand queries over 3 months, that is order of tens queries per day. Thus, it is possible that the very longest user session is the result of some *automatic* query-issuer, instead of a human user.

In order to analyze the behavior of our TSDP algorithm, we extract 2 long-term sessions from the *top*-1000, i.e, *top*-2. Moreover, we were also interested in verifying the results of our TSDP algorithm when running on shorter long-term sessions. Thus, we extract 2 user sessions of middle length (i.e., *middle*-2), and 2 of the shorter user sessions (i.e., *bottom*-2), respectively.

Therefore, we executed the following runs of our TSDP clustering algorithm on the *top*-2, *middle*-2, and *bottom*-2 long-term sessions, using different weighted-features for computing the distance function μ:

1. $\mu = \alpha \cdot \mu_{content} + \beta \cdot \mu_{time}$ where ($\alpha = \beta = 0.5$ and $\gamma = 0$);
2. $\mu = \beta \cdot \mu_{time} + \gamma \cdot \mu_{wiktionary}$ where ($\beta = \gamma = 0.5$ and $\alpha = 0$);
3. $\mu = \alpha \cdot \mu_{content} + \beta \cdot \mu_{time} + \gamma \cdot \mu_{wiktionary}$ where ($\alpha = 0.5$ $\beta = 0.2$ and $\gamma = 0.3$).

and we used two different values for the threshold t: $t_1 = 0.6$ and $t_2 = 0.75$.

For comparative results, we also implemented a *baseline* algorithm, which simply splits a long-term session using a timeout cutoff of 30 minutes. As for the TSDP, we run this baseline algorithm on the *top*-2, *middle*-2, and *bottom*-2 long-term sessions respectively.

Finally, we also run our TSDP algorithm on the whole *top*-1000 long-term sessions for the sake of completeness and for future evaluations. The output of this run is available for downloads at the home page of one of the authors[9].

[9] `http://miles.isti.cnr.it/~tolomei/tsdp-AOLtop1000-dump.tgz`

5.2 Results

The obtained results can be evaluated according to a *qualitative* approach, which is typically based on human efforts due to the lack of automatic *quantity*-based evaluators.

In this Section, we show and compare some sample clusters obtained using either the baseline algorithm and the other three runs of our TSDP algorithm on *top*-2, *middle*-2, and *bottom*-2 long-term sessions.

The results shown in Table 1 highlight that our clustering algorithm produces less clusters than the baseline solution. This is due, mainly, to the ability of our technique to discover sessions corresponding to tasks on which users take more than the half hour considered by the baseline. As it is shown next in this section, this is actually a valid point. Note that, due to the introduction of the exponential score decay, TSDP is also resilient to misclassifying a re-submission with a different user goal as part of sessions in which the previous instances of that query were submitted.

Table 2 shows two clusters obtained with either the *baseline* or the TSDP, which mainly contains *navigational* queries according to [3]. Moreover, the table indicates that several interleaving tasks might be present within a 30-minutes search session. However, using our TSDP algorithm, we can extract a single-task session by considering also queries that were issued after the timeout cutoff.

Looking at Table 3, which shows two related clusters sharing both *informational* as well *transactional* queries [3], we can state something similar to what we said about Table 2 before. In addition, here it is worth to notice the real benefit of integrating Wiktionary for computing the distance metrics on which our TSDP algorithm relies. In fact, the similarity relatedness between terms provided by Wiktionary is evident if we consider that TSDP algorithm puts into the same cluster two queries such as "80th birthday party favor" and "birthday party idea senior citizen"[10].

Again, Table 4 shows that our TSDP algorithm provides better quality clusters with respect to the baseline, identifying a single-task session. However, the task-based session here identified by our algorithm spans across a large time window, resulting into a "broad" session. Basically, a too long task-based session could either mean that the user has not achieved her goal during the whole session time or that she is simply interested in accomplishing the same task by re-formulating the same queries periodically [32]. However, this behavior of our TSDP algorithm might be due to the low value we assigned to the exponential time decay feature β when computing the function μ ($\beta = 0.2$). We plan to investigate how to modify that behavior by testing our algorithm with different values of β.

Finally, Table 5 provides an example in which both the baseline and our TSDP algorithm output the same cluster of queries. Moreover, if we look carefully at the query log, we can see that the next query is "albany ny home sale", which it was issued at the following time-stamp "2006-03-12 13:43:04". This query obviously deals with something similar with the previous two, i.e, looking for a home to buy. However, neither the baseline nor our TSDP algorithm was able to group those three queries into the same cluster. In particular, the baseline algorithm clearly did not consider

[10] Wiktionary entries for "80th" and "senior" show that they are related and this strong relationship overcomes the poor correlation in terms of time.

Table 1: Compare resulting clusters

session class	$\alpha/\beta/\gamma$	threshold	#queries	#clusters	avg #queries per cluster
top-2	*baseline*	0.6	3473	661	5.25
top-2	0.5/0.2/0.3	0.6	3473	238	14.60
top-2	*baseline*	0.6	3156	750	4.21
top-2	0.5/0.2/0.3	0.6	3156	266	11.86
middle-2	*baseline*	0.6	1128	392	2.88
middle-2	0.5/0.2/0.3	0.6	1128	479	2.35
middle-2	*baseline*	0.6	980	272	3.60
middle-2	0.5/0.2/0.3	0.6	980	506	1.93
bottom-2	*baseline*	0.6	366	120	3.05
bottom-2	0.5/0.2/0.3	0.6	366	204	1.79
bottom-2	*baseline*	0.6	272	107	2.54
bottom-2	0.5/0.2/0.3	0.6	272	195	1.39

Table 2: Looking for medical supplies: Baseline vs. TSDP

Baseline Queries			TSDP Queries		
Query ID	Query String	Time-stamp	Query ID	Query String	Time-stamp
			2053	vital medic suppy.com	2006-04-21 17:01:44
			2054	vital medic supply.com	2006-04-21 17:01:53
2062	imatchup.com	2006-04-21 19:03:12			
2063	yahoo.com	2006-04-21 19:10:57			
2064	medic supply.com	2006-04-21 19:12:06	2064	medic supply.com	2006-04-21 19:12:06
2065	medicalsupply.com	2006-04-21 19:13:16	2065	medicalsupply.com	2006-04-21 19:13:16
2066	etxgem.accpaconline.com	2006-04-21 19:19:19			
2067	medic supply.com	2006-04-21 19:20:50	2067	medic supply.com	2006-04-21 19:20:50
			2069	vital medic supply.com	2006-04-21 19:50:43

Table 3: Looking for organizing a birthday party: Baseline vs. TSDP

Baseline Queries			TSDP Queries		
Query ID	Query String	Time-stamp	Query ID	Query String	Time-stamp
			645	person favor 80th birthday	2006-04-17 19:55:11
			647	80th birthday party momento	2006-04-17 20:16:35
			648	80th birthday party favor	2006-04-17 20:19:04
			649	party favor 80th birthday party	2006-04-17 20:43:51
			658	80th birthday corsage	2006-04-17 21:20:07
			675	80th birthday craft	2006-04-18 18:39:51
676	queen heart crown	2006-04-18 18:45:25			
677	shindz	2006-04-18 18:50:55			
678	shindig	2006-04-18 18:50:59			
679	birthday party idea senior citizen	2006-04-18 18:57:43	679	birthday party idea senior citizen	2006-04-18 18:57:43

Table 4: Looking for shoes: Baseline vs. TSDP

Baseline Queries			TSDP Queries		
Query ID	Query String	Time-stamp	Query ID	Query String	Time-stamp
			789	diesel yo yo oxford	2006-03-27 10:07:15
			801	diesel yo yo oxford	2006-03-27 13:30:03
			802	diesel shoes	2006-03-27 13:34:07
			804	diesel shoes	2006-03-27 13:59:02
			814	diesel 6pm.com	2006-03-27 15:02:10
			867	diesel cloth u.k.	2006-03-28 14:06:48
1140	up	2006-04-04 08:03:27			
1141	up track	2006-04-04 08:04:18			
1142	lowel sun	2006-04-04 08:08:21			
1143	metlife	2006-04-04 08:12:28			
1144	aso	2006-04-04 08:17:50			
1145	lowel sun	2006-04-04 08:18:51			
1146	nhra	2006-04-04 08:26:32			
1147	diesel shoes	2006-04-04 08:27:49	1147	diesel shoes	2006-04-04 08:27:49
			1242	diesel shoes	2006-04-05 10:11:27
			1244	diesel shoes	2006-04-05 10:23:01
			1246	diesel shoes	2006-04-05 10:42:19
			1252	diesel shoes	2006-04-05 12:27:54
			2099	diesel shoes	2006-04-23 11:57:22
			2847	diesel shoes	2006-05-10 13:49:16

Table 5: Looking for home for sale: Baseline vs. TSDP

Baseline Queries			TSDP Queries		
Query ID	Query String	Time-stamp	Query ID	Query String	Time-stamp
12	buffalo ny	2006-03-12 12:03:12	12	buffalo ny	2006-03-12 12:03:12
13	buffalo ny home sale	2006-03-12 12:03:42	13	buffalo ny home sale	2006-03-12 12:03:42

that query to be a part of the same cluster because it was issued after the 30-minutes threshold. Instead, regarding our TSDP algorithm, we claim that it lacks of a concept-based semantic knowledge, although the Wiktionary collection is a valuable source of semantic information on a term-based level. Therefore, we plan to investigate how to enrich our model by introducing the Wikipedia knowledge base when computing the distance metrics μ.

6. CONCLUSIONS AND FUTURE WORK

This paper shows a technique to split into user sessions a very large, long-term log of queries submitted to a Web Search Engine (WSE). Despite the large amount of papers using the concept of user session, only a few of them addressed explicitly the problem of performing a correct split of the stream of queries into sessions. We formally introduce the *Task-based Session Detection Problem* as the problem of extracting from a stream of query submitted by a user subsequences of queries all related to the same user goal. We, then, devise a clustering-based solution leveraging three different aspects of queries: content, submission time, semantics. In particular, we have defined a methodology we called *Wiktionarization* to infer the semantics of a query through the analysis of the wiktionary entries of its composing terms. Eventually, we qualitatively test the technique by comparing the results obtained by our clustering algorithm with those obtained by adopting the commonly used algorithm of splitting a query stream into sessions of queries submitted within a 30 minutes timeframe. Results show the superiority of our approach. It is, in fact, able to capture the similarity of queries differing a lot in terms of content, but not in terms of meaning.

As a first immediate future step, we will evaluate the possibility of inserting more metrics in the convex combination shown in Equation 1. We are currently experimenting with a metrics involving Wikipedia, in addition to wiktionary, as a way to infer the semantics of a query also for those queries not having terms defined in wiktionary. Also relevance-feedback will be considered, clicked results and their relative content will be used to enrich a query definition. As a drawback, the clustering algorithm will involve more complex computations during the distances evaluation phase. A study of an efficient, and scalable, algorithm for clustering queries to detect user-based session will be subject of another research work.

Apart from picking these "*low hanging fruits*", we consider this work not as a final results but, rather, as a starting point from which a lot of new, and challenging, problems can be defined. This novel way of searching the Web "*by tasks to be executed*" instead of "*by documents to be retrieved*" has to be enabled by new mechanisms, which should be able to deal with Web-mediated processes. Our first research challenge is to evaluate this belief by exploiting the results of analysis done with the techniques introduced in this paper to devise a way to associate meaningful semantic labels with the extracted tasks, i.e., task-based sessions. This large knowledge base will constitute a starting point for building models of users' behaviors. Another research challenge is to devise a novel recommender system that goes beyond the simple query suggestion of modern WSEs. Our system has to ex-

ploit the knowledge base of Web-mediated processes and the learned model of users' behaviors, to generate complex insights and task-based suggestions to incoming WSE users.

7. ACKNOWLEDGMENTS

This research has been partially funded by the IST FP7 European Project S-Cube Grant Agreement no. 215483.

8. REFERENCES

[1] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00*, pages 407–416. ACM, 2000.

[2] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM '08*, pages 609–618. ACM, 2008.

[3] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[4] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW '05*, pages 2–11. ACM, 2005.

[5] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78, August 2003.

[6] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, 2003.

[7] Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[8] L. Fu, D. H.-L. Goh, S. S.-B. Foo, and J.-C. Na. Collaborative querying through a hybrid query clustering approach. In *ICADL*, pages 111–122, 2003.

[9] N. S. Glance. Community search assistant. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 91–96, New York, NY, USA, 2001. ACM.

[10] D. He and A. Göker. Detecting session boundaries from web user logs. In *In Proceedings of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, pages 57–66, 2000.

[11] D. He and D. J. Harper. Combining evidence for automatic web session identification. In *IPM*, pages 727–742, 2002.

[12] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[13] B. J. Jansen and A. Spink. How are we searching the world wide web?: a comparison of nine search engine transaction logs. *IPM*, 42(1):248–263, 2006.

[14] A. Järvelin, A. Järvelin, and K. Järvelin. s-grams: Defining generalized n-grams for information retrieval. *Information Processing and Management*, 43(4):1005 – 1019, 2007.

[15] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*, pages 699–708. ACM, 2008.

[16] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05*, pages 391–400. ACM, 2005.

[17] K. W.-T. Leung, W. Ng, and D. L. Lee. Personalized concept-based clustering of search engine queries. *IEEE TKDE*, 20(11):1505–1518, 2008.

[18] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08*, pages 509–518. ACM, 2008.

[19] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[20] A. L. Montgomery and C. Faloutsos. Identifying web browsing trends and patterns. *Computer*, 34(7):94–95, 2001.

[21] H. C. Ozmutlu and F. Çavdur. Application of automatic topic identification on excite web search engine data logs. *Inf. Process. Manage.*, 41(5):1243–1262, 2005.

[22] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[23] F. Radlinski. Query chains: Learning to rank from implicit feedback. In *In KDD*, pages 239–248, 2005.

[24] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 344–350, New York, NY, USA, 1995. ACM.

[25] M. Richardson. Learning about the world through long-term query logs. *ACM TWEB*, 2(4):1–27, 2008.

[26] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. pages 355–364, 1997.

[27] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[28] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[29] F. Silvestri, R. Baraglia, C. Lucchese, S. Orlando, and R. Perego. (query) history teaches everything, including the future. In *LA-WEB*, pages 12–22, 2008.

[30] A. Spink, M. Park, B. J. Jansen, and J. Pedersen. Multitasking during web search sessions. *IPM*, 42(1):264–275, 2006.

[31] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005.

[32] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158, New York, NY, USA, 2007. ACM.

[33] J. R. Wen, J. Y. Nie, and H. Zhang. Query clustering using user logs. *ACM TOIS*, 20(1):59–81, 2002.

[34] D. R. Yahoo and D. E. Rose. Understanding user goals in web search. In *WWW '04*, pages 13–19. ACM, 2004.

[35] J. Yi and F. Maghoul. Query clustering using click-through graph. In *WWW '09*, pages 1055–1056. ACM, 2009.

[36] T. Zesch, C. Müller, and I. Gurevych. Extracting

lexical semantic knowledge from wikipedia and wiktionary. In *LREC '08*, 2008.

[37] T. Zesch, C. Müller, and I. Gurevych. Using wiktionary for computing semantic relatedness. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 861–866. AAAI Press, 2008.

6 Dubois, D. J.; Nikolaou, C.; Voskakis, M.: *A model transformation for increasing value in service networks through intangible value exchanges*. In : International Conference on Service Science, ICSS2010, May 13-14, Hangzhou, China, 2010 – accepted for publication.

A Model Transformation for Increasing Value in Service Networks through Intangible Value Exchanges

Daniel J. Dubois
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milan, Italy
e-mail: dubois@elet.polimi.it

Christos Nikolaou, Manolis Voskakis
Transformation Services Laboratory
University of Crete
Heraklion, Greece
e-mail: {nikolau,voskakis}@tsl.gr

***Abstract*—One of the main goals in service science is to find efficient ways to analyze and increment the value in a service network. The approach we propose in this paper is to increase the agility of the system in such a way that the network and the underlying business processes are able to spontaneously react to changes in the requirements or in the environment. This is done by making extensive use of knowledge transfer and intangible interactions among network participants. The final outcome is a transformation and analysis methodology that may be applied to a wide variety of service networks with the aim of finding possible reconfigurations for increasing customer satisfaction, reducing transaction risks, and therefore increasing the overall value of the network.**

Keywords-component; value networks, business process management, service networks, value analysis

I. INTRODUCTION

Modern business systems are becoming more complex over time since the cost and speed of storing and exchanging information is becoming lower and the number of participants is becoming higher. These systems are not only large, but they are set in an environment that is always changing both in requirements and in the number of participants. Therefore we need an intelligent way to analyze them and provide a methodology to keep or improve the value of participation into the system even in such dynamic conditions. The ability of the system to react to environmental perturbations is defined as the agility of the system. In this context we propose a methodology for transforming a business network, known also as a *service network*, to improve the value for their participants even if their requirements change over time or some participant acts in a way that decreases the value of the network. The two key technologies that we are exploiting are: (1) value estimation based on revenues and offerings from each participant; (2) risk reduction and transparency that emerges after making extensive use of intangible interactions and knowledge transfers. The methodology we propose has been applied in the context of a traditional car sharing company [20] and its possible transformation into a more "agile" network. The final result is that a business network designer has now a method for taking strategic decisions whether to modify its business network or not, basing his decisions on the environment/participants characteristics gathered over a certain period of time.

Related work on this type of business networks, known also as service value networks, or simply service networks may be found in [15, 14, 8, 16, 5, 4, 13]. In these networks entities are either companies or different roles within a company, connections are offerings from one entity to another. These networks may be agile, in such a case agility is their level of flexibility in dealing with changing requirements [17]. In [7] we can also find a definition of value as "*benefits of an agent accrued by his participation in the network minus any costs involved in setting up the network links directly and indirectly*". However there is a problem with the actual evaluation of this "*amount of benefits*". The evaluation method we will base the rest of the paper is explained in [9]: in this work all the interactions are expressed as offerings and payments occurred per time unit, then the total revenues for all offerings exported by each participants are computed and used to estimate future revenues. However according to this work the total value is not simply a subtraction of revenues minus costs, but there is another value-contributing component called *Satisfaction Index*: it measures the perceived preference for a relationship.

Other metrics for evaluating service networks are proposed in the following works: the work from Parolini [15] describes a methodology called "Value NET" for taking strategic decisions on the service network by doing a qualitative analysis over it to identify bottlenecks, dominant relationships, and to predict the effects of possible structure/relationship modifications; in Allee works [1,2,3,4] there are other metrics related to the structure of the network such as stability and risk, she also points out an analysis methodology that focuses on the intangible exchanges from the network; additional works that try to understand and evaluate the value of services networks may be found in [19,5,8,18].

Our way for evaluating value takes some inspiration from all the works above, especially [9,1] since structure is not stable in an agile dynamic network, therefore we will focus on the measurement on actual revenues, costs, and satisfaction of each participant of the network.

Another concept that this work relates to is the concept of business ecosystem coordination mechanism [12], in which business networks are kept together by so-called

"*keystone companies*" (for example eBay and Amazon) whose objective is not to remove competitors, but to cooperate with them by establishing some value-creating virtuous circles and obtain an emergent value [11].

The final key concept taken from literature is the concept of trust, defined as "*an attitude of positive expectation that one's vulnerabilities will not be exploited*" [10]. In this context trust is needed when dealing with risk, and the usual way to reduce risk is to increase the knowledge transfers (intangible interactions) in the system. A possible way that we will use for evaluating trust is to define it as *the willingness to pay for not having the risk* (insurance), and representing it as a value transfer (offering) from the Trustee to another entity that guarantees for the Trustor.

The rest of this paper is organized as follows. Section II discusses the problem. In Section III we propose a possible methodology for dealing with such problem. Section IV gives an example of how the methodology may be applied in the context of a car sharing company. Finally Section V concludes the paper and gives an overview of possible new research directions.

II. PROBLEM STATEMENT

In this section we define a common pattern in service networks and provide a problem formalization.

A. Model and Assumptions

Under the same definition of service network discussed in Section I, we will consider the following simplified pattern in which there are only two entities: a service provider and a user (See Figure 1). The provider offers a service and the user buys that service, obviously there may be many competing providers and customers. The main assumptions in this network is that the environment is not controllable by participants (risk), there is lack of knowledge for matching users to the most suitable service providers (no transparency), and lack of knowledge for allowing a user to act as a provider (no information flow).

B. The Problem

The problem we are facing is to find a qualitative and quantitative way to decide if a network transformation that tries to reduce risk, that tries to increase transparency among participants, and that allows knowledge transfer, may be more profitable than the original one. In particular we focus our attention to the insertion of a new entity in the network with the role of "matchmaker" between providers and users. Its purpose is to reduce risk and allow knowledge transfer using a reputation management system.

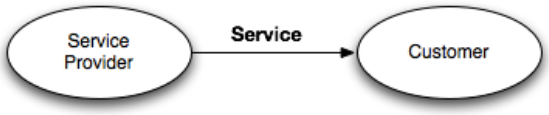


Figure 1. Service Network Fragment.

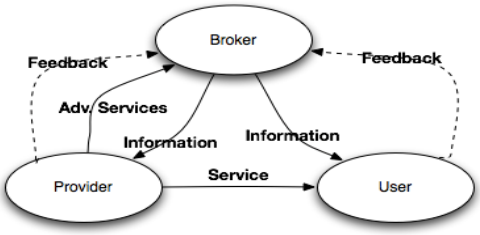


Figure 2. Transformed Network Fragment.

III. MODEL TRANSFORMATION

In this section we explain how the original network transfer discussed in the previous section may be transformed into a new fragment.

A. General Idea

The first model represents one of the traditional ways of creating value. Common firms tend to focus on their products and/or their services without sharing knowledge or having an efficient feedback mechanism from their customers. This way of business organization makes them unable to self-adapt to market condition and therefore to quickly respond to new challenges. To address these issues and provide a possible transformation we had an idea of a second network where each participant focuses on its core competencies and on the appropriate alliances to increase its profitability and flexibility to changes. These participants will still act as (more specialized) service providers and customers, but with a third type of participants that focuses on the transfer or knowledge, on managing feedbacks, and on making the emergence of such "agility" possible.

B. The Transformation

In the first network we have two roles: the service provider who offers a service and the user who receives the service (see Figure 1).

In the second model we have one new entity: a broker who supports and coordinates the interactions between the provider and the user. The broker provides to the user all the possible providers of the service he asked. Moreover the broker cooperates with the providers who advertise their services or products via the broker. In addition the broker collects user feedbacks and informs future users about the quality of available service providers using a reputation mechanism. This is very important because it reduces the risk for the users of being cheated. Furthermore the services offered above increase users' satisfaction and loyalty, which means more value for the network. To keep the network simple we are omitting some entities that produce some costs and revenues (for example labor costs, depreciation, utilities, and government incentives), however the associated value exchanges will be considered as inner costs and revenues.

C. Value Evaluation

In this subsection we show how to estimate the value in several situations, this information will then be used to take decisions about the profitability of participating in the network. The outcome of these calculations will be to find

out under which circumstances it is profitable for an entity to participate in the network. We define first some parameters that will be used in the final analytical definition of value. Each of these parameters is evaluated after every specified time interval that should be properly chosen by the business process manager based on the situation. To distinguish different time intervals we use the notation $T_1, T_2, \ldots T_N$. We define also c_{ij}^k as the cost component k that participant i pays to j to consume his services, Ic_{ij}^k as the *inner costs* associated to the same transaction (which correspond to payments to entities that are not represented in the network, such as labor cost and utilities), and finally Ip_{ji}^k as the inner profits that come from external entities involved in the process such as, for example, government incentives.

Revenues are the payments that each participant receives for the services he offers. Revenues of an entity i coming from another participant j are defined as the amount the other participant pays to consume services that come from entity i, plus the inner profits associated to the same transaction:

$$R_{ij}(T_N) = \sum_k (c_{ji}^k + Ip_{ji}^k). \quad (1)$$

Costs are the amount of money each participant has to pay to be able to participate in the network and offer his services. Costs of an entity i coming from another participant j are the payments for the services that node i consumes from j during his participation in the network, plus the inner costs associated to the same transaction:

$$C_{ij}(T_N) = \sum_k (c_{ij}^k + Ic_{ij}^k). \quad (2)$$

Satisfaction index is an important factor that measures the importance of a relationship and the willingness to repeat it. Its analytical definition and evaluation depends on the actual problem.

Past values of Revenues, Costs, and Satisfaction index can be used to predict future expected values of the same parameters using, for example, autoregressive with mobile average models as done in [9], or simply using the last actual evaluation of the parameters, as we have done in Section IV since past data information is limited. To refer to the expected values we simply put a dash on top of the parameter name.

Finally the **total value** for each participant i at time T_N is the sum of the value that comes from the relationship with any other participant j:

$$v_i(T_N) = \sum_j v_{ij}(T_N). \quad (3)$$

Where the definition of the relationship value v_{ij} is different for customers (entities that do not have direct revenues) and other entities. For customers it is defined this way:

$$v_{ij} := u_{ij}^k(\overline{\delta SAT_{ij}^k}, C_{ij}^k). \quad (4)$$

where $u(\ldots,\ldots)$ represents the utility function that measures the value of the customer who uses the service.

For all the other entities it is defined this other way:

$$v_{ij}(T_N) = \delta\overline{SAT_{ij}}(\overline{R_{ij}(T_N)} - \overline{C_{ij}(T_N)}). \quad (5)$$

These definitions are different because the final purpose of a customer is simply to obtain the service in the most convenient way, and the convenience is expressed by the (problem-dependent) utility function definition. For the other entities it is measured by the expected profits (defined as expected revenues minus costs) times the value of the relationship (expressed by the satisfaction index). If the expected profits are smaller than the ones of competing networks, then the potential participants will not enter and the service system has a questionable future. It is important to note that the time horizon considered for deriving the value of a service system is a parameter that has to be properly set; it must be long enough to compensate for the changes of the dynamic system and short enough to offer the right incentives for updating the participants' strategies.

IV. Case Study: A Car Sharing Company

This section shows how the methodology we described in the previous section can be applied to estimate the value of each participant. In our example we will use the car sharing company of Milan. The car sharing system is a system for motorized personal mobility that is alternative to traditional public/private transportation. The system gives the opportunity to the customers to use a car from half a hour to several days and the access to the service is completely automatic.

A. Traditional Car Sharing

The traditional car sharing has two entities. The car sharing provider, who offers car sharing services and owns all the cars; and the user who consumes the service. In this case the provider is unique and the customer may either cooperate with him or leave the network.

B. Car Sharing after Transformation

In the second model we have three different roles: a *provider***,** who offers his car(s) for sharing and can be either a single car owner or a company; a *user*, who chooses to use car sharing service system instead of buying a car; a *broker*, who collects information from both the providers and the users and coordinates their cooperation. The broker gives to the users information about the availability of cars, and to the providers information about the availability of customers. After this transformation, more participants will be able to join because any car owner may become a provider and therefore the network will be able to spontaneously grow and serve more users with a more differentiated offer.

C. Model of Value

In this section we present the equations we used to estimate the value of this service system. All the symbols are explained in Table 1 and costs/revenues/values, which are partly derived from publicly available data [20], are in Euros. For the traditional model revenues of the provider depend on the usage of the cars from the customers:

$$R_{provider} = (Fc + Ph \times h + Pk \times g) \times Nu + i \times Nc. \quad (6)$$

The costs of the provider depend on the number of customers and the car maintenance cost:

$$C_{provider} = Nc\times(Pi+Pp+a+m)+k\times Nc\times g+ \\ +Pm\times k\times Nc+(P_b+Pc+p)\times v\times Nu+ \\ +Of+Oc\times Nc+Or\times r\times Nu+Ou\times Nu. \quad (7)$$

The expected value of the provider is the difference between the revenues and the costs times a satisfaction index of 1 since there is only one provider and customers are assumed equal. The expected value of the user, defined using the custom utility function below, depends on his satisfaction and other parameters that have to do with the cost of using the service:

$$v_c = u_{ij}^k(\overline{\delta SAT_{ij}^k}, C_{ij}^k) := \frac{Sup}{(Fc+Ph\times h+Pk\times k)\times Nu}. \quad (8)$$

In the new model we have 3 entities. So we will estimate values and costs for each one of them. Total revenues of the provider are the sum of all the revenues he receives from both user and broker. Revenues of the provider coming from the broker are $R_{p,b}$=0, revenues of the provider coming from the user depend on the number of the users, the hours and the kilometers that the each user will use the car:

$$R_{p,u} = Pk\times k\times Nc+h\times Nc\times Ph+Ou\times Nc. \quad (9)$$

Total costs of the provider are the sum of the costs coming from the broker and the user. Costs due to the broker are the annual fee paid from the provider to the broker and the commission that the broker receives for any transaction between the provider and the user:

$$C_{p,b} = Fp + c\times R_{p,u}. \quad (10)$$

Costs due to the user are the costs of the depreciation of the car due to the car sharing with the user:

$$C_{p,u} = Nc\times(a+m+Pp)+k\times Nu\times Pm+ \\ +(P_b+p+Pc)\times v\times Nu. \quad (11)$$

Consequently total costs of the provider are:

$$C_p = C_{p,u} + C_{p,b}. \quad (12)$$

Expected revenues of the provider are:

$$\overline{R_p} = R_{p,u}\times Ssu+ R_{p,b}\times Spb. \quad (13)$$

The expected costs of the provider are:

$$\overline{C_p} = C_{p,b}\times Spb+C_{p,u}\times Ssu. \quad (14)$$

Total revenues of the broker are the sum of the revenues coming from the other 2 entities. The revenues of the broker due to the provider are:

$$R_{b,p} = Fp\times Np+c\times R_{p,u} \quad (15)$$

and the revenues coming from the user are:

$$R_{b,u} = Nu+Fc \quad (16)$$

consequently:

$$R_b = R_{b,p} + R_{b,u} \quad (17)$$

Total cost of the broker is the sum of any costs that broker has, to have the ability to offer his services to the other participants:

$$C_b = Of+Or\times r\times i+Ou\times i. \quad (18)$$

The expected cost of the broker is the cost due to the user multiplied by the satisfaction of user from their relationship plus the costs due to the provider multiplied with its satisfaction:

$$\overline{C_b} = C_{b,u}\times Sbu + C_{b,p}\times Sbp. \quad (19)$$

TABLE I. PARAMETERS USED IN THE CAR SHARING MODELS

Name	Sym.	Values Mod. 1	Values Mod. 2
Number of cars	**Nc**	200	200
Insurance	**Pi**	2000	1000
Number of customers	**Nu**	4000	4000
Depreciation per car	**a**	3600	900
Maintenance per car	**m**	1500	375
Annual cost of parking per car	**Pp**	4000	1000
Total km per car	**k**	5000	5000
Annual fee of the customer	**Fc**	120	60
Annual fee of the provider	**Fp**	0	60
Price per hour	**Ph**	2.50	2.50
Hours per year	**h**	300	300
Price per km	**Pk**	0.60	0.60
Gas consumption per km per car	**g**	0.20	0.20
Price km maintenance per car	**Pm**	0.05	0.05
Bank fee per invoice	**P_b**	0.10	0.10
Controvercy management per invoice	**Pc**	1	1
Total invoices per customer	**v**	6	6
Fixed operational costs (information system)	**Of**	40000	40000
Operational costs per car	**Oc**	200	200
Operational costs per customer	**Ou**	50	50
Government incentives per car	**i**	1000	1000
Operational costs per reservation	**Or**	1	1
Reservations per customer	**r**	50	30
Postage cost per invoice	**p**	0.50	0.50
Number of providers	**Np**	50	50
Commission of broker	**c**	0.30	0.30
SAT index of broker (from provider)	**Sbp**	1	1
SAT index of broker (from user)	**Sbu**	1	1
SAT index of provider (from broker)	**Spb**	1	1
SAT index of provider or seller (from user)	**Ssu**	1	1
SAT index of user (from provider or seller)	**Sup**	0.1	0.2
SAT index of user (from broker)	**Sub**	0.2	0.2

Expected revenues of the broker are:

$$\overline{R_b} = R_{b,p}\times Spb+ R_{b,u}\times Sub. \quad (20)$$

Finally the cost of the user is the sum of the costs that are due to the provider and the broker. The cost of the user due to the broker is:

$$C_{u.b} = Fc\times Nu. \quad (21)$$

The cost of the user due to the provider depends on the kilometers and the hours he will use the car sharing service and the gas he will consume:

$$C_{u,p} = k\times Pk\times Nc+Ph\times h\times Nc+g\times k\times Nc+Pi\times Nc. \quad (22)$$

The expected value of the user is calculated with this custom utility function:

$$v_c = u_{ij}^k(\overline{\delta SAT_{ij}^k}, C_{ij}^k) := \frac{Sup}{C_{u.b}} + \frac{Sub}{C_{u.p}}. \quad (23)$$

D. Value Analysis

We calculated all the values for both models using the values of Table 1. Then we have seen that the variation of some parameters of such table is able to change the analysis results, and therefore the winning network.

The variation example we will show focuses on the fixed costs (maintenance, insurance, parking, and depreciation): in the first network all these costs are charged to the user, while in the second network it is assumed to be lower since the car is not shared 100% of the time (providers in the second model can decide when to share their car and when to reserve it for their personal use). So if it is shared for example 25% of the time (and the rest of the time is used for their personal use), we can assume that fixed costs are 75% less with respect to the first model. The results of the simulation (as can be seen in Table 2) show that if fixed costs in the second model are 25% of the first model, then the second model is more profitable for all the participants, while if the fixed costs are 50%, than the first model would be better (because some participants have negative value). In conclusion we expect that the second network would become better than the traditional one if users that already own a car, start sharing their car when they are not going to use it, so that only a relatively small amount of fixed costs is charged to other users.

TABLE II. EXPERIMENTS ON VALUES WHEN CHANGING THE COSTS

Increase in Expected Values w.r.t. to the first	**Fixed costs of 2nd model are 25% w.r.t. the first**	**Fixed costs of 2nd model are 50% w.r.t. the first**
User 1st model	65 (normalized)	65 (normalized)
User 2nd model	1439 (normalized)	1258 (normalized)
Provider 2nd model	121540 (Eur)	-333460 (Eur)
Broker 2nd model	168000 (Eur)	168000 (Eur)

V. CONCLUSIONS

In this paper we presented a methodology for transforming a recurrent pattern in service networks to increase their agility. We have seen that, by collecting information on the actual configuration of the network and its context at any given period of time, it is possible to guide the service network evolution toward a more valuable configuration. This may be done analyzing direct context characteristics (such as the internal costs for providing some services), and indirect ones (such as the reaction of other participants to external changes). The study is supported by a technique for estimating the value and therefore to support the final decision whether to reconfigure the network or not.

Possible future research is the development of software modules to automate such transformation in current Business Process Management architectures. Another possible research direction can be to find and study new transformation patterns and to improve the value estimation technique.

ACKNOWLEDGMENT

This research has been partially funded by the European Community's FP7/2007-2013 Programme, grant agreement 215483 (S-Cube).

REFERENCES

[1] V. Allee. “A value network approach for modeling and measuring intangibles,” Proceedings of Transparent Enterprise Conference, Madrid, 2002.

[2] V. Allee. “Reconfiguring the value network,” Journal of Business Strategy, vol. 21, num. 4, pp. 36–39, 2000.

[3] V. Allee. “A value network approach for modeling and measuring intangibles,” Proceedings of Transparent Enterprise, Madrid, 2002.

[4] V. Allee. “The future of knowledge: Increasing prosperity through value networks,” Butterworth-Heinemann, San Francisco, 2003.

[5] Barlow and F. Li.. “Online value network linkages: integration, information sharing and flexibility,” Electronic Commerce Research and Applica tions, vol. 4, pp. 100–112, 2005.

[6] R. C. Basole and W. B. Rouse. “Complexity of service value networks: conceptualization and empirical investigation,” IBM Systems Journal, vol. 47, num. 1, pp. 53-70, 2008.

[7] F. Bloch and M. O. Jackson. “The formation of networks with transfers among players,” Journal of Economic Theory, vol. 133, num. 1, pp 83–110, 2007.

[8] D. Bovet and J. Martha. “Value nets: breaking the supply chain to unlock hidden profits,” Wiley, 2000.

[9] NS Caswell, C. Nikolaou, J. Sairamesh, M. Bitsaki, CD Koutras, and G. Ia- covidis. “Estimating value in service systems: A case study of a repair service system,” IBM Systems Journal, vol. 47, num. 1, pp. 87–100, 2008.

[10] D. Gambetta. “Can we trust trust. Trust: Making and Breaking Cooperative Relations,” University of Oxford, pages 213–237, 2000.

[11] M. Iansiti and R. Levien. “Strategy as ecology,” Harvard Business Review, vol. 82, num. 3, pp. 68–78, 2004.

[12] M. Iansity and R. Levien. “The keystone advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability,” Harvard Business School Press, 2004.

[13] Kothandaraman and D. T. Wilson. “The future of competition value-creating networks,” Industrial Marketing Management, vol. 30, num. 4, pp. 379–389, 2001.

[14] Normann and R. Ramirez. “From value chain to value constellation: designing interactive strategy,” Harvard business review, vol. 71, num. 4, pp. 65–77, August 1993.

[15] C. Parolini. “The value net: A methodology for the analysis of value-creating systems,” Proceedings of the Strategic Management Conference, 1996.

[16] F. K. Pil and M. Holweg. “Evolving from value chain to value grid,” MIT Sloan Management Review, vol. 47, num. 4, p. 72, 2006.

[17] B. Rouse. “Agile information systems for agile decision making,” Agile Information Systems: conceptualization, construction, and management, 2006.

[18] Straub, A. Rai, and R. Klein. “Measuring firm performance at the network level: A nomology of the business impact of digital supply networks,” Journal of Management Information Systems, vol. 21, num. 1, pp. 83–114, 2004.

[19] B. Wetzstein, et al. “Towards monitoring of key performance indicators across partners in service networks,” In Workshop on Service Monitoring, Adaptation and Beyond, 2008.

[20] Guidami s.r.l., “IoGuido: Websites of Milan Car Sharing Companies,” (1) http://www.carsharingitalia.com (2) http://www.guidami.net (italian language).

7 Cappiello, C.; Pernici, B.: QUADS: Quality-Aware Design of dependable Service-based processes. To be submitted to Transactions on Software Engineering.

QUADS: Quality-Aware Design of dependable Service-based processes

Cinzia Cappiello and Barbara Pernici

Abstract

Service-based processes are typically executed by composing and invoking a number of available Web services. Such services are often not under the control of process designers since they are offered by external providers. This introduces critical dependencies between service-based processes themselves and the services they are exploiting. These last ones, in fact, could fail without notice or be unavailable for undetermined time intervals. Failures may be of various nature and they can concern either the inability to provide a given service or a loss in the service quality. Therefore, in service-oriented systems in which applications are required to have a high level of autonomy, processes should be appositely designed to satisfy dependability requirements even in faulty situations. The process dependability could be improved by enriching the design of the applications with monitoring features, repair mechanisms in order to trigger suitable corrective actions when failures occur, and/or other preventive actions able to decrease the risk of failure. The literature proposes various corrective and preventive strategies to improve dependability and their effectiveness depends on the application context and users' requirements. In fact, each strategy is characterized by different properties (e.g., complexity, execution time, architectural constraints) and its impact on the process can be positive for some features, but also negative for other ones. Taking into account all these elements, we advocate that in order to cover all needs that may arise and guarantee a dependable application, the most suitable strategies should be selected and programmed at design time. For this reason, in this paper we propose a design for dependability approach that supports process designers in the usage of the available preventive and corrective improvement actions that enhance dependability.

Index Terms

service-based process, dependability.

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy

I. INTRODUCTION

In many software applications, users tend to require the full-time system effectiveness and errors, faults, or systems crashes are not tolerated. The ability of a system to deliver services than can be trusted is called *dependability* [1]. Dependability is an aggregate concept that considers the following attributes: (i) Reliability: the probability that a system remains operational over a specified period of time; (ii) Availability: the probability that a system is operational at a specified point of time; (iii) Repairability: probability that a non-operational (failed) system can be made operational within a specified period of active repair time [2]. The three concepts are strictly correlated. In fact, in order to increase system reliability, it is necessary to adopt mechanisms that guarantee system availability and/or mechanisms that minimize repair time. The development of a dependable system should include techniques to prevent faults, to correct errors, and to reduce the number or severity of faults. These techniques are in general more effective in environments where an extensive knowledge about the software applications and the running context is available. However, there are systems such as the ones based on Web services in which the behaviour of the computing systems is often variable and unpredictable and the realization of a dependable application cannot only rely on an extensive analysis of the process and the related execution environment. In fact, in this context, business processes are often executed by means of service-based applications that are able to offer complex and flexible functionalities in widely distributed environments by composing different types of services. Such services are often not under the control of systems developers, but they are simply invoked to obtain a specific functionality. Therefore, while the service approach, on the one side, improves the processes in terms of interoperability and flexibility, on the other side, it introduces some issues related to possible unexpected service behaviors. The literature proposes a good number of approaches that deal with self-healing or self-adaptation of service-based applications to manage unforeseen changes. Most of the contributions address this issue by including in the service code exceptions handlers or rules that are triggered only when some specific and known events happen. Clearly, this approach does not cover all the needs may arise since all the available techniques are not considered. The approach presented in this paper is based on the idea that at design time the process designer should model the business process by introducing suitable techniques able to guarantee the process dependability. We assume that the service-based processes will

be managed by a platform, similar to the one described in [3], that allows the adoption of techniques to enable a flexible and adaptive execution. In such scenario, the available techniques for the improvement of the dependability are various and include both preventive and corrective strategies. In fact, dependability can be guaranteed by means of permanent process modifications that aim at avoiding failures or repair actions that are able to restore the system when a failure occurs. The selection of the most suitable strategies to adopt in a specific process is not a trivial task. It requires a thorough analysis of the characteristics of the available strategies and of their impact on the process. To the best of our knowledge, the literature does not provide any design principles and gudelines to support the process designer in this task. For this reason, this paper goes in the direction of the design for dependability approach and aims at offering a methodology to support the selection of the preventive and/or corrective strategies to adopt and thus to define the way in which the business process should be modified. The paper is organized as follows. Section II presents the rationale of the approach presented in the paper. Section III describes how processes are modeled and executed in a service-oriented environment. Section IV describes all the most important dependability improvement strategies that can be considered in the service-based applications field. Section V presents the overall methodology that aims at supporting the process designer in the process design and its maintenance over time. Finally, Section V-E discusses some examples able to show the weaknesses and strenghtness of the proposed methodology.

II. QUALITY-AWARE DESIGN OF SERVICE-BASED PROCESSES

Service-based applications are based on processes that are the result of the composition of different services selected at runtime. Service selection should guarantee the invocation of the best services available, taking into consideration process constraints, but also end-user preferences and the execution context. Web services are usually selected from a set of functionally equivalent services, that is, services which implement the same functionality but may differ for nonfunctional characteristics, i.e., Quality of Service (QoS) parameters [4]. Most of the time, the service discovery is mainly driven by non-functional properties of the services. In fact, assuming that the functional requirements are satisfied, the system should select the set of services that maximizes the process quality and also satisfies some quality constraints specified on the basis of the users' requirements or inferred from the characteristics of the execution context. Those constraints

usually define the minimum value that is considered acceptable for a specific quality parameter or provide information about the relevance of each quality dimension from the users' perspective.

The set of the suitable services to compose should be redefined each time that some changes occur. In fact, as stated in Section I, in the service-oriented approach, it often happens that services change without notice. Changes can be due to modifications related to both functional and non-functional aspects. The set of services should be also redefined in case of process redesign. In fact, the process redesign could alter the need for services or could simply affect the control worklow and in both cases changes may impact on the overall quality. In the former case, the insertion or elimination of services modify the values to consider in the quality evaluation while in the latter case changes in the workflow (e.g, insertion of check points, activities' parallelization or sequentialization) modify the way in which the quality values should be aggregated to calculate the overall process quality.

Since the process quality is one of the main element to consider in the selection of the services and in the redesign choices, we want to use it as the base for the design for dependability approach discussed in this paper. In our perspective, process quality could be one of the main drivers for the selection of the dependability improvement strategies to adopt in a service-based process.

III. Process and service model

This section we formally describe the service-based process model along with its QoS criteria. In particular, Section III-A provides the process and service definitions and discuss how these concepts are related in a service-based application. Section III-B specifies the set of the most relevant quality dimensions to consider in the process quality assessment and provides a formal model for their description and management.

A. Process and service specification

A business process $\mathcal{BP}$ is generally defined as a collection of related, structured activities or tasks that produce a specific service (or product) to serve a particular goal specified by one or more final users. Processes are usually modelled at a business level independently from implementation technology and platforms [5]. In fact, the process designer defines the workflow able to provide the desired output by linking simple activities through control structures (e.g., sequence, choice, cycle, or parallel) to form tasks t_i. Therefore, a business process $\mathcal{BP}$ results

from the combination of a set $\mathcal{T}$ of tasks t_i, which correspond to significant portions or stages of the process. A task can be seen a small procedure able to produce specific outputs by elaborating inputs received by other process activities. We can provide the formal definition of a task $t_i \in \mathcal{T}$ as follows:

$$t_i = \langle n_i, IN_i, OUT_i, f_i \rangle \tag{1}$$

where: n_i is the task name; IN_i and OUT_i are the sets of task inputs and outputs, respectively; $f_i : IN_i \rightarrow OUT_i$ is the transformation associated with the task.

Considering the implementation technology, in service-oriented environments, a business process can be executed invoking services that can also be selected at run time [4]. The definition of service $\mathcal{S}$ imposes a series of constraints such as: (i) services are self-contained, that is, they do not require context or state information of other services; (ii) services are connected to other services and clients using standard, dependency reducing, decoupled message-based methods such as XML document exchanges. Moreover, a service is a conceptual unit of work that takes one or more inputs and creates an output perceived as a tangible value by clients. A process can be composed of one or more services connected by control structures that constitute the orchestration mechanism.

Considering the provided definitions of "service" and "task", it is possible to establish a correspondence between these two concepts. More formally, at design time, a task t_i can be defined as the abstract representation of corresponding service operations to be executed. At run-time each task t_i has to be bound to a selected service that we call *concrete service* cs_j able to provide the desired functionalities. Each task t_i is characterized by functional and non-functional (e.g., quality features) requirements. The former refer to the operations that should be performed while the latter define expectations about quality aspects. Since several service providers may offer similar services, the binding function has to consider the set of the published concrete services $\mathcal{C}S$ and extracts the set $\mathcal{C}S_i \subset \mathcal{C}S$ that includes only the concrete services cs_j characterized by functional and non-functional features suitable for the execution of task t_i. In details, each concrete service cs_j is defined together with the set of its implemented operations OP_j. We will refer to $\text{cs}_{j,o}$ in order to define the invocation of the operation $o \in \text{OP}_j$ of Web service cs_j. At run-time, each task t_i is then associated with a corresponding concrete service operation $cs_{j,o}$ and the set of couples $< t_i, cs_{j,o} >$ defines the process execution plan $\mathcal{E}P$.

B. Quality model

1) Quality dimensions for processes: As described in Sections II and III-A, non-functional requirements define preferences and constraints about quality criteria that can be associated with the process execution. Therefore, the selection of the concrete services may depend on the quality features that the process should guarantee in order to satisfy the users' requirements. In general, each quality dimension $qd_k \in \mathcal{Q}D$ can be described as $\langle n_k, \mathcal{V}_k \rangle$, where n_k is the quality dimension name and $\mathcal{V}_k$ corresponds to either categorical or interval values. In the former case, the values V will be included in a specific admissible vector AV = $(v_{k1}...v_{kh})$ while, in the latter case V will be defined by its extremes, i.e., $V = [v_{kmin}, v_{kmax}]$. The actual values of the quality dimensions can be obtained at the execution time. In the service selection phase the process designer analyzes the *quality capabilities* that are the estimated quality values that the service provider publishes together with the operational description of the service. The service is considered as a candidate concrete service cs_j if the quality capabilities satisfy the service *quality requirements* expressed by the process designer and the other process stakeholders.

Quality dimensions are various and can be defined at different architectural levels (i.e, from hardware to user level). For our purposes, we refer to the quality dimensions commonly used in the current literature [4][6][7]. It is possible to identify a set $\mathcal{Q}D$ of relevant quality dimensions qd_k that can be decomposed in a set of (i) data quality dimensions ($\mathcal{D}Q$), (ii) process quality dimensions ($\mathcal{P}Q$), and (iii) provider quality dimensions ($\mathcal{P}rovQ$).

Data quality dimensions aim at evaluating a data collection with regard to users' requirements [8]. In a process, information is used either to control the process flow or to produce values for the users. Information is a relevant aspect in the quality evaluation and it cannot be useful if it is incorrect, incomplete or just not updated. In fact, the basic data quality criteria to guarantee are accuracy, completeness, and timeliness. Accuracy and completeness assess data along their correctness and numerical extension [9][10]. More formally, accuracy acc_j is defined as the proximity of a value *v* returned by cs_j to a value *v'* considered as correct [9]. Completeness $comp_j$ is defined as the degree to which a given data collection output of cs_j includes all the expected data values. Timeliness $time_j$ evaluates the validity of data along time [11]. It expresses how current exchanged data are for the subsequent services of cs_j or users that receive them. From this perspective, data can be useless because they are *late* for a specific task.

The measures of these data quality dimensions are considered as real numbers in the range of [0,1], where 1 represents the most desirable score. By considering the introduced dimensions, a low data quality level can be determined by value mismatch, missing data, and/or delay in response.

Process quality dimensions are related to the environment in which the process is executed and also evaluate the suitability of the technological infrastructure. Process quality criteria include availability and execution time. Availability av_j can be defined as the probability of the process, and thus of the concrete services, to be operational when they are invoked. Availability is a number in the range [0,1]. Execution time et_j considers the time interval between the time instant in which cs_j is invoked and the time instant in which the result is obtained. Execution time is usually proportional to the functional complexity of the service. In order to evaluate the suitability of temporal information, as discussed in [12] we need explicit temporal information, which are a maximum process duration and response times of asynchronous relationships between an invoking and a receiving node. We can apply [min, max]-intervals for all kinds of temporal information, given in a specified basic time-unit, which will be minutes or seconds. Expected service response times may stem from empirical knowledge (extracted from logs) or be estimated by experts.

Data and Process quality dimensions are strictly related to services and to the infrastructure in which they are executed. It is also necessary to consider the providers involved in the service provisioning. In fact, we can distinguish a process owner that is responsible for providing the whole service-based application and the different providers associated with the invoked concrete services. In order to evaluate providers' trustworthiness that can be the discriminant driver in the service selection phase, it is possible to consider two *provider quality* dimensions, that are: reputation and fidelity. Reputation rep_j provides a preliminary value of the reliability of the service by considering an evaluation of its provider. In general, reputation is the public opinion about the character or standing (such as honesty, capability, reliability) of a provider [13]. It is an objective measure that can be assessed by using a reliable and efficient mechanism to get service rankings from the execution environment [14]. in fact, the reputation of a service increases if its execution satisfies the requirements.

Fidelity fid_j considers a measure to define the degree of relationship between the process owner and the provider of a concrete service. If the process owner contacts often the same

providers, and thus has a strong relationship with them, fidelity can be a relevant driver for the selection of the concrete service from the service registry.

2) Quality Capabilities: Quality capabilities reflect the quality levels that can be guaranteed by a Web service provider. We define a capability c_{jk} as a restriction on the range of admissible values of the quality dimension qd_k for the j-th concrete service. More precisely:

$$c_{jk} = \langle Off_{jk}, p_{jk}, Price_j \rangle \tag{2}$$

where $\mathcal{O}ff_{jk} \subseteq \mathcal{V}_k$ represents the restriction on the range of admissible values. In this way, the provider defines, given a quality dimension, which are the actual values that are guaranteed. In addition, the provider also defines the price p_{jk} function: $Off_{jk} \rightarrow Price_{jk}$ that maps the dependency between the offered values and the price per use associated with such a provisioning. According to this model, the provider, during the publication process of a Web service, will specify the set $\mathcal{C}_j$ of the supported service capabilities. Therefore, a concrete service can be defined as:

$$cs_j = \langle n_j, IN_j, OUT_j, f_j, C_j \rangle \tag{3}$$

where: n_j is the task name, IN_j, OUT_j and f_j correspond to the same fields used in the task description (see Equation 1) and $\mathcal{C}_j$ describes the non-functional properties.

At run time, the description of the concrete service will be associated with its instance cs_j^z. Each instance can be described as the following:

$$cs_j^z = \langle n_j, IN_j^z, OUT_j^z, C_j^z \rangle \tag{4}$$

where C_j^z contains the quality values v_{jk}^z for the k-th quality dimension and measured during the z-th execution of the j-th concrete service.

3) Quality Requirements: At the design time the process designer should associate each task t_i with the most suitable concrete service operation $cs_{j,o}$ considering the service capabilities and the requirements of all the process users. Formally, regarding the specification of the non-functional requirements, we can distinguish two classes of users: final users ($\mathcal{F}U$) and process owners ($\mathcal{P}D$).

The final users specify their requirements on the $\mathcal{D}Q$ and $\mathcal{P}Q$ dimensions associated with the whole process since they ignore the process decomposition and the availability of several concrete services per each process task. Process owner instead specifies the requirements for all

the $qd_k \in \mathcal{Q}D = \mathcal{D}Q \cup \mathcal{P}Q \cup \mathcal{P}rovQ$ associated with each task t_i. Anyway, both types of users operate a restriction on the admissible range of values but at different levels. In details, the $\mathcal{F}U$ requirements related to the quality dimension qd_k, associated with the whole business process, can be specified as follows:

$$FUReq_{uk} = R(Fu_u, qd_k) = \tag{5}$$

where $Fu_u \in FU$ and $FUReq_{uk} \in V$ represents the restriction on the range of admissible values. This restriction corresponds to the values required by the user for the given quality dimension.

The process designer $\mathcal{P}D$ defines quality requirements for each task t_i as described in the following:

$$Req_{ik} = R(PO, t_i, qd_k) \tag{6}$$

where $Req_{ik} \in V$ represents the restriction on the range of admissible values for the dimension qd_k related to the task t_i. Theoretically, the service execution should satisfy the quality requirements and v_{jk}^z is the actual value of the quality dimension in the z-th execution and should be also $v_{jk}^z \in Req_{ik}$.

IV. Strategies to improve process dependability

Dependability of a computing system is the ability to deliver service that can be trusted [1]. A service is correctly delivered when it provides the desired functionality without failures. A system failure is an event that occurs when the delivered service deviates from correct service [15]. At run time, the system may fail for situations that cause unexpected behaviour such as errors, changes, and inefficiencies. In fact, the process execution may be affected by predictable and unpredictable exceptions, which cannot be anticipated at design time or for which the additional costs for considering them at design time would not justify their implementation. In order to avoid the failure of the process, also for unanticipated exceptions, process designers should consider possible actions to support the failures detection and recovery.

Thus, a systematic description of the concepts of dependability consists of three parts: the threats to, the attributes of, and the means by which dependability is attained [15]. The threats include all the types of failures that can occur. The attributes refer to the dimensions that are related to the dependability concept as described in Section I. Finally, the means include all the techniques that can be used and combined to develop a dependable computing system. The

literature presents different dependability improvement strategies that at run-time are able to prevent failrues, react to inefficiencies by correcting errors, or providing alternative execution paths (e.g., [16] [17] [18]). The set of strategies to improve the process dependability $\mathcal{DIS}$ contains different mechanisms whose efficiency is strictly related to the execution context and to the process functional and non-functional requirements. Therefore, for each pair $< t_i, cs_{j,o} >$, a set of suitable improvement strategies $DIS_{ij} \subset \mathcal{DIS}$ is defined. Each element $dis_{ijr} \in DIS_{ij}$ identifies a possible dependability improvement strategy suitable to react at unexpected events or changes associated with the related task t_i.

It is possible to divide the set of strategies $\mathcal{DIS}$ in two disjunct subsets: $\mathcal{DIS}_p$ as the set of *preventive strategies* able to modify the process workflow in order to prevent failures and $\mathcal{DIS}_c$ *corrective strategies* able to change the process workflow in order to react at some occurred faults. The former are adopted in order to address a weakness in a system that is not yet responsible for causing nonconforming service. These mechanisms are very important and designers provide them in order to enable a system to react to some exceptional situations that is possible to predict. Each strategy can be associated with a task and invoked under some a-priory known conditions. Actually, such strategies may be considered just as a part of the whole workflow model: a preventive action is just a scope of activities that are invoked at a given time moment if some conditiona are fulfilled. The main disadvantage of the prevention is that it can not cover all cases and all exceptional situations that are related to the environment in which an activity is executed. In fact, the condition specifying when the preventive strategy should be executed depends only on the internal objects within the handled scope. It never depends on the environment variables within the whole workflow or on other scopes. In this work, we consider also the case in which the system automatically reacts to some expected and unexpected faults. To this aim, corrective strategies are needed to generate repair plans at run time or, alternatively, to suggest manually performed ad-hoc operations. These strategies must be able to react to changes by executing a set of simple actions in a given order bringing the system state back into a normal mode.

Preventive and corrective strategies could be also distinguished on the basis of their cause, i.e., the type of service change occurred: functional change and non-functional change. The class of strategies that are associated with functional changes includes all exceptional situations where the internal business-logic of the workflow activities is somehow corrupted and wrong and abnormal results are provided. The class of non-functional faults includes all faults that are

instead related to process performance and QoS aspects.

Finally, corrective and preventive strategies can be also distinguished on the basis of the level of their application and related effects: instance level, class level, and infrastructural level. Instance level strategies correct the single instance that fails at a certain time instant. Class level actions extend their impact on several instances of the same process. Furthermore, some faults are caused by the environment in which activities are performed. Workflow management systems and web services execution containers have their own parameters and requirements. When these constraints are violated, we say that the action is at an infrastructural level. Reasons of violations may be found in the workflow design: parameters of the activities depend often on the concrete workflow where they are used. In the following sections, we introduce details about the main corrective and preventive strategies.

A. Preventive dependability improvement strategies

Preventive dependability improvement strategies are used at design time in order to avoid that failures occur. They usually imply changes in the process workflow and include: Insertion of monitors, Exception handlers, Service redundancy, and QoS constraints. Figure 1 describes the preventive depensability strategies on the basis of the properties discussed in the previous section.

Strategy	Type of change	Impact Level
Functional monitors	Functional change	Instance Level
Exception handlers	Functional change	Instance Level
Service redundancy	Functional change	Class Level
QoS constraints monitors	Non-functional change	Instance, Class, and Infrastructure Level

Fig. 1. Preventive dependability improvement strategies

1) Insertion of functional monitors: The first preventive strategy that it is possible to use to design service-based processes is the insertion of functional monitors. We refer to active functional monitors that analyze messages exchanged between modules in order to detect anomalies and to trigger actions to prevent faults or recover them to avoid the failure of the whole processes. In this paper, we consider that the correctness of a process from the functional point of view can be assessed by considering the correctness of the data that are processed to provide the final

output to the final users. For this reason, the functional monitors considered by QUADS are also called *data quality monitors* [19]. They are inserted in the process flow to evaluate data quality dimensions associated with exchanged data. The conformity of these dimensions with respect to users expectations is checked. When data quality values are below specified thresholds, alarms are sent to the systems manager. Repair actions for data quality at design time require the identification of the causes of data errors and their permanent elimination through an observation of the whole process where data are involved.

Each data quality monitor relies on a model of the system itself to interpret a set of measurements, to detect the presence of faults and to identify the specific causes of the fault. It is necessary to model the task execution and the role of the manipulated data considering both the data flow and structure. In this scenario, for each task t_i, it is not sufficient to consider only input and output information flows, but it is also necessary to consider external data, which are data that are used by the task but do not derive from previous activities executed in the process.

According with this model an error in the output data can be consequence of:

- an error generated by the activities that precede the analyzed one
- an error generated by the analyzed activity. This type of error can be classified as self-generated error.

In this case the error detection and the correction can be performed using different methods:

- Data cleaning by manual identification: comparison between value stored in the database and value in the real world;
- Data bashing (or Multiple sources identification): comparison of the values stored in different databases;
- Cleaning using Data edits: automatic procedures that verify that the inserted data satisfies specific requirements.

In case a self-generated error occurs, the causes can also be related to the data structure or external processes. In fact, it is necessary to consider that the activity can be influenced not only by the previous activities in the process but also by other external processes that for example might use the same data sources.

2) Exception handlers: Exception handlers detect and correct faults occurring in a single instance. Exception handling mechanisms are provided by all service comparison and workflow environments. For instance, considering the de-facto standard for Web-Service Orchestration,

Ws-BPEL [20], it provides standard patterns for managing exceptions. In fact, at design time, specific handlers (fault, compensation, event, and termination handlers) can be associated with an activity included in a task t_i or with a scope, that is, a set of activities. The following basic handlers are provided:

- *Fault handler*: is used to explicitly catch errors and handle them by executing specified subroutines. It terminates all the activities contained within the scope in order to undo the partial and unsuccessful work of a scope.
- *Compensation handler*: while a business process is running, it might be necessary to undo one of the steps that have already been successfully completed. The specification of these undo steps are defined using compensation handlers that can be defined at the scope level. Each handler contains one activity which should be exwecuted when a scope needs to be compensated.
- *Event handler*: the whole process as well as each scope can be associated with a set of event handlers that are invoked concurrently if some specific event occurs. Event handlers correspond to any type of activity that can be triggered by events such as incoming messages and temporal alarms.
- *Termination handler*: forces the termination of a scope by disabling the scope's event handlers and terminating its primary activity and all running event handler instances.

The main disadvantage of inserting handlers in the process workflow is the lack of flexibility. More powerful and flexible instruments could be built, but this effort is currently fully in charge of the designer.

3) Service redundancy: In the process design, it is possible to insert redundant elements in order to reduce the probability of failure in the process execution and to increase the process availability. The adoption of this strategy modifies the process flow in order to assure the correctness of multiple instances of the same process. For this reason, this strategy is classifiable as a class level strategy. In [21] three patterns are proposed (see Figure 2). In all the three arrangements, actions are linked by an AND split followed by an "1 service out of n-services join": the join condition synchronizes only the first finishing service. For the other part of the parallel arrangement we consider three sub-structures: (a) the best alternative candidate is put into an redundant AND-split with an 1-out-of-n-join arrangement (b) the quickest of alternative candidates is synchronized by an AND-split followed by an 1-out-of-join (c) an alternative

candidate is synchronized by a XOR split followed by an XOR join. The first arrangement that suggests alternative candidates improves the execution time if the alternative candidate provides a quicker execution time than the original candidate. The actions are linked by an AND split followed by a 1-out-of-n join that means that from a parallel arrangement all n tasks are started, but at least one task is required to finish for the synchronization. The cost raises by the cost of the additionally executed task. The availability improves because every additionally invoked service raises the probability for the successful execution of the arrangement. The reputation can be reduced if the alternative service offers a lower reputation than the original one.

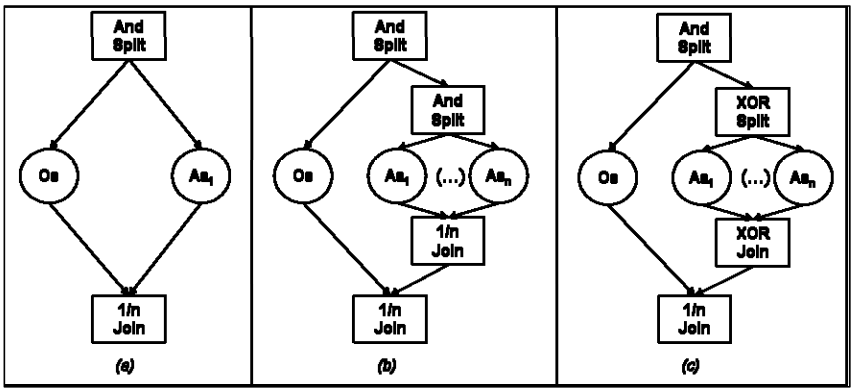


Fig. 2. Redundancy actions [21]

The second solution arranges the original service in a parallel structure containing the alternative candidates in a parallel AND-split with 1-out-of-n-join structure. Both joining elements will synchronize upon the first candidate ends. This arrangement reduces the execution time if one of the alternative candidates provides a quicker execution time than the original service. The cost raises by the sum of all additionally executed tasks. Like before, the reputation can be reduced if an additional candidate offers a lower reputation. And the availability improves because every additionally invoked service raises the probability for the successful execution of the arrangement. Finally, the third solution is different from the previous replacement structures: this structure invokes only one of the available alternative candidates. It is assumed that the probability of executing the individual candidates is equally partitioned. Thus, the execution time improves if the selected candidate executes quicker. For a high number of executions, the cost raises by the mean value of the individual costs. Again, the reputation may lower for this

arrangement if alternative candidates show a lower reputation than the original service.

4) QoS constraints monitors: In order to avoid failure, it is possible to define QoS constraints that have to be satisfied during the service execution. QoS constraints monitors are in charge to catch the constraints violation and trigger related recovery actions. Violations can be referred to local and global constraints. Local constraints define quality of Web services to be invoked for a given task in the process i.e., candidate Web services are selected according to a desired characteristic, e.g., the price of a single Web service invocation is lower than a given threshold. Global constraints specify requirements at process level, i.e., constraints posing restrictions over the whole service-based process execution can be introduced, e.g. the price of the composed service execution is lower than a fixed budget. Constraints may be specified on a set of N pre-defined quality dimensions. They are monitored and, if they cannot be satisfied, before that a service execution fails, suitable negotiation mechanisms are performed in order to determine new quality values for Web service invocations. If the negotiation succeeds, an agreement on the new price and quality parameters for a given operation invocation is achieved. If the negotiation fails, it is necessary to relax the quality constraints in order to identify the largest set of global constraints specified by the user which could be fulfilled. Subsequently, the quality parameters of the operation invocations which lead to constraints violation are negotiated.

B. Corrective dependability improvement strategies

Corrective dependability improvement strategies do not modify the process flow, but they are procedures that require additional components into the computing systems that should support the recovery procedure when a failure occurs. For example, they need the support of a service management infrastructure that includes the presence of a registry that collects the description of the services with their functional and non-functional properties. Corrective dependability improvement strategies can be also implemented as semi-automatic procedures. Figure 3 lists the strategies considered by the QUADS methodology and describes them in terms of the criteria discussed in Section IV. The following corrective dependability improvement strategies are considered:

- *Redo*: re-execute the service with possibly new values of input parameters.
- *Retry Web services invocation*: this recovery action is applied when faults point out a temporary unavailability of one or more services invoked during the process execution.

Strategy	Type of change	Impact Level
Redo/retry service invocation	Functional change	Instance Level
Service substitution	Functional change	Instance Level
Architectural reconfiguration	Non-functional change	Class and Infrastructure Level

Fig. 3. Corrective dependability improvement strategies

In this case, the process should be suspended and the invocation of the unavailable services should be repeated until they return available. Note that this strategy is applicable for example in case Web service wraps a human activity. This action differs from Redo activity, because a) it uses the same input parameters of the failed execution the activity without correcting/changing/adjusting them b) it can be repeated several times, provided that there are no constraints on the execution time.

- *Service substitution*: a more complex situation is the case where one or more services are considered as definitely unavailable and, in order to complete the process execution, it is necessary to substitute each failed service. The service substitution strategy allows us to change the service to invoke by finding a service that provides the same operations. Service matching can be performed by using semantic descriptions of the links between operations, e.g., operation o1 at partner A is equivalent to operation o5 at partner B, which means that they have the same sets of outputs, and the same or similar set of inputs. Having such a library of ”similar” operations we can apply the substitute action.
- *Architectural reconfiguration*: this type of corrective strategy is useful for the particular subset of QoS violation faults that derive from a lack of hardware or software resources on the service provider side. In this situation, reallocating and executing the service on different machines or application servers can solve the problem. Reallocation is possible only if Web services are provided with an ad-hoc management interface and the recovery manager has free access to all the resources (e.g., the recovery manager can determine the load balancing or the application priority in the operating system). Reallocation may be performed as reactive actions, when QoS violations are detected, but also as proactive actions, when optimization of service execution plans is performed using predictive techniques on future states of the execution environment. As discussed for substitution, the mechanisms are

the same as those used for infrastructure repair, where additional constraints have to be considered when the service is executed within a process.

V. A METHODOLOGY TO IMPROVE PROCESS DEPENDABILITY

According with the model introduced in Section III, the dependability of a process could be improved by enriching the process with monitoring features, preventive and corrective strategies in order to avoid or repair failures before that users perceive them. The literature presents a good number of approaches that propose suitable strategies to improve the dependability of service-based processes. Anyway, most of these contributions present methods to hard-code in the functional logic and that are related to a limited number of actions that are triggered only when some specific and known events happen. Clearly, these approaches are based on a local perspective (i.e., service perspective) and are not able to cover all needs that may arise. In fact, in some cases these needs are unknown and cannot be foreseen once for all. As discussed in the previous sections, our approach is based on a global perspective in which the whole process related to the service-based application is considered. From this perspective, the process designer has a complete knowledge of the logic of all the tasks t_i included in the process and of the control structures that link them and s/he is able to consider all the actions that can be programmed at design/implementation time and be associated with triggering events whenever possible, either before the execution or during the execution itself. However, it is possible to use alternative strategies in order to prevent or react to a failure. The identification of the most suitable strategy is not a trivial task since each strategy is characterized by its complexity and its functional and non functional properties. On the basis of the process context and goals, a repair strategy can be can more suitable than another. Thus, it is necessary to adopt a systematic approach to support the strategy selection. By using the approach proposed in this paper, the designer should be able to define the most suitable corrective and/or preventive strategy to apply and consequently modify the process workflow or enrich the computing systems with all the required components and metadata.

In this paper we propose a methodology composed of four phases:

1) **Process analysis**: in this phase processes are analyzed in order to identify the actors involved, their quality requirements and the tasks that should be associated with services at run time;

2) **Dependability Improvement Strategies analysis**: the strategies are analyzed in order to evaluate their impact on the process correctness;
3) **Ranking of the Dependability Improvement strategies**: the results of the process and dependability improvement strategies analysis are combined in order to define, for each process task, the ranking of the suitable strategies to adopt in the process design;
4) **Selection of the final set of strategies**: the process designer analyzes, from a global perspective, the top-k strategies defined for each task in order to identify the combination that better satisfies the quality requirements defined for the whole process.

As represented in Figure 4, the methodology supports the process designer in the service-based applications design. At the end of the design phase, the process designer issues the process workflow that includes the components or constructs required for the execution of the dependability improvement strategies. At run time, the tasks t_i that compose the process are bound with the concrete services published in the available service registry. During the process execution, dependability should be guaranteed by a *Process Monitor* able to detect unexpected behaviours such as changes, errors, and failures or inefficiencies. All the events that worsen the process dependability are communicated to the *Dependability Rule Manager* that is responsible for the automatic activation of the corrective strategies (e.g., service substitution) or for informing the process designer that a process redesign might be needed. The efficiencies and inefficiencies of the monitoring phase are also transmitted to the *Reputation Assessment Module* that respectively will increase or decrease the reputation of the executed service. In summary, the selection of the most suitable dependable improvement strategies in the design phase supported by the methodology proposed in this paper and the continuous monitoring in the run time phase are the basis for an environment able to guarantee a good level of process dependability. In the next sections the methodology will be explained by means of a running example and of the description of the software tool that we have implemented.

A. Running example

Let us considered as example a reference scenario taken from the WS-Diamond Project[1] that concerns a company that sells and delivers food using Web service technology. The company has

[1] wsdiamond.di.unito.it

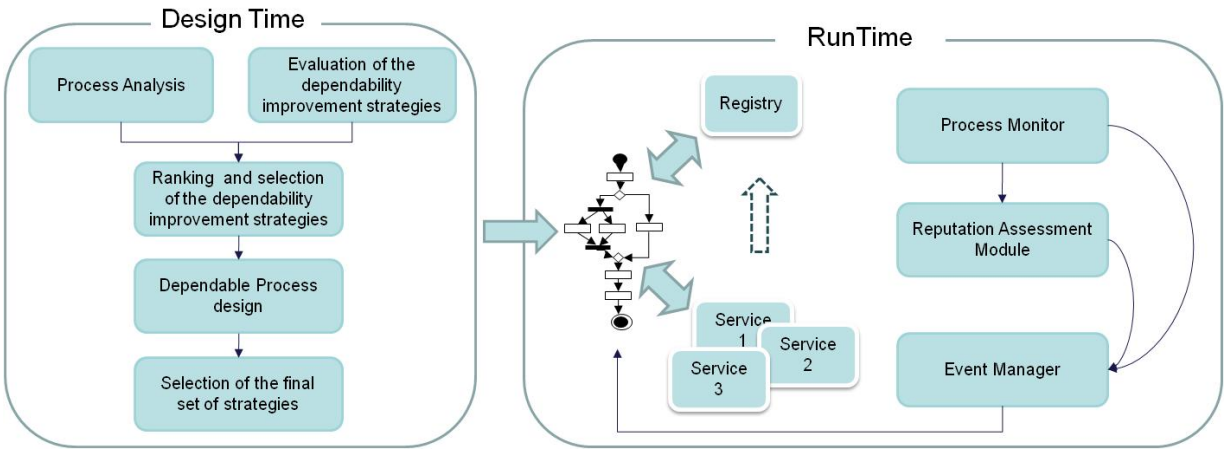


Fig. 4. Architecture that enables the dependability improvement

an online Shop that customers use to select and order food. The Shop does not have a physical counterpart, as it stores and delivers food using either Warehouses or Suppliers. Warehouses are responsible for stocking unperishable goods and physically delivering items to customers. In case of perishable items, that cannot be stocked, or of out-of-stock items, the company interacts with a Supplier. The business process, executed by means of several services, covers all the order management activities from the order receipt to the parcel delivery.

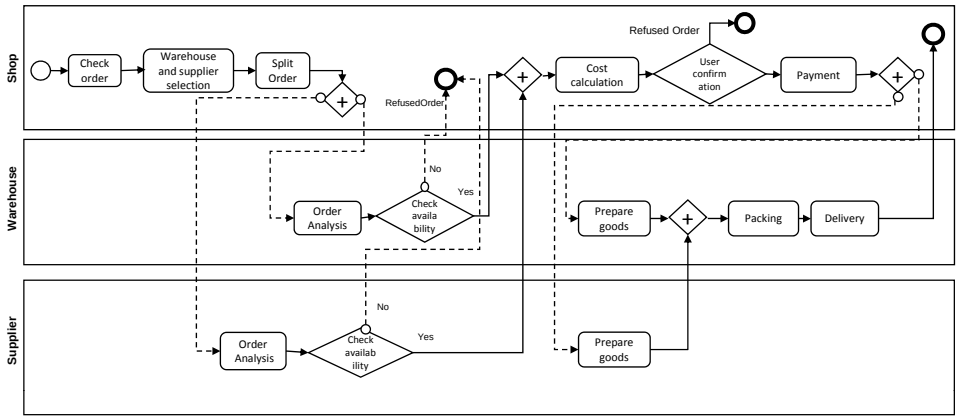


Fig. 5. Food Shop workflow

Figure 5 represents the process workflow. When a Customer places an order, the Shop first selects the Warehouse that is closest to the customers address, and that will thus responsible of

the goods delivery. Ordered items are split into two categories: perishable (cannot be stocked, so the warehouse cannot possibly have them in stock) and unperishable (the warehouse might have them). The first step is to check whether the ordered items are available, either in the warehouse or from the suppliers. If they are, they are temporarily reserved in order to avoid conflicts between several orders. Once the Shop receives all the answers on availability, it can decide whether to give up with the order (if there is at least one unavailable item) or to proceed. In the former case, all item reservations are canceled and the process ends. If the order goes on, the Shop computes the total cost related to items and with the aid of the Warehouse defines the shipping costs. Then the Shop sends the bill to the Customer, that can decide whether to confirm the order or not. If the Customer does not confirm the order, all item reservations are canceled and the activities flow ends here. If the Customer confirms the order, all item reservations are confirmed and all the Suppliers (in case of perishable or out-of-stock items) are asked to send the goods to the Warehouse. The Warehouse will then assemble a package and send it to the Customer.

B. Process analysis

As described in Section III, service-oriented processes are executed by means of services that are considered units of work provided by external providers or internally developed. The first step of the process analysis aims at identifying the possible process components, that is the composite tasks that can be associated with web services operation. The process componentization can be performed by consider the several guidelines that the literature proposes to support the component (i.e., service) identification. Most of the contributions base their analysis on the task independency that can be evaluated on the basis of cohesion and loosely coupling principles [22]. In our work, we perform this activity by using the P2S methodology [23] that provides a comprehensive framework to identify components from the workflow.

The identification of process components defines the set of tasks $\mathcal{T}$ that compose the process and that support both the binding activities and the selection of the most suitable dependability improvement strategy. Each task t_i at run time has to be bound with one operation of the concrete service cs_j in order to define the execution plan $\mathcal{E}P$ composed of the pairs $< t_i, cs_{j,o} >$.

Considering the example described in Section V-A, it is possible to identify five abstract services to link to suitable service operations (see Figure 6). Anyway, in the binding phase,

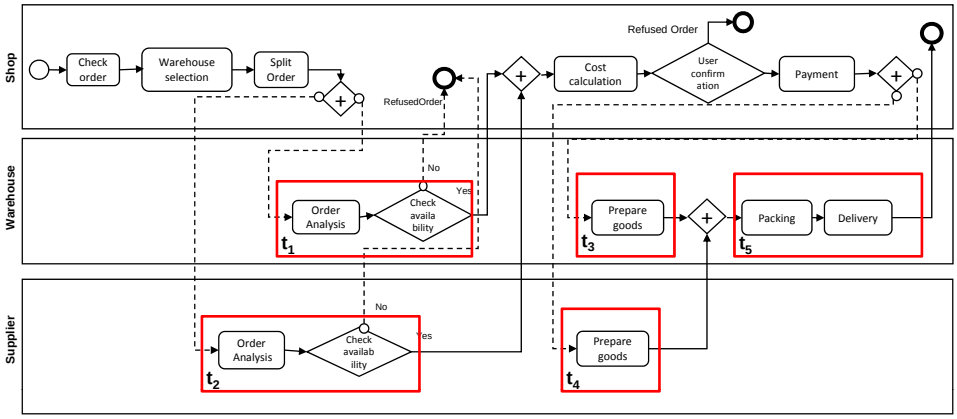


Fig. 6. Services invoked in the foodshop workflow

once that the warehouse or the supplier has been selected for the analysis of the order, the same provider must be selected for the preparation of the goods. This means that formally, considering the pairs $< t_1, c_{j_1,o_1} >$ and $< t_3, c_{j_2,o_2} >$, the service selection should consider the following constraints $j_1 = j_2$ and $o_1 <> 0_2$. Since the same rules are valid for the task t_2 and t_4 the identified services are: (a) *Service 1* is the *Warehouse service* that is in charge to check the order and prepare the non-perishable goods if they are available; (b) *Service 2* is the *Supplier service* that is in charge to check the order and prepare the perishable goods if they are available; (c) *Service 3* that receives the goods and is responsible for their packing and delivery.

Note that the binding phase, i.e., the definition of the concrete services, is performed by considering functional and non-functional properties. In cases in which more than one provider offer similar services the non-functional properties may drive the selection phase. In fact, the services characterized by a quality level able to satisfy users' requirements will be preferred. As stated in Section III-B, the service owner expresses quality requirements for each task while the final users define their preferences along the whole process. It means the binding functions should define all the pairs $< t_i, cs_{j,o} >$ able to satisfy both local and global constraints. For the satisfaction of the local constraints, it is sufficient that at design time for each specific concrete service cs_j, the capability value of each quality dimension Off_{jk} satisfies the requirements Req_{ik}. If the same service cs_j is associated with different tasks of the process, quality capabilities should satisfies the quality requirements of each task. As regards the satisfaction of the global constraints $FUReq_{uk}$, it is necessary to aggregate the requirements expressed by all the users in order to

have a unique requirement for each quality dimension $FUReq_k$.

The aggregation of the requirements can be obtained by using one of the classical aggregation functions, i.e., average, minimum or maximum. Also the capabilities associated by a quality dimension value Off_{jk} should be aggregated along all the concrete services. As described in [4] for each dimension, a specific aggregation pattern could be applied (see Figure 7). Global availability is given by the product of availabilities provided by executed services. All the data quality dimensions can be instead aggregated by using the minimum function. The addition is suitable for aggregating execution time and thus calculating the end-to-end process execution time. The reputation of the composed service is calculated as the average reputation of selected services. Thus, reputation of the process owner could decrease if the process fails for an internal fault or another provider's fault. Finally, fidelity is the unique dimension for which the definition of a global measure has no meaning since it is a dimension that condition the selection of each concrete services but it has a null impact on the quality of the whole process.

Quality Dimension Class	Quality Dimension	Aggregation function
Data Quality	Accuracy	$Acc(BP) = Min_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Acc_j$
	Completeness	$Comp(BP) = Min_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Comp_j$
	Timeliness	$Time(BP) = Min_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Time_j$
Process Quality	Availability	$Av(BP) = \prod_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Av_j$
	Execution Time	$Et(BP) = \sum_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Et_j$
Provider Quality	Reputation	$Rep(BP) = \frac{1}{\lvert EP\rvert}\sum_{(t_i,cs_{j,o})\in\mathcal{E}P}\, Rep_j$

Fig. 7. QoS aggregation function

In the evaluation of the requirements satisfaction, it is necessary to consider that the quality dimensions might not be considered equally important. In the process execution a quality dimension can be considered more critical than another one and consequently can be associated with a different degree of relevance. Therefore there could be a distinction between strong and weak constraints. The search for the best combination of concrete services to use for the execution of a business process can be modeled as an optimization problem. If no solution respecting all constraints exists, the execution plan can be modified or a negotiation phase between the process owner and the providers of the concrete services included in the current execution plan is needed for the definition of a Service Level Agreement.

The definition of the composite task t_i and the definition of the possible concrete services provide useful information for the selection of the most suitable dependability improvement strategy. In order to enrich the context description it should be also considered that the users and the process owner may have a different perception about the relevance of the various quality criteria. In order to model these aspects, we introduce an element w_{ku} that represents the weight that identifies how much the k-th quality dimension qd_k influences the overall perception of the quality of the service from the u-th user perspective. It is worth noting that the weight assignment activity is a crucial point of the method. The users should be able to define the importance of a quality dimension in the service provisioning. The simplest way to collect this information could be to let users associate with each quality dimension a value between 0 and 1 to express the importance that the dimension has for the specific user class. In this case the only constraint is that the sum of the weights associated with all the dimensions has to be equal to 1. This method is difficult to apply, since the absolute relevance of a dimension on the total quality is hardly identifiable. For this reason, we assume that a suitable method for the weight assignment could be the Analytic Hierarchy Process (AHP) approach, a decision making technique developed by T.L. Saaty [24]. This is a qualitative approach in which the users decompose their decision problem into a hierarchy of more easily comprehended sub-problems, each of which can be analyzed independently. In fact, following this method the users analyze the quality dimensions in pairs and only state if a dimension is more influent than another one on the overall quality. We assume to consider only quality dimensions that are independent. AHP is a decision-making technique that assigns to each dimension a score w_{ku} that represents the overall relevance of qd_k dimension for the u-th user.

Figure 8 show the interface of the tool that we have developed in which the process described in Section V-A through which the users can define their preferences. The quality dimensions are compared in pairs and for each pair, users can indicate the importance of one dimension with respect to the other one simply positioning the sliding bars.

At the end, in general, the evaluation of the total importance of the k-th quality dimension (I_k) is calculated by using a weighted average of the weights defined by the different users. Users and process owner defines the importance of the data quality dimension with different granularities. In fact, the process owner can specify different preferences for each task while the other users are limited to the whole process. In order to have quality requirements for each task,

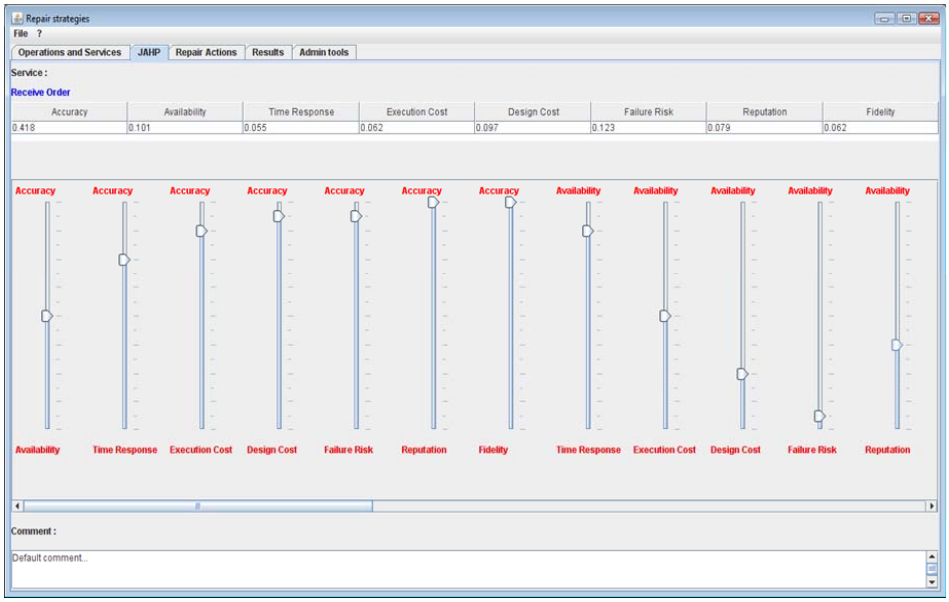


Fig. 8. AHP interface

we consider that the preferences expressed by the users on the entire process can be considered valid for all the tasks that compose the process. In this way theimportance of the k-th dimension quality is calculated for each task t_i as follows:

$$I_{ik} = \frac{\sum_{u=1}^{U} w_{iku} \cdot imp_u}{U}$$

In the formula, users feedbacks regarding the relevance of the dimension qd_k are weighted based on the role (and hence) importance of the users in the process (imp$_u$). Users can be more or less relevant for the process designer considering their fidelity and the associated revenues.

For each task, the ranking of the total scores identifies the set of relevant quality dimensions that have to be considered to evaluate the repair strategy to apply when a fault occurs. In fact, the process analysis defines the list of service quality dimensions that have to be assessed, analyzed and improved along the specified requirements.

Let us consider the Supplier Service of the example describe in Section V-A. The order reception and check availability and goods preparation are operations that have to be performed by the same provider. We can focus on the different perspective that different types of users may have on the same service. Considering that the main process actors are the process users and

the process owner we may consider to model a context in which the process users requirements are more stronger than the process owner requirements and a context in which the situation is opposite: the process owner requirements are the most relevant ones. For instance, when considering the process owner view, accuracy is much less important than in the users' view. The two contexts can be characterized by the weights presented in Figure 9.

	Accuracy	**Availability**	**Timeliness**	**Execution Time**	**Completeness**	**Reputation**	**Fidelity**
Process Owner View	0,08	0,39	0,16	0,13	0,085	0,099	0,1
Users View	0,2	0,2	0,25	0,074	0,115	0,078	0,08

Fig. 9. Quality dimensions relevance weights

At the end of the Process Analysis phase, we have weights assigned to each task in the process as described in Figure 10 that represents the interface through which designers can select a single activity (e.g., received order) or the entire process and to visualize all the information about it.

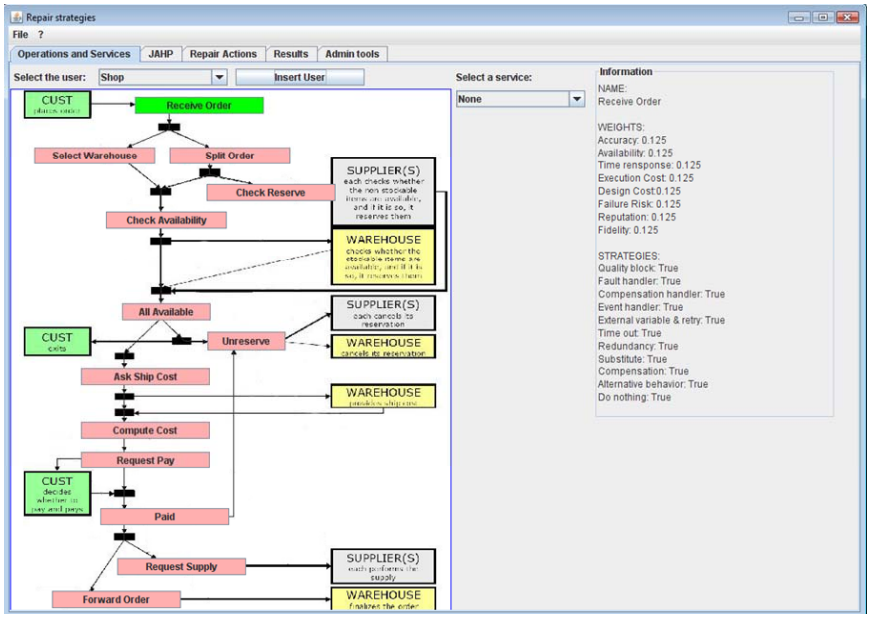


Fig. 10. Process visualization and interaction

C. Dependability improvement strategies analysis

The comparison across the different dependability improvement strategies can be very difficult due to the heterogeneity of their functionalities. For this reason, we base our analysis on the impact of each strategy on the process execution. In particular, the influence of each strategy dis_r on the quality dimensions qd_k is considered. In general, such influence is positive on some quality dimensions, and negative on other ones. However, there might be cases in which a general trend cannot be defined in principle. On the basis of our experience and some empirical studies we defined for each strategy the influence on each of the quality dimension (see Figure 11).

	Accuracy	Completeness	Availability	Timeliness	Execution Time	Reputation	Fidelity
Functional Monitors	+	+	= or -	-	-	+	=
Exception Handlers	+	+	+	-	-	+	=
Service Redundancy	+	+	+	-	-	=	=
QoS constraints Monitors	+	+	+	-	-	+	=
Redo/retry service invocation	+	+	+	-	-	=	=
Service substitution	+ or -	+ or -	+	-	-	+ or -	-
Architectural Reconfiguration	=	=	+	=	-	+	=

Fig. 11. Impact of the repair actions on the quality dimensions

In the Table 11, symbol ”+” means that the quality dimension trend is improved by the adoption of the strategy, while ”-” means that the strategy has a negative influence on the quality dimension. For example, the insertion of the quality block increases, and then worsens the execution time while improve the data accuracy since errors can be corrected. The symbol ”=” means that repair strategy have null influence on the quality dimension. The values inserted in Figure 11 are general trends that are almost valid for each context and process. This information supports the process designer in the selection of the dependability improvement strategies to apply. In order to allow a systematic comparison of the different strategies, it is necessary to

define a numerical value Imp_{rk} to evaluate the degree of the influence that the strategy dis_r has on the quality dimension qd_k. A first approach is to ask each process designer to specify a value of the different impacts along a specific process in order to obtain a precise value Imp_{rk}. However, since it would be difficult for a designer to associate an exact value with each strategy-dimension pair, we propose to adopt fuzzy numbers for such evaluations. Fuzzy numbers provide the logical-mathematical instrumentation that combines the capabilities of the natural language with the advantages of algebraic formalization and numerical representation [25]. In this way, the weights associated with the dimensions is established by the process designer that express their opinion by using natural language expressions, e.g., high negatively influence, medium negative influence, irrelevant influence, low positive influence. It is possible to use the conversion scales similar to the one introduced in [26] to convert the linguistic expressions in fuzzy numbers. Note that the conversion scales also consider the risk aversion and propensity of the users. Figure 12 shows the interface that the process designer can use the fuzzy number to evaluate the different strategies.

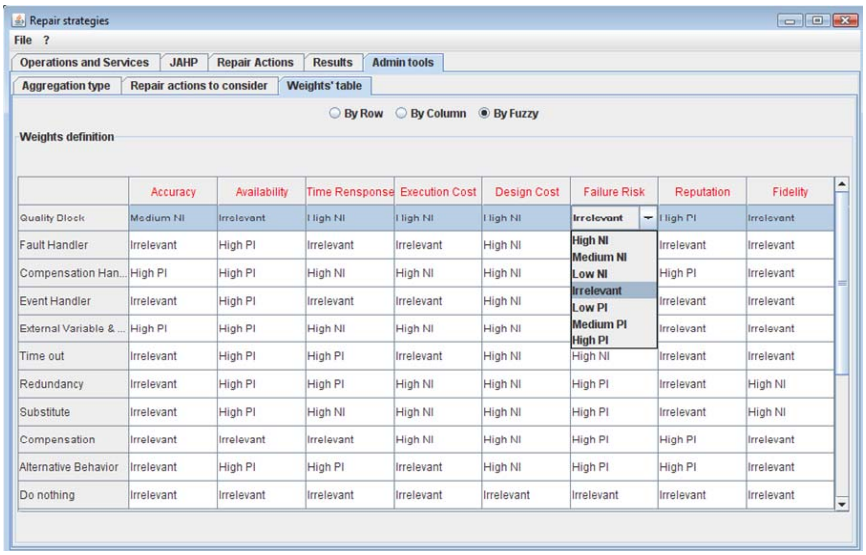


Fig. 12. Dependability improvement strategies evaluation interface

D. Ranking of the dependability improvement strategies

Process and *Dependability Improvement Strategies* analysis provide all the elements for the definition of the most suitable repair action along a specific service. In fact, the process analysis phase provides the measures I_{jk} that is the evaluation of the quality dimensions by considering their relevance for the users. The evaluation of the improvement strategies performed by the process designer provides the values Imp_{rk} that is the assessment of the impact of the strategy on the quality dimensions. By considering these two set of data, it is possible to define the appropriateness A_{ir} of the r-th strategy for the i-th task as follows:

$$A_{ir} = \sum_{k=1}^{K} (I_{ik} * Imp_{rk})$$

Considering the example and combining the process analysis with the dependability improvement strategies we obtain the ranking for the most suitable strategy to use in both the *Process owner* and *Users* contexts. The Process Owner view is characterized by weights that assign more relevance to the architectural part of the application respect to the dimensions more related with the functional characteristics while the Users view favours the accuracy, timeliness, and availability dimensions.

Process Owner View	**Users View**
Exception handler	Service redundancy
Redo/retry service invocation	Architectural reconfiguration
Service redundancy	Redo/retry service invocation

Fig. 13. Ranking of the dependability improvement strategies for the FoodShop example

Figure 13 shows the ranking of the strategies that are considered as suitable in the FoodShop example. The strategies proposed for the Process Owner View are the ones that both assure availability and slightly worsen the execution time and design effort while the Users View prefers the correctness and the availability of the service are the most relevant features. In fact the Users View suggests the strategies that assure a correct service provision but that are more costly and complex to realize.

E. Evaluation of the efficacy of the methodology

As described in the previous sections, the QUADS methodology supports the process designer at design time by providing guidelines about the most suitable dependability improvement strategy for each task t_i. Once that the process designer views the list of the suggested strategies, s/he has to analyze, for each service that is involved in the process, the first two or three options in the ranking and define the strategy that is effectively suitable for the analyzed process. It is not sufficient to select and adopt only the first ranked strategy since there might be some local or global incompatibilities. Local incompatibilities are related to the selection of a strategy for a single service and can occur if the dependability improvement strategy is not applicable for technical reason or for the violation of quality constraints. Global incompatibilities are instead related to the analysis of the whole process and they occur when the strategies associated with the different tasks are acceptable from a local point of view but their combination violate global quality constraints. In this case, new strategy selections might be necessary.

In fact, the application of a r-th strategy associated with the i-th task will affect, at run time, the evaluation of the values associated with the different quality dimensions. Considering the pairs $< t_i, cs_{j,o} >$ and that the final quality evaluation depends on the selected cs_j, it is possible to state that the adoption of a strategy dis_r impacts on the values of the different quality dimensions by increasing or decreasing their value of a quantity Δv_{jkr}. The analysis of the process designer for detecting local incompatibilities should check that the new capability values of each quality dimension $Off_{jkr} = Off_{jk} \pm \Delta v_{jkr}$ still satisfies the requirements Req_{ik}. Once that for each task, the strategy has been selected, it is possible to define the Dependability Improvement Plan $\mathcal{DIP}$ as the set of pairs $< t_i, dis_r >$.

In order to define the final plan, $\mathcal{DIP}$ should be further analyzed for the identification of possible global incompatibilities. In fact, even if all the identified strategies satisfy the local constraints, their union might not be satisfy the global constraints. It means that the aggregation of the new capability values Off_{jkr} does not satisfy the $FUReq_{uk}$. The aggregation of the different capability values depends on the defined process workflow $\mathcal{PW}$. The evaluation of the process workflow at design time consider the set of tasks that compose it $(t_1 ... t_n)$ and the control logic that links all the tasks assigning a probability of execution $freq_i$ associated with every task. Details on the formulas that can be used to aggregate quality dimensions in different

process pattern can be found in [4].

As regards the local incompatibilities, let us consider the case in which the Service 2, that is the Supplier service, is analyzed. Figure 15 describes the local constraints and the characteristics of the selected Service α before the adoption of the dependability improvement strategy.

	Accuracy	**Availability**	**Timeliness**	**Execution Time**	**Completeness**	**Reputation**	**Fidelity**
Local Constraints	$\geq 0,80$	$\geq 0,95$	$\geq 0,90$	$\leq 12h$	$\geq 0,90$	$\geq 0,70$	$\geq 0,70$
Service 2α	0,9	0,95	0,9	$\leq 6h$	0,95	0,8	0,75

Fig. 14. Local contraints and service quality dimensions

On the basis of the Provider View (see Figure 9), the process designer should use the following repair actions: redundancy, architectural reconfiguration, and redo/retry. The architectural reconfiguration is, in this case, an action that is not applicable for technological reasons. In fact, the invoked service is mostly composed of manual operations and architectural actions would not have any effects. Redundancy can be instead suitable in critical situations in which time constraints are tight. In this case, the order is sent to different suppliers and the shop collects the goods from the supplier that makes them available in the shortest time. The redundancy implies the selection of, at least, another new service β. This selection should be driven by the constraints desctibed in Figure in order to have the guarantee that local constraints are still satisfied. The constraints are calculated by considering that the services are executed in parallel and applying the formulas for the aggregation of quality dimensions defined in [4]. The only contraints that

	Accuracy	**Availability**	**Timeliness**	**Execution Time**	**Completeness**	**Reputation**	**Fidelity**
Redundant Constraints	$\geq 0,80$	≥ 0	$\geq 0,90$	$\leq 12h$	$\geq 0,90$	$\geq 0,70$	-

Fig. 15. Contraints for the selection of the redundant service β

change concern availability and fidelity dimensions. The former has been calculated on the basis that the unavailability of the service occurs only if both services are unavailable. Since the Service α satisfies the local constraints, it is sufficient that the service β is available with a probability greater than zero. The latter has no influence on the choice of the service since fidelity is relevant only for the preferred service (i.e., Service $alpha$).

The process designer should anyway consider that this approach has a negative impact on the service price and on the fidelity. The increase of the service price is due to the increase of the execution cost casued by the fees paid for the parallel execution of the services offered by multiple suppliers. In fact, once that one of the suppliers finishes to prepare the goods, the other orders have to be cancelled and penalties have to be paid. The cancellation of the order also might affect the fidelity dimensions since if cancellation recurs with high frequency, it could worsen the relationships between the Supplier and the Food Shop. Note that redundancy might also not be appropriate for technical reasons since its applicability starts from the assumption that similar services are available. Redundancy cannot be adopted if the registry does not contain equivalent services to invoke.

The redo/retry service invocation strategy is not limited by technical constraints and it only affects the execution time delaying the goods delivery in case of failures (see Figure 11). In this scenario, this action is the most preferrable one, since counting the re-execution of the Service 2α only once, the local contraints are still satisfied: in the worst case the execution time is $\leq< 12h$.

On the basis of the Users View the suggested use of exception handlers is also extremely feasible. It just requires the modification of the task worflow in order to include all the exceptions needed to handle with all the failures that can be predicted. This strategy also affects the execution time but the delay in the execution time is lower than the one cauaed by the redo/retry action.

As regards the global incompatibilities, $\mathcal{D}IP$ has to be analyzed in order to define if the combination of the selected stategies for the composite services satisfies the global constraints $FUReq'_{uk}$. Let us consider the three services identified in Figure 6 will be executed by means of corresponding concrete services characterized by the capabilities Off_{jk} described in Figure 16. Figure 16 also contains the global contraints associated with the service-based process.

	Accuracy	**Availability**	**Timeliness**	**Execution Time**	**Completeness**	**Reputation**	**Fidelity**
Global Constraints	$\geq 0,90$	$\geq 0,90$	$\geq 0,90$	$\leq 24h$	$\geq 0,90$	$\geq 0,70$	$\geq 0,70$
Service 1δ	0,95	0,99	0,9	$\leq 6h$	0,95	0,7	0,8
Service 2α	0,9	0,95	0,9	$\leq 6h$	0,95	0,8	0,75
Service 3δ	0,95	0,99	0,95	$\leq 8h$	0,95	0,7	0,8

Fig. 16. Global contraints and service quality dimensions

Data in Figure 16 suggest that it is not possible to adopt the redo/retry service invocation strategy for both Service 2α (or Service 1δ) and Service 3δ since constraints about the execution time could be violated. In fact, if both services fail, the sequentiality between the different services increases the delay in goods delivery. The delay can be descreased by combining the exception handler action with the redo/retry service invocation and thus by guaranteeing more flexibility with respect to a process in which all the faults are managed by exception handlers.

VI. Related work

The concept of dependable computing has been largely investigated in different contexts and under different perspective [27]. The concept of dependable computing appears in the early 60's when some authors developed theories for using redundancies to mask failures (e.g., [28]). These represent the first contributions in the failure tolerance research area. In [29], [30], these studies were extended and the literature present several approaches for the inclusion of techniques of error detection, fault diagnosis and recovery. From there, a lot of techniques for the prevention, removal and forecast of failures have been proposed (e.g. [31], [32]). In the last years, the dependability concept has been considered as the basis for the development of niche research areas such as self-healing systems and adaptive systems. The former include systems that have some reflective capabilities [33]. In fact, the system should be designed in order to perform introspection. That is, the system has to be able to monitor the running state of the system to identify any anomalous behaviour during its operation. Whenever any failure is detected, some intercession actions have to be performed, i.e., carry out a certain procedure to recover from the failure and repair faults without interrupting any of the functional services it provides, so that it returns back to a stable state. The latter assume that changes related to the internal state of the system are not the only cause that determines the need for adaptation. Another cause has to do with changes in the context in which applications are executed. For instance, this context may include the information about the users of the applications or. Users, in fact, may have special needs in terms of the component services to be used.

Furthermore, with the diffusion of service-based application, the literature presents a good number of approaches that deal with self-adaptation of service-based applications. Most of these try to predict the list of failures that can occur in the system and address this issue by managing the exception changing the software code in a static way (e.g., [34]). Doing this, a limited number

of adaptation strategies that are triggered only when some specific and known events happen. Some authors focus on triggering repair strategies as a consequence of a requirement violation [35], of the need for optimizing QoS of the service-based application [36] or for satisfying some functional requirements such as application constraints. The goal of the repair strategies usually proposed range from service selection to rebinding and application reconfiguration. We argue that this approach does not necessarily cover all needs that may arise. In some cases these needs are unknown and cannot be foreseen once for all. Furthermore, all aforementioned approaches show interesting features, but even those that enable the definition of various repair strategies lack a coherent design approach to support designers in this complex task. The approach that we advocate is based on the idea that dependability improvement strategies can be selected and programmed at design/implementation time on the basis of the context in which applications are used and of the users' needs. The design for process dependability has been analyzed in some contributions such as [37] and [38]. However, even in these cases the emphasis is on the mechanisms offered to design strategies and to trigger them, more than on a design for adaptation approach that supports developers in the usage of the available mechanisms. The methodology we propose in Section V can be considered as a contribution in this direction. To support the design of self-healing system, the quality of the self-healing strategies were analyzed in [39]. Authors analyze the functional aspects of the strategies and support the process designer by providing information about the software quality of the different strategies. The impact of the different strategies on the application is not considered. In the design phase it is important to verify the functional suitability of the improvement strategies but it is also important to evaluate their impact on the functional and non-functional characteristics of the composite process as suggested in Section V-C.

VII. Conclusions and future work

In this paper a methodology to support the process designer in the definition of the most suitable dependability improvement strategy is proposed. Currently, many contributions in the literature about service-based applications focus on dependability issues and propose several dependability improvement actions. Anyway, there is the lack of a coherent design approach to support designers in the evaluation and the selection of the improvement actions. As we think that dependability works properly only in the case the application is designed to be dependable, we

focus, in particular, on the identification of a number of design principles and guidelines that can drive the selection of the dependability improvement actions to adopt. The effectiveness of such principles and guidelines to build an acceptable ranking of the different actions is analyzed with reference to a running example that is the simplification of a real-world process. Future work will focus on the extension of the algorithm able to define automatically different dependability improvement strategies that are acceptable on the basis of the global constraints. Furthermore, other real-word scenarios will be considered in order to test the effectiveness of the proposed approach. Finally, The methodology can be also extended by considering issues about process adaptability in general and providing also guidelines for the design of adaptable process. In this case quality constraints should be analyzed together with context characteristics and the set of techniques that can be included in the process at design time ahould be redefined.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community 7th Framework Programme under the Network of Excellence S-Cube Grant Agreement no. 215483 and from the Italian MIUR-FIRB TEKNE Project.

REFERENCES

[1] A. A. Ucla, A. Avizienis, J. claude Laprie, and B. Randell, "Fundamental concepts of dependability," 2001.

[2] D. Dyer, "Unification of reliabiltiy/availability/repairability models for markov systems," *IEEE Transactions on Reliability*, vol. 38, no. 2, 1989.

[3] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani, "Paws: A framework for executing adaptive web-service processes," *IEEE Software*, vol. 24, no. 6, pp. 39–46, 2007.

[4] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transaction on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[5] S. H. Chang and S. D. Kim, "A service-oriented analysis and design approach to dveleoping adaptable services," in *Proceedings of the International Conference on Service Computing*, 2007.

[6] M. Ouzzani and A. Bouguettaya, "Efficient access to web services," *IEEE Internet Computing*, vol. 8, no. 2, pp. 34–44, 2004.

[7] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[8] Y. Wand and R. Y. Wang, "Anchoring data quality dimensions in ontological foundations," *Commun. ACM*, vol. 39, no. 11, pp. 86–95, 1996.

[9] T. Redman, *Data Quality for the Information Age*. Artech House, 1996.

[10] R. Wang and D. Strong, "Beyond Accuracy: What Data Quality Means to Data Consumers," *Journal of Management Information Systems*, vol. 12, no. 4, 1996.

[11] D. Ballou, R. Wang, H. Pazer, and G. Tayi, "Modeling Information Manufacturing Systems to Determine Information Product Quality," *Management Science*, vol. 44, no. 4, 1998.

[12] H. Pichler, M. Wenger, and J. Eder, "Composing time-aware web service orchestrations," in *CAiSE*, 2009, pp. 349–363.

[13] Y. Wang and J. Vassileva, "A review on trust and reputation for web service selection," in *ICDCS Workshops*, 2007, p. 25.

[14] D. Bianculli, W. Binder, M. L. Drago, and C. Ghezzi, "Reman: A pro-active reputation management infrastructure for composite web services," in *ICSE*, 2009, pp. 623–626.

[15] A. Avižienis, J.-C. Laprie, and B. Randell, "Fundamental concpets of dependability," in *Information Survivability Workshops (ISW)*, 2000.

[16] W. diamond Team, *WS-DIAMOND: Web Services-DIAgnosability, MONitoring and Diagnosis*. MIT press, 2009, pp. 213–239.

[17] M. Fugini and E. Mussi, "Recovery of faulty web applications through service discovery," in *International Workshop on Semantic Matchmaking and Resource Retrieval: Issues and Perspectives (SMR 2006)*, 2006.

[18] B. Pernici and A. M. Rosati, "Automatic learning of repair strategies for web services," in *ECOWS*, 2007, pp. 119–128.

[19] C. Cappiello and B. Pernici, "Quality-aware design of repairable processes," in *International Conference on Information Quality (ICIQ 2008)*, 2008, pp. 382–296.

[20] O. W. T. Committee, "Web services business process execution language v2.0," OASIS Standard, Tech. Rep., 2007.

[21] M. Jaeger and H. Ladner, "Improving the qos of ws compositions based on redundant services," in *Proceedings of the Next Generation Web Services Practices (NWeSP)*, 2005.

[22] M. P. Papazoglou and W.-J. van den Heuvel, "Business process development life cycle methodology," *Communications of ACM*, vol. 50, no. 10, pp. 79–85, 2007.

[23] D. Bianchini, C. Cappiello, V. D. Antonellis, and B. Pernici, "P2s: a methodology to enable inter-organizational process design through web services," in *Proceedings of the 21st International Conference in Advanced Information Systems (CAiSE'09)*, 2009, pp. 334–348.

[24] T. Saaty, *The Analytical Hierarchy Process*. McGraw Hill, 1980.

[25] C. Chen, *Fuzzy Logic and Neural Network Handbook*. McGraw Hill, 1996.

[26] S. Chen and C. Hwang, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*. SpringerVerlag, 1992.

[27] D. Jackson, "A direct path to dependable software," *Commun. ACM*, vol. 52, no. 4, pp. 78–88, 2009.

[28] P. W.H., *Failure-Tolerant Computer Design*. Academic Press, 1965.

[29] A. Avižienis, "Design of fault-tolerant computers," in *AFIPS '67 (Fall): Proceedings of the November 14-16, 1967, fall joint computer conference*, 1967, pp. 733–743.

[30] W. G. Bouricius, W. C. Carter, and P. R. Schneider, "Reliability modeling techniques for self-repairing computer systems," in *Proceedings of the 1969 24th national conference*, 1969, pp. 295–309.

[31] J. Floch, S. O. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjørven, "Using architecture models for runtime adaptability," *IEEE Software*, vol. 23, no. 2, pp. 62–70, 2006.

[32] Y. Xie and A. Aiken, "Context- and path-sensitive memory leak detection," in *ESEC/SIGSOFT FSE*, 2005, pp. 115–125.

[33] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, "Composing adaptive software," *IEEE Computer*, vol. 37, no. 7, pp. 56–64, 2004.

[34] J. Walkerdine, L. Melville, and I. Sommerville, "Dependability properties of p2p architectures," in *Peer-to-Peer Computing*, 2002, pp. 173–174.

[35] G. Spanoudakis, A. Zisman, and A. Kozlenkov, "A service discovery framework for service centric systems," in *IEEE SCC*, 2005, pp. 251–259.

[36] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *WWW*, 2003, pp. 411–421.

[37] M. Colombo, E. D. Nitto, and M. Mauri, "Scene: A service composition execution environment supporting dynamic changes disciplined through rules," in *ICSOC*, 2006, pp. 191–202.

[38] E. Rukzio, S. Siorpaes, O. Falke, and H. Hussmann, "Policy based adaptive services for mobile commerce," in *The Second IEEE International Workshop on Mobile Commerce and Services, 2005. WMCS '05*, July 2005, pp. 183–192.

[39] S. Neti and H. Müller, "Quality criteria and an analysis framework for self-healing systems," in *International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007)*, 2007.

Cinzia Cappiello Biography text here.

Barbara Pernici Biography text here.

8 Bucchiarone, A.; Kazhamiakin, R.; Cappiello, C.; di Nitto, E.; Mazza, V.: *A Context-driven Adaptation Process for Service-based Applications*. In: Workshop on Principles of Engineering Service-Oriented Systems (PESOS 2010). – accepted for publication

A Context-driven Adaptation Process for Service-based Applications*

Antonio Bucchiarone and Raman Kazhamiakin*
Cinzia Cappiello, Elisabetta di Nitto and Valentina Mazza
*FBK-IRST, Trento, Italy
{bucchiarone,raman}@fbk.eu
Politecnico di Milano, Italy
{cappiell,dinitto,vmazza}@elet.polimi.it

ABSTRACT

When building service-oriented systems the evolution of requirements and context is the norm rather than the exception. Therefore, it is important to make sure that the system is able to evolve as well without necessarily starting a completely new development process, and possibly on the fly. In this paper we specifically focus on the role of the *context* in the adaptation activities. For us context has various different facets as it includes information ranging from the situation in which users exploit a service-based application to the conditions under which the component services can be exploited. We elaborate on how and when the context should be defined, exploited, and evolved, and on the impact it has on the various activities related to adaptation of service-based applications. We use a case study to exemplify our first findings on this subject.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*Methodologies*

Keywords

Service-oriented Systems, Context-awareness, Adaptation

1. INTRODUCTION

Traditional software systems are usually designed in order to address a specific set of requirements within a specific execution context. Even though the evolution of requirements and execution context is considered possible, it is assumed that it will have an impact on the new versions of the software system, not on the one that is currently under operation. This assumption is not anymore true when we consider those open systems that are built by composing existing services available on the network. In these cases we should consider that execution can be affected by both requirements and context defined as composed of any information that can be used to characterize persons, places or objects that are considered relevant to the interaction between a user and the application, including users and applications themselves [4]. The approach presented in this paper considers the evolution of requirements and context as the norm rather than the exception, and, as such, it is necessary to design a system that is able to evolve as well without necessarily starting a completely new development process, and possibly on the fly. As we discuss in the related work section (see Section 2), the literature offers technical approaches to manage the on the fly adaptation of service-based applications. However, to our knowledge, a comprehensive approach to design and develop adaptable Service-Based Applications (SBAs) is still missing. Our work tries to fill this gap. In this paper we specifically focus on the role of the *context* in the adaptation activities. For us context has various different facets as it includes information ranging from the situation in which users exploit a service-based application to the conditions under which the component services can be exploited. In general, the identification of the aspects that should be part of the context depends on the specific application and should be performed very early. In fact, starting from the requirement analysis phase where, in parallel to the precise definition of requirements, a proper *context model* has to be defined. This context model is the basis for the definition of those situations that trigger the adaptation or evolution of a service-based application, and, at runtime, enables the identification and the collection of the proper context information. Of course, as any other software artifact, this context model is not fixed once for all but can evolve together with the application, and therefore its evolution has to be managed and kept under control as well.

In this paper we start studying these aspects in detail and exemplify them through a case study. Consistently, the paper is structured as follows. Section 2 discusses previous contributions that dealt with context-aware applications and adaptation needs and describes the main open issues that are addressed in this paper. Section 3 describes a scenario that shows how the adaptation requirements might be relevant in the development of a service-based application. Section 4 identifies the additional requirements and phases that should be considered in the design life-cycle of an adaptive SBA while the context elements considered as relevant

*The research leading to these results has received funding from the European Community Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

PESOS '10, May 2-8 2010, Cape Town, South Africa

in the adaptation process are detailed and modelled in Section 5. A context-driven adaptation process is proposed in Section 6. Finally, Section 7 presents the instantiation of the proposed process on a case study in the e-government domain.

2. RELATED WORK

In many SBAs the role of the context is fundamental in realizing the adaptation functionalities [5]. Indeed, contextual changes are often the key factors that entail the SBA adaptation and somehow drive the way the adaptation is performed. [15] contains an analysis of the context factors in context-aware computing and in software engineering. The knowledge of the computing environment (network characteristics,resources), of the user (profile, location, social situation), of the physical parameters (all the measurable properties of the environment such as the noise level, the pressure, the temperature) contribute to the context knowledge. This information is often codified in a so called *context model*. The literature is rich of proposals for context models. In [16] context representations are classified as *Key-value models* (key-value pairs are used to represent the features of a system), *Mark-up scheme models* (a hierarchical structure for context representation), *Graphical models* (general purpose modeling instruments are used to represent context information), *Object Oriented Models* (each context information is represented by a different object and data can be accessed by defined interfaces), *Logic Based models* (context is expressed in terms of facts, expressions, and rules) and, finally, *Ontology Based models* (information about real world is represented using data structure understandable by the computers). In [17] a Context Dimension Tree is used to represent the context factors with different levels of detail. The model includes constraints and relationships among dimension values to eliminate meaningless context configurations. Such model has been proposed for data tailoring in the database domain, but could be easily adapted to the context of SBAs (see Section 5). Context-aware applications are able to sense their current context and adapt their behaviour accordingly. Even if automatically enacting adaptation is desired [8], some approaches in the literature suggest the need to have a semi-automatic approach in which users are able order to choose the best adaptation strategy [6]; this would avoid the execution of undesired adaptation actions. [10] proposes an approach for context-aware service adaptation based on an ontology for the context modeling and a learning mechanism for the mapping of the context configuration to the appropriate adapted service. Due to the uncertainty behind the context of an application, Cheung et al. [14] propose a fuzzy-based service adaptation model to improve the effectiveness of service adaptation by means of fuzzy theory. The formalization of the service adaptation process is made using fuzzy linguistic variables to define the context situations and the rules for adopting the policies for the implementation of the services.

While the short overview we have provided does not aim at being comprehensive, it should have given the idea that a number of approaches to context modeling and to the development of context-aware adaptable SBAs are being proposed. However, none of them provides a method to develop and execute adaptable SBAs. In [2] we have defined some design guidelines and a life-cycle for the development of adaptable service based applications. In this paper we want to enrich the life-cycle adding information about the context. Therefore, the aim of this paper is to propose a context model for the service based applications and to define how the information contained in the context model could be exploited to enact adaptation during each phase of the life-cycle.

3. E-GOVERNMENT SCENARIO

The motivating scenario that we use in this paper refines the e-government case study [13] of the S-Cube Project. The idea is to have a SBA that realizes e-government processes able to support citizens in various activities. The application aims at provisioning and integrating various public services, including the health-care services, administrative procedures towards citizens, information services, personal assistance services, etc. These services range from Internet services (e.g., for booking a visit to a doctor online, online payment, route planning) to local services (e.g., specific medical centers and labs, monetary assistance), to personal and human-provided ones (medical assistance at home, fiscal procedures, monetary assistance, census operations). Moreover, public authorities own and expose to the citizens various e-government processes and procedures, where the single services of private bodies and public organizations are composed. An example of an integrated process is the reservation of a medical visit, where the reservation service is integrated with the local services of the medical centers, with the route planning system and public transportation services to drive the patient to the doctor and inform it about the available transport means. Figure 1 presents the general picture of the scenario, it includes all possible services, subdivided in different categories, and different actors of the application: Citizen, Service Providers, Public Authorities and the Service Integrator. The Service Integrator will be the principal actor that realizes the e-government SBA using the principles that we define in this paper.

In the scenario introduced above different contextual dimensions are heavily exploited, ranging from computational context (e.g., devices and channels used by the citizens to access/receive information from the administrative services), to physical context (e.g, user location), user context (its. preferences, social status, medical card), and even to business aspects (e.g., the emergency mode of the e-government procedures). In the following we will see how to develop a service-based application able to adapt in the presence of changes within these contextual dimensions.

4. ADAPTATION IN SBA

Figure 2 shows the life-cycle for adaptable SBAs we have presented in [2]. It highlights not only the typical design-time iteration cycle that leads to the explicit re-design of an application, but it also introduces a new iteration cycle at run-time that is undertaken in all cases in which the adaptation needs are addressed on-the-fly. The two cycles coexist and support each other during the lifetime of the application. Figure 2 also shows the various adaptation- and monitoring-specific actions (boxes) carried out throughout the life-cycle of the SBA, the main design artifacts that are exploited to perform adaptation (hexagons), and the phases where they are used (dotted lines). The initial phase of the life-cycle is the (Early) *Requirement Engineering and Design*; at this phase the adaptation and monitoring require-

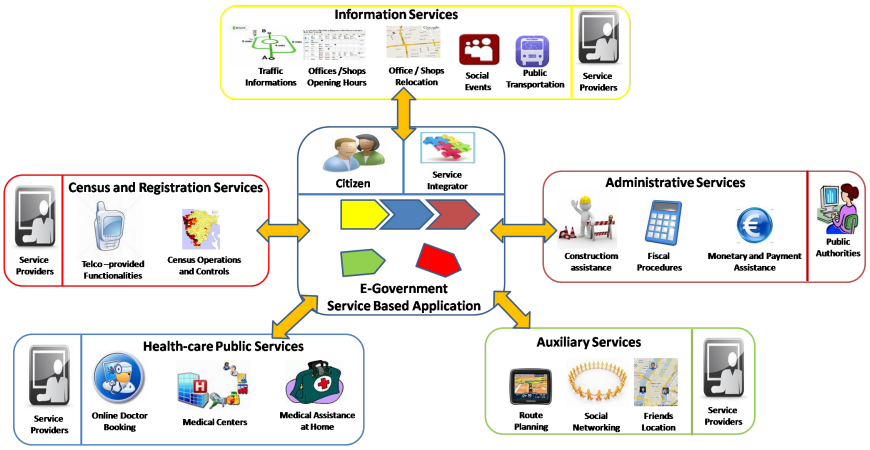


Figure 1: General picture of the e-government scenario.

ments are used to perform the design for adaptation and monitoring. During SBA *construction*, together with the construction of the SBA, the corresponding monitors and the adaptation mechanisms are being realized. The *deployment* phase also involves the activities related to adaptation and monitoring: deployment of the adaptation and monitoring mechanisms and deployment time adaptation actions (e.g., binding). During the *operation and management* of the application phase, the run-time monitoring is executed, using some designed properties, and help the SBA to detect relevant context and system changes. Here, we can proceed in two different directions: executing evolution or adaptation of the SBA. In the first case we re-start the right-side of the cycle with the requirements engineering and design phase while in the second case we proceed identifing adaptation needs that can be triggered from monitored events and adaptation requirements. An adaptation need formally characterizes a specific problem-situation that demands for adaptation. It takes into account each monitoring events and tries to answer the following questions: What needs to be adapted? What is the cause? What should be the outcome/what is the aim?. For each adaptation need it is possible to define a set of *suitable strategies* that define the possible ways to achieve the adaptation requirements and needs given the adaptation mechanisms made available. Each adaptation strategy (e.g., service substitution, re-execution, re-negotiation, compensation, etc..[2]) can be characterized by its complexity and its functional and non functional properties. The identification of the most suitable strategy is supported by a *reasoner* that also bases its decisions on multiple criteria extracted from the current situation and from the knowledge obtained from previous adaptations and executions. After this selection, the *enactment of the adaptation strategy* is automatically or manually performed. The execution of all activities and phases in all runtime phases may be performed autonomously by SBAs or may involve active participation of the various human actors.

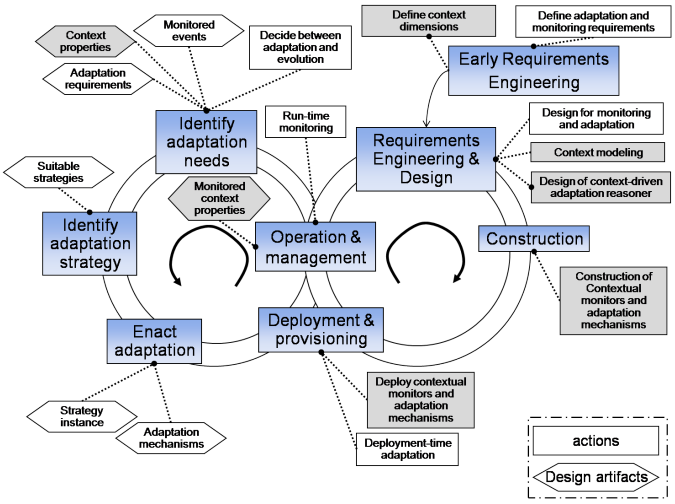


Figure 2: The Life-Cycle of Adaptable SBAs.

5. THE CONTEXT MODEL

The aim of our context model is the formalization of the most relevant aspects characterizing the SBA. The model we propose has been inspired by [17] and is a XML representation of the main components of the context for service based applications. It contains six main dimensions able to describe the status of an application: *Time, Ambient, User, Service, Business* and *Computational* Context. Each dimension in the XML tree, can have sons able to refine each factor. An example of the model is shown in Figure 3 and is discussed in Section 7). The *TimeContext* dimension refers to the information about the time in which the access to the application occurs; it could be expressed in absolute terms (defining a precise date) or it could indicate a part of the day (morning, afternoon, evening, night). The *AmbientContext* can be related to the space factor (expressed in terms of an address by which the user is accessing) or to the environmental condition of the user (the value of some measurable physical parameter). The *UserContext* dimension contains the information about the privileges, the roles, or the preferences the user has in the application. Moreover, such dimension permits to express the goal the user wants to achieve. Information about the services in the application are codified under the *ServiceContext* dimension. This element lists all the services together with their status (if a previous execution reported an error, or if it is available),

the time of the last failure (if it makes sense), and the similar services. The latter information could be exploited if a service needs to be substituted with another one. The *BusinessContext* dimension takes into account business application factors. Finally, the *ComputingContext* dimension specifies the software and/or hardware characteristics that are available at the end user side. Such element permits to specify, for example, the device, the operating system, or the web browser the user is using for accessing the application services.

6. A CONTEXT-DRIVEN ADAPTATION PROCESS

The role of context, the influence of its factors on the executability of the applications, and the possible mechanisms for their self-adaptation or human-assisted adaptation, should be properly modelled, designed, and engineered through the whole life-cycle of the system. Referring to the life-cycle of Figure 2, we introduce here a number of actions and design artefacts that are needed to properly build a context model and exploit it to support the adaptation of applications. The context information should be considered and explicitly captured since the very beginning of requirements engineering. In parallel with the elicitation and refinement of requirements we start understanding, which context factors should be considered for the purpose of possible adaptations. The identification and characterization of the context factors of the SBA is continued through the design phase and results in the activity that we call *context modeling*. In parallel, another important aspect of the design activities is the identification of *Context-specific adaptation triggers and requirements*. This refers to the rules and criteria that define the critical context properties and configurations from the adaptation point of view: the context properties that entail SBA adaptation (i.e., to decide *adaptation needs*) and the rules and criteria that define what should be adapted and how (i.e., to decide *adaptation strategies*). These properties will be used to construct the *context-driven adaptation reasoners* that drive the adaptation process at runtime. During the construction phase *contextual monitors* and *contextual adaptation mechanisms* are developed. The first ones are needed to deliver important information about context states and changes during SBA execution. The latter are used to realize different adaptation strategies. Even if some adaptation mechanisms could be independent of a specific context, some others could be dependent on it, for instance, a user-specific service selection policy or the mechanism needed to adapt the result of a service invocation on the basis of the characteristics of the specific devices used by end users (e.g., PC, PDAs, or conventional phones). We remark that the process of identification, modeling, and refinement of the contextual factors and properties is iterative. Indeed, with the definition of new elements of the application and monitoring/adaptation, new factors may be discovered, triggering new adaptation needs and corresponding strategies.

The artifacts and elements developed at design time, i.e., the context model, the contextual monitors, and the adaptation mechanisms are exploited at run-time to activate and drive SBA adaptations. First, the contextual monitors are used to evaluate the *context properties*. Based on them the contextual adaptation reasoners make decisions whether the adaptation should be triggered or not. The decision is driven by the rules identified at design phase and encoded during the construction of the appropriate mechanisms. As a next step, the appropriate adaptation strategy is selected and then enacted by the adaptation framework. As in case of adaptation needs, the selection of the adaptation strategy, as well as its activation, may rely upon and exploit the knowledge about context, e.g., exploit information about the user device to select services and deliver information, about network properties (such as use of GPRS or 3G networks) to activate SLA negotiation.

6.1 Context Modelling

During context modeling the model described in Section 5 is instantiated by identifing the list of context dimensions that can trigger an adaptation or evolution of the functionalities provided by the analyzed SBA. In fact, different context dimensions are relevant for different applications. For example, the *ambient* dimension is irrelevant for all the applications that do not require to change on the basis of the geographical position in which they are used. Once the list of the relevant context dimensions has been defined, the context requirements should be refined by modelling the context dimensions in terms of their reference domain. To enable the design for adaptation aspects, it is first necessary to define if the standard context representation hierarchy is fine for the application or new categories should be defined (e.g., seasons) and new representations identified (e.g., room numbers instead of GPS positions). Second, it is necessary to define the granularity to be used for measuring the context dimensions. For example, as regards our scenario and the *ambient* dimension, the service for the identification of the health-care public services should adapt its output on the basis of the exact geographical position in which the user accesses it while for the identification of the administrative services, it could be sufficient to retrieve the city from which the user accesses the platform. This first analysis allows the designer to provide the following outcomes:

- The instantiation of the generic context model (Section 5) with respect to the given SBA: the relevant context dimensions, relevant sub-elements of these dimensions, and important set of values (ranges) these elements may have with respect to the analyzed SBA.
- The description of the relationships between SBA elements (services, processes and/or subprocesses, etc.) and the context dimensions in terms of type of impact and specific context values to monitor. This mapping defines the starting point for the definition and modeling of adaptation triggers and adaptation requirements.

We would like to stress the fact that the context modeling is an iterative process: the new dimensions and elements of the model may be added when the application is detailed and new elements (including also monitoring and adaptation mechanisms) are added.

6.2 Modeling Adaptation Triggers and Requirements

Once the context model and its relation with the application model is defined, it is necessary to properly capture and define the adaptation aspects. In particular, it is important to define when the contextual changes are critical for the

SBA functioning (i.e., adaptation triggers) and what should be done or achieved when these changes take place (adaptation needs). Depending on the context dimension/element and on the specific requirements of an SBA, the definition of adaptation trigger may vary. As a result, the corresponding context monitors needed to detect those trigger will have different, sometimes application-specific, forms and realizations. Furthermore, in certain cases the adaptation trigger does not correspond to some value of a particular single context element, but is characterized by the complex combination of different context dimensions/factors. In the simplest case, such situations may be directly encoded using the capabilities provided by the existing monitoring frameworks, such as Dynamo [1]. In other cases, a sophisticated reasoners may be necessary. In particular, in [7], for instance, a context represents the combination of users personal assets (agenda, location, social relations, and money), and the critical context changes characterize some critical combination of those assets. A dedicated analysis mechanisms is used to identify those situations (asset conflicts) and to trigger adaptation solutions.

Each adaptation trigger can be associated with the adaptation requirements that define what should be done in order to align the SBA with its context. Considering the contextual aspects, it is necessary to take into account that adaptations may be performed (*i*) to *customize* the system in order to fit to the situation in which the application currently operates, (*ii*) to *optimize* the system in order to improve certain (usually quality-of-service) issues and characteristics of the system, or (*iii*) to *prevent and avoid* future faults or undesirable situations in the execution of the system. The customization is often driven by changes in the application context and especially by time, ambient, user, business, and service context dimensions. Optimization is more related to the service and computing context dimensions but can be also triggered by changes in the users' preferences. Finally, prevention is mostly related to the service and computing context since changes of the execution environment might increase the risk of failures and the need for prevention. The adaptation requirements are captured and realized by a set of adaptation strategies. The selection of an adaptation strategy to apply may depend on variety of factors [2], such as scope of the change (i.e., whether the change affects only a single running instance of the SBA or influences the whole model), its impact (the possibility of the application to accomplish its current task or necessity to retract), etc. Besides, the contextual factors should also be considered: the different strategies may be applied upon the same trigger. For example, the decision to substitute a service or to retry with the existing one may depend on the profile of the user. Again, the selection of the adaptation strategy may be characterized by a simple rule or a requires advanced reasoning, based on some ontologies or ad-hoc models, or may involve user decisions [7]. Table 1 relates some of the common adaptation strategies with the changes in different context dimensions. In conclusion, the outcomes of this modeling step include the following components:

- Characterization of adaptation triggers as the description of the context configuration, in which the adaptation should be applied.
- Relation of adaptation triggers to possible adaptation strategies used to realize the corresponding adaptation needs. This relation may be also equipped with the characterization of the contextual conditions that drive the selection of one or another strategy.

6.3 Construction of Contextual Monitors and Adaptation Mechanisms

The results of the analyses performed in the previous phases support the SBA designers in the realization of the monitors and adaptation strategies that should be included in the SBA design. The need for the analysis of the context and the deployment of specific adaptation strategies requires the construction of dedicated platforms and the insertion of monitors able to detect the context changes and trigger the correspondent action. As we already discussed in previous subsection, the realization of such mechanisms may rely on the existing solutions or may require their extension and even completely novel approaches. For example in the Dynamo monitoring framework [1], the monitoring components for context observation are realized as services, which allow for the extension of the platform and even for run-time addition/removal of monitors. Among the other enabling platforms for realizing such context-based monitoring and adaptation solutions one can consider dynamic aspect-oriented programming [12] that provides means for application developers to state when and how behaviour of an application should be adapted. These approaches have a limited level of semantic expressiveness since they rely on general-purpose language constructs. Hence, application developers have to express adaptations in the format of the supporting platform which often remains difficult to be understood. Domain-specific languages offer a solution to enable programmers to reason at a higher semantic level. In [11] a language that provides higher-level constructs for expressing the adaptation of application behaviour due to a change in context is proposed. Other approaches are based on middleware-based approaches. For example, in [3], authors present a framework for self-adaptive components that follows the Event-Condition-Action pattern and makes use of events emitted by a context-aware service that provides information about the execution context of the application. Furthermore, various rule engines can be also used. In [9], a dynamic adaptation framework that adapts service objects in a context-aware policy-based manner, using metatypes is proposed. The system includes in the policy document a-priori information for adaptation in the form of default behaviours and known adaptations. The system is also able to deal with adaptation requirements that were unprecedented when the service was designed and compiled. In fact, reconfiguration intelligence can be incorporated at runtime by modifying the policy declarations and the inclusion of new behaviours in the form of new metatypes. The outcomes of this construction step include the following components:

- Design and realization of monitors able to detect changes in the context dimensions on the basis of the specifications defined in the previous phase.
- Realization of a platform that on the basis of the context analysis define and trigger the corresponding adaptation action.

7. PROCESS EVALUATION

In this section we want to explain how the context-aware design process described in the section 6 is applied to a case

Table 1: Suitability of adaptation strategies to react to context changes

Strategy	Time	Ambient	User	Service	Computing	Business
Service substitution	X	X	X	X	X	X
Re-execution				X	X	
Re-composition				X	X	X
Fail			X	X	X	X
Service concretization	X	X	X	X	X	X
Re-negotiation			X		X	X
Compensation				X	X	
Trigger Evolution		X	X	X	X	X

study in the e-government domain that has been defined in the S-Cube Project [13]. In the scenario introduced above

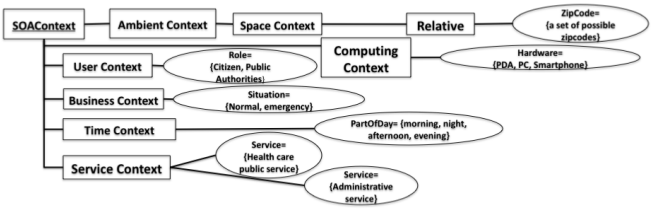


Figure 3: The Context Model for the e-Government scenario.

different contextual dimensions are heavily exploited, ranging from computational context (e.g., devices and channels used by the citizens to access/receive information from the administrative services), to physical context (e.g, user location), user context (its. preferences, social status, medical card), and even to business aspects (e.g., the emergency mode of the e-government procedures). In particular, let us consider the Health-care public service that one user can access to search for a doctor. If the request occurs during the night, the list of the available doctors is different with respect to the one returned during the morning. For this reason, the *TimeContext* dimension will be expressed by means of the *PartOfDay* element whose values will belong to the following set: {*morning, afternoon, evening, night*} (see the figure 3). In a similar way, some of the service outputs could depend on the location of the user accessing the system. In fact, if a citizen wants the list of the nearest pharmacies, the sorting of the returned items depends on the zipcode of the area from which the request comes. In a similar way the behaviour of the application will depend on the situation (normal life or emergency conditions) the users are living. The study of the application and of the involved services will guide the definition of the set of values associated to the context dimensions. An example of a simple context model for the proposed scenario can be found in the Figure 3; the model reports, for each considered dimension, all the admissible values.

After the definition of the context model, it is possible to go through the next phase devoted to the identification of the associations of the critical context changes (adaptation triggers) to the adaptation needs. In Table 2, for each service involved in the application the relationship with the context factors is highlighted. It is possible to notice, for example, that the access to the Administrative or to the Health-care public services, could depend on the values of the *TimeContext* dimension in the context model.

Changes in the context have to be managed and suitable adaptation strategies have to be defined. Referring to the e-government scenario, if we consider a mobile user accessing to the e-Health service, we can image that modifications in the location (expressed using the zipcode for the specific context dimension) will require the re-execution or the substitution of the specific e-Health service. Moreover the transition from a normal to emergency situation could require the recomposition of the application. During the second phase of the process, all the context changes and the corresponding adaptation strategies have to be identified. On the basis of the analyzed scenario, the context requirements, and the characteristics of the adaptation strategies, process designers in the last phase should build a table *Service/Adaptation Strategy* in which for each service the selected adaptation strategies are identified. Table 3 details the *Service/Adaptation Strategy* table for the e-government scenario. Since almost all the functionalities depend on the whole set of the context dimensions, they are also suitable for the adoption of a large set of adaptation strategies. The public ownership of the administrative and census and registration services and thus the unavailability of a registry containing similar services make instead the service substitution and concretization not adoptable. Compensation is not suitable for all the Information services since the only read interaction with these services does not require the intervention of compensation operations. The table *Service/Adaptation Strategy* provides a comprehensive view of all the applicable strategies and thus supports the designer in the identification of the elements to design and develop to enable the construction of a contextual adaptive SBA. For each service and for each related context dimension, one or more adaptation strategies should be selected on the basis of technical constraints and functional and non-functional requirements. For example, for all the critical services in which a rapid response is needed, the adoption of adaptation strategies that increase the response time (e.g., re-execution, re-negotiation, re-composition, trigger evolution) is not acceptable.

8. CONCLUSION AND FUTURE WORK

This paper focuses on the role of the context in the adaptation activities. It proposes a framework to support the design of SBAs that targets the adaptation requirements raised by context changes. The approach has been described on the basis of a novel life-cycle that emphasizes the relevance of the context elements in the different facets of adaptation, both during the design phase and at run-time. The paper considers all the issues related to the design of SBAs able to evolve together with the requirements and the execution context. The context has been modeled by considering a set of all the dimensions that can generally influence the system behaviour. On the basis of this context model, the proposed approach provides guidelines for the identification of the relevant context dimensions to monitor and for the definition

Table 2: Service/Context table for the E-government scenario

Application	Time	Ambient	User	Service	Computing	Business
Health-care public services	X	X	X	X	X	X
Administrative services	X	X	X	X	X	X
Census and registration services		X	X	X	X	X
Information services	X	X	X	X	X	X
Auxiliary services		X		X	X	X

Table 3: Service/Adaptation Strategy table for the e-government scenario

Adaptation Strategy	Health-care public services	Administrative services	Census and registration services	Information services	Auxiliary services
Service substitution	X			X	X
Re-execution	X	X	X	X	X
Re-composition	X	X	X	X	X
Fail	X	X	X	X	X
Service concretization	X			X	X
Re-negotiation	X			X	X
Compensation	X	X	X		X
Trigger Evolution	X	X	X		

of the adaptation triggers able to link context changes with suitable adaptation strategies. The effectiveness of the discussed principles and guidelines has been evaluated by considering a real case study based on an e-governement scenario. Results witness the capability of the context-driven adaptation process to capture the key aspects of adaptation and support designers from the requirements elicitation to the construction of proper adaptation mechanisms. Our future roadmap includes a refinement of the adaptation process presented in this paper, its formalization, and validation. We also intend to work on the development of mechanisms and tools supporting the methodology, building on top of the actions and artifacts identified in the proposed life-cycle.

9. REFERENCES

[1] L. Baresi and S. Guinea. Dynamo: Dynamic monitoring of ws-bpel processes. In *ICSOC*, pages 478–483, 2005.

[2] A. Bucchiarone, C. Cappiello, E. di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore. Design for adaptation of Service-Based applications: Main issues and requirements. In *International Workshop on Engineering Service-Oriented Applications (to appear)*, 2009.

[3] P.-C. David and T. Ledoux. Wildcat: a generic framework for context-aware applications. In *MPAC*, pages 1–7, 2005.

[4] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.

[5] A. K. Dey and G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, 2000.

[6] C. Efstratiou, K. Cheverst, N. Davies, and A. Friday. An architecture for the effective support of adaptive context-aware applications. In *MDM '01: Proceedings of the Second International Conference on Mobile Data Management*, pages 15–26, London, UK, 2001. Springer-Verlag.

[7] R. Kazhamiakin, P. Bertoli, M. Paolucci, M. Pistore, and M. Wagner. Having Services "YourWay!": Towards User-Centric Composition of Mobile Services. In *Future Internet Symposium*, 2008.

[8] J. Keeney and V. Cahill. Chisel: A policy-driven, context-aware, dynamic adaptation framework. *Policies for Distributed Systems and Networks, IEEE International Workshop on*, 0:3, 2003.

[9] J. Keeney and V. Cahill. Chisel: A policy-driven, context-aware, dynamic adaptation framework. In *POLICY*, pages 3–14, 2003.

[10] M. Moez, C. Tadj, and C. ben Amar. Context modeling and context-aware service adaptation for pervasive computing systems. *International Journal of Computer and Information Science and Engineering*, 2008.

[11] J. Munnelly, S. Fritsch, and S. Clarke. An aspect-oriented approach to the modularisation of context. In *PerCom*, pages 114–124, 2007.

[12] A. Nicoara and G. Alonso. Dynamic aop with prose. In *International Workshop on Adaptive and Self-Managing Enterprise Applications (ASMEA)*, pages 125–138, 2005.

[13] E. D. Nitto, V. Mazza, and A. Mocci. Collection of industrial best practices, scenarios and business cases, 2009.

[14] J. C. Ronnie Cheung, Gang Yao and A. Chan. A fuzzy service adaptation engine for context-aware mobile computing middleware. *International Journal of Pervasive Computing and Communications*, 2008.

[15] S-Cube. Codified human-computer interaction (HCI) knowledge and context factors, 2009.

[16] T. Strang and C. L. Popien. A context modeling survey, September 2004.

[17] L. Tanca, A. Miele, and E. Quintarelli. A methodology for preference-based personalization of contextual data. In *ACM EDBT 2009*, 2009.

9 Gehlert, A.; Bucchiarona, A.; Kazhamiakin, R.; Metzger, A.; Pistore, M.; Pohl, K.: *Exploiting Assumption-Based Verification for the Adaptation of Service-Based Applications*. In: Proceedings of the 25th Annual ACM Symposium on Applied Computing, Track on Service Oriented Architectures and Programming, March 21 - 26, 2010, Sierre, Switzerland. 2010.

Exploiting Assumption-Based Verification for the Adaptation of Service-Based Applications

Andreas Gehlert
University of Duisburg-Essen
Schützenbahn 70
45117 Essen, Germany
andreas.gehlert@sse.uni-due.de

Antonio Bucchiarone
FBK-IRST
via Sommarive 18
38100 Trento, Italy
bucchiarone@fbk.eu

Raman Kazhamiakin
FBK-IRST
via Sommarive 18
38100 Trento, Italy
raman@fbk.eu

Andreas Metzger
University of Duisburg-Essen
Schützenbahn 70
45117 Essen, Germany
andreas.metzger@sse.uni-due.de

Marco Pistore
FBK-IRST
via Sommarive 18
38100 Trento, Italy
pistore@fbk.eu

Klaus Pohl
University of Duisburg-Essen
Schützenbahn 70
45117 Essen, Germany
klaus.pohl@sse.uni-due.de

ABSTRACT

Service-based applications (SBAs) need to operate in a highly dynamic world, in which their constituent services could fail or become unavailable. Monitoring is typically used to identify such failures and, if needed, to trigger an adaptation of the SBA to compensate for those failures.

However, existing monitoring approaches exhibit several limitations: (1) Monitoring individual services can uncover failures of services. Yet, it remains open whether those individual failures lead to a violation of the SBA's requirements, which would necessitate an adaptation. (2) Monitoring the SBA can uncover requirements deviations. However, it will not provide information about the failures leading to this deviation, which constitutes important information needed for the adaptation activities. Even a combination of (1) and (2) is limited. For instance, a requirements deviation will only be identified after it has occurred, e. g., after the execution of the whole SBA, which then in case of failures might require costly compensation actions.

In this paper we introduce an approach that addresses those limitations by augmenting monitoring techniques for individual services with formal verification techniques. The approach explicitly encodes assumptions that the constituent services of an SBA will perform as expected. Based on those assumptions, formal verification is used to assess whether the SBA requirements are satisfied and whether a violation of those assumptions during run-time leads to a violation of the SBA requirements. Thereby, our approach allows for (a) pro-actively deciding whether the SBA requirements will be violated based on monitored failures, and (b) identifying the specific root cause for the violated requirements.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Monitor*

Keywords

Service-oriented computing, verification, monitoring

1. INTRODUCTION

1.1 Need for Adaptation

Service Based Applications (SBAs) are composed of services provided by service providers that are often different from the company that is operating the SBA [28]. Such a distribution of computational resources and software comes with the advantage to flexibly use any service available on the network and, therefore, to adapt the SBA to new business situations by, for instance, exchanging one service for another. This flexibility, however, comes at the cost of losing tight control over the SBA, as the SBA owner cannot control the provisioning, execution, management and evolution of externally provided services [28]. This means that the SBA designer must rely on the ability of the service providers to meet the expected functionality and quality of those services (encoded, for instance, as service-level agreements).

Once the SBA is put into operation, those expectations may—intentionally—or unintentionally – be violated; for instance, a service might fail. The operator of the SBA must not only recognise these violations but also decide whether those violations mean that the overall SBA requirements will no longer be met. In such a case an adaptation of the SBA can become necessary.

1.2 Limitations of Monitoring

Currently, monitoring is used to trigger the adaptation of a service-based application. However, existing monitoring techniques—as detailed below—exhibit several limitations

which impact on taking adaptation decisions. Failing to make those decisions may lead to unnecessary or harmful adaptations [20].

Monitoring individual services: It is possible to monitor specific events and elements of the SBA, such as monitoring the constituent services [15]. Such approaches recognise whether a service delivers the expected functionality or quality. However, it is unclear whether this violation of the contract eventually leads to a violation of the SBA's requirements. Without this information we cannot decide whether the SBA should be adapted or not. Assume, for instance, that a service takes $1s$ longer than expected. It may be the case that the service is part of a parallel control flow in the service composition and that such a delay does not have any impact on the performance of the parallel control flow and thus the overall quality of the SBA.

Monitoring service compositions / SBAs: The requirements to the whole service composition may be monitored [3]. In this way it is possible to check whether the composition behaves as required. However, in this case, the identification of the source of the requirements violation is not trivial. Assume, for instance that a service composition takes $30s$ longer than expected to terminate. "Debugging" as an additional step would then be needed to determine, which service(s) caused that delay. It is important to know the cause of the problem to compensate for it by adapting the system; e. g., one could replace the service that caused the delay.

Combined efforts: Even a combination of the above two techniques has limitations. Indeed, in case of complex SBAs, a variety of events and violations may occur. How to debug and identify a specific cause of the requirements violation in order to trigger proper adaptation actions remains an open problem in such a case. Additionally, even with the combined approach we can only identify a problem of the service composition when this is identified by monitoring. This especially means that it is not possible to "predict" whether a violation of a service contract (detected by monitoring individual services) may eventually lead to a violation of the requirements of the service composition.

1.3 Contribution of the Paper

In this paper we present an approach that aims at addressing the above limitations. More specifically, our approach is able to detect run-time problems and violations of SBA's requirements and to identify specific root causes for those problems in order to determine appropriate adaptation actions. To achieve this, our approach augments monitoring techniques (to detect service failures) with formal verification techniques (to determine requirements violations). The central idea of our approach is to observe specific properties—assumptions—that (1) are explicitly related to the requirements and (2) characterize the constituent services of the SBA. Thereby, our approach allows (a) verifying whether a problem can lead to a violation of requirements, and (b) tracing the violation to its root cause, which facilitates adaptation.

The remainder of the paper is structured as follows: In Section 2 we introduce the notion of assumptions and how those are exploited in our approach. This is followed by a scenario to illustrate the approach in Section 3. We detail the individual steps of our approach in Section 4 and discuss related work in Section 5. Section 6 discusses the results and possible future work.

2. ASSUMPTION-BASED VERIFICATION

The concept of "assumption" is well understood in software and requirements engineering. While requirements can be influenced and realised by the designer building the system, assumptions can only be affected by agents in the system's environment [19, 30]. Thus, neither the designer nor the system itself have any influence on the violation or validity of assumptions [31]. Provided that a system fulfils its requirements under a given set of assumptions, a violation of those assumptions may lead to a situation, in which the software system does not provide the expected quality anymore and, therefore, deviates from its requirements[9].

In the case of service-based applications, assumptions may characterize the constituent services (e. g., their interfaces, QoS parameters, etc.) and/or the context (e. g., infrastructure, business, context, user, etc.). Once the assumptions are established (e. g., expected QoS is agreed through a contract such as a service-level agreement), the designer of the SBA expects that those assumptions are valid during the design- and run-time phases of the system (e. g., the contract will not be violated by the provider).

To decide whether a violation of an assumption (e. g., local deviation from contracts of one or more services) leads to a violation of the SBA requirements, we need an analysis technique. For the purpose of this paper, we propose to exploit verification techniques in two ways. First, a verification of the system at design time allows us to prove that the design corresponds to the requirements provided that the assumptions hold. Second, if the re-verification executed at run-time reports a violation of the corresponding requirements, we conclude that the observed violation of an assumption will lead to problems in the SBA. As the assumptions characterize a specific element (e. g., a specific service), an appropriate adaptation action may be triggered (e. g., substitute a service).

The presented approach benefits from (1) the possibility to identify the specific source of the problem and trace it to the critical element of the domain that should be handled by adaptation; (2) only specific assumptions need to be monitored to identify potential problems (thus, no need to monitor the whole requirements and compositions); and (3) in case of long-running applications the run-time monitoring and analysis results may be exploited to trigger adaptation activities pro-actively, before the failure actually takes place.

3. ILLUSTRATIVE SCENARIO

We use the following example to illustrate our approach. The company "Green Transport IT Solutions" wants to support passengers in cities to use the available public transport systems and develops a service-based application, which computes travel routes for helping users navigate from one place in a city to another. This application is deployed on a mobile device and uses different services for each city depending on the location of the device. The system implements the following workflow (cf. Figure 1): after the user has entered his or her destination, the system locates the user and computes the quickest route to the next bus stop. After this step, the system calculates the public transport route to the destination bus stop. Lastly, the shortest footpath from this bus stop to the final destination is calculated.

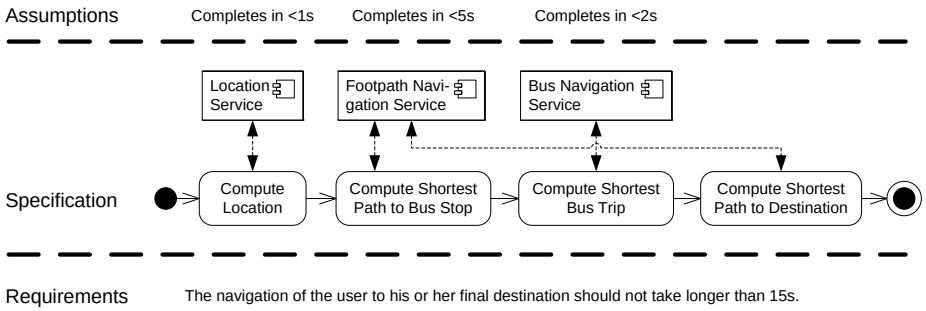


Figure 1: Assumptions, Requirements and Specifications of a Mobile Navigation Scenario

Figure 1 presents a non-functional requirement of the SBA at the bottom, the specification of the workflow in the middle and the assumptions regarding execution time at the top. The linear workflow of four steps should be executed within $15s$. The workflow invokes a location service once, a footpath navigation service twice and a bus navigation service once. According to the services' individual SLAs (i. e., service assumptions) their time to complete the request are at most $1s$ for the location service, $5s$ for the footpath navigation service and $2s$ for the bus navigation service. Furthermore, it is assumed that the service interaction times may be neglected (context assumption). Based on these assumptions, the initial verification performed during run-time proves that even in the worst case (upper bound for response time), the process terminates in less than $13s$ and, therefore, the requirement of the SBA is fulfilled.

Assume now the following two situations during run-time:

- *The location service takes* $3s$. While, its SLA is violated, an adaptation is not needed as the process takes now exactly $15s$. Monitoring the whole requirement or following our approach would thus have avoided an unnecessary adaptation.
- *The location service takes* $3s$ *and the the footpath navigation service takes* $7s$. If only the SBA would be monitored in that case, it would not be possible to understand which of the services have failed and which of the services should be adapted. As an example, if a different location service was chosen, the overall time of the service composition would be reduced to $16s$ and, therefore, the requirement would still be violated. In our approach, instead, if we re-verify the system, we can detect that the SBA requirement is violated regardless whether the assumption about the location service is violated or not. Therefore, in this case, it is necessary to exchange the footpath navigation service.

Besides the timed properties above, it is also possible to consider, monitor, and analyze functional aspects of the service composition. For example, the services may have complex interaction protocols consisting of several interactions, such as "make a navigation request" or "refine the destination". The SBA requirement in this case may express the deadlock freeness, while the assumption may express the necessity of all possible service implementations to follow the same interaction protocol. Indeed, if a specific service implementation does not follow the same protocol, run-time re-verification may check whether the new protocol is still compatible with the requirement.

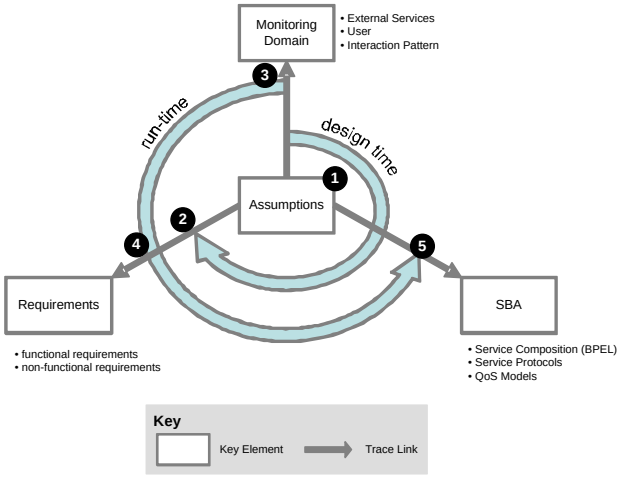


Figure 2: High-Level View on the Proposed Approach

4. DETAILED APPROACH

Figure 2 depicts the main elements of our approach. As explained in the introduction the approach builds on a clear separation between requirements for the SBA and the assumptions under which it is supposed to operate. Furthermore, we distinguish between the system itself and its context (or domain). As the SBA's constituent services are provided by different service providers and as these providers control those services with regard to functionality and quality, they belong to the SBA's domain.

Our approach involves the following five steps (cf. numbers in Figure 2):

1. This first step addresses the design-time where the designer documents assumptions and requirements separately.
2. In the second step the designer verifies the system and deploys the system only if the system passes the verification step.
3. At run-time the assumptions are monitored.
4. A violated assumption triggers a run-time re-verification.
5. If the SBA does not pass the verification given the violated assumptions, an adaptation is triggered, because this means that the SBA requirements are violated.

Those steps are described in more detail in the following subsections. Each subsection contains two aspects. The first aspect describes the concepts of each step in more detail. The second aspect describes techniques that can be used to implement the concepts of that step and thus demonstrates how our approach can be realized based on existing techniques for monitoring and verification. The aim of the selected techniques is to demonstrate the feasibility of our concepts only. Thus, we do not claim that we have selected the most appropriate techniques for each phase and, therefore, do not provide a discussion of possible alternative techniques.

4.1 Step 1: Separating Requirements & Assumptions

During the design step of the SBA its requirements and assumptions are documented together with the SBA model. Assumptions may be defined for the user behaviour, for the device on which the SBA runs and for the services provided by service providers. Here, we only focus on assumptions for services.

The assumptions may be extracted in different ways. In particular, it is possible to exploit domain knowledge (e. g., the properties of the telecommunication infrastructure adopted in different cities), to rely on historical information obtained and continuously updated during the execution of previous versions of the SBA, etc. The specific methods for eliciting domain assumptions is outside of the scope of this paper and will be addressed in future work.

4.1.1 Modeling Assumptions

Concepts: Assumptions may address functional (e. g., the interfaces and the protocol used by the service) and quality aspects (e. g., response time or availability) of services. As we verify the system later on (see Section 4.2) we will need formal techniques to describe these functional and quality aspects.

During this step, the assumptions, requirements and the elements of the SBA's specification are related by two types of trace links: First, the link between assumptions and requirements allows tracing a violation of this assumption back to its requirement. This link ensures that we do not only know the violated assumption but also the consequence for the SBA with respect to the fulfilment of its requirements. Second, the link between the assumption and the specification allows us to trigger an adaptation of the relevant part of the SBA once a violation of an assumption is detected.

Implementation: We propose to produce a set of template-based documents, which describe assumptions verbally and formally as well as their relations to the requirements and SBA specification. Such templates allow providing a name, a natural language description, a type, a formal description, the related requirements and the affected SBA element(s) (cf. Table 3 for examples). The assumption is described by a short and unique name as well as a natural language description. In addition, we document the assumption's type. Here, we distinguish between assumptions on functional, behavioral and non-functional properties of services. The formal description of the assumption is needed for verification and to monitor it at run-time. Lastly, the assumption is related to a requirement (related requirement row) and to a specific SBA element, which was designed using this assumption (affected SBA element row).

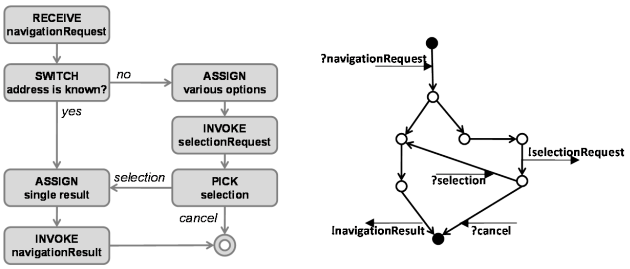


Figure 4: BPEL Protocol (left) and STS (right) of the Footpath Navigation Service

4.1.2 Modeling the Service-based Application

Concepts: A composite SBA may be defined using a language such as BPEL (Business Process Execution Language). The component services are represented by their interfaces (defined, e. g., in WSDL) that define a set of possible service operations, their parameters, data types, and binding protocol information.

Implementation: Formally, we represent the composed application and the protocol of the component services as a State Transition System (STS, [21]). A STS describes a dynamic system that can be in one of its possible states and can evolve to new states as a result of performing some actions. Actions are distinguished in input actions, which represent the reception of messages ("receive" and "onMessage" BPEL activities), output actions, which represent messages sent to external services ("invoke" and "reply"), and internal actions, modeling internal computations and decisions ("assign", "switch", etc.). The STS formalism may be extended to Timed Transition System [21] to capture time properties for representing assumptions on duration of activities, BPEL timeouts, etc.

An example of the abstract BPEL protocol and the corresponding STS of the Footpath Navigation Service is shown in Fig. 4. The protocol (left) starts when the navigation request is received. If the specified location is known the service returns an appropriate route (sends "navigationResult" message). Otherwise, the service prepares and sends to the requester the set of possible options with the "selectionRequest" operation. The requester may perform a selection (i. e., send the selection message) or cancel the procedure (sending the cancel message). In the former case the route is returned. In the latter case the procedure terminates. The protocol is reflected by the corresponding STS.

4.2 Step 2: Perform Design-Time Verification

Concepts: Once the system is designed, we need to formally verify that it satisfies its requirements under the specified assumptions. This formal verification is a necessary activity in our approach since this verification formally ensures that the deployed SBA corresponds to the requirements and assumptions documented in step 1. If this activity was not included in our approach and we detected a violation of an assumption at run-time (see Section 4.4), we would not know whether this violation is the source of the problem or whether the problem existed in the SBA in the first place.

Implementation: There are many SBA verification techniques available, which can be used for our purposes [25, 14, 22]. To verify for functional correctness and quality correctness the approach takes STSs and models in linear

Assumption:	Timing Assumptions
Description:	a1) The location service completes in at most 1s. a2) The footpath navigation service completes in at most 5s. a3) The bus navigation service completes in at most 2s.
Type:	Non-Functional
Formal Specification:	a1) $time(RECEIVED(locationResult)\ Since\ SEND(locationRequest)) < 1$ a2) $time(RECEIVED(footpathNavigationResult)\ Since\ SEND(footpathNavigationRequest)) < 5$ a3) $time(RECEIVED(busNavigationResult)\ Since\ SEND(busNavigationRequest)) < 2$
Related Requirement:	The system should be able to calculate the entire route to the destination in at most $15s$.
Affected SBA Element:	Location service, footpath navigation service, bus navigation service.

Figure 3: Documented timed assumptions

temporal logic (LTL) as input. Together with approaches, which translate property sequence charts into timed Büchi automata [2], the approach allows to bridge the gap between the design step and the analysis step.

To check whether the SBA satisfies its functional and non-functional requirements we use model checking techniques [14, 25, 26] that are able to check if the behavioural model of the SBA is conform to the given functional correctness properties representing the SBA requirements. We apply the approach proposed in [22] to verify that the SBA of our running example is correct. The approach takes as input the STSs representing the SBA and the services as well as the behavioral correctness properties defined in linear temporal logic (LTL). The properties may express for instance, deadlock freeness, ordering constraints on events, starvation, etc. It uses NuSMV [8] as a model-checker.

Instead of writing directly temporal properties, one can use the algorithm proposed in [2] that translates Property Sequence Charts (PSCs) into a Büchi automaton [7]. The notation of PSC [2] is very close to UML sequence diagrams, but has a formal underlying semantics suitable for verification purposes. After the verification of the behavior of the composition model at design-time, and in case a violation is detected at run-time, a counterexample (also in PSC-like form) that demonstrates the erroneous execution of the composition is provided.

In our scenario it is necessary to verify non-functional requirements, such as time properties. In particular, the global time constraint (execution time $\leq 15s$) should be satisfied. This requirement can only be satisfied if all the services participating in the application fulfill their own local timed constraints. To verify such time constraints, the approach proposed in [21] may be exploited. The approach extends the verification techniques presented above to take time properties and assumptions of service compositions into account. As input the approach takes the specification of the SBA and service behavior enriched with temporal aspects such as timeouts and constraints on activity durations, expressed as *Timed Transition Systems*. For specifying correctness properties that explicitly speak about time, the approach exploits the *duration calculus* (DC) notation [17]. Formally, the notation is close to the model of LTL and PSC; it enables the use of similar verification algorithms (the NuSMV model checker has been used for this purpose).

4.3 Step 3: Monitor Assumptions at Run-Time

Concepts: Upon the completion of the second step the system is deployed and used. To identify service faults, we need to monitor the services—more precisely we need to monitor the services w. r. t. the fulfilment of their service level agreements. Therefore, the assumptions (expressing the SLAs) should be mapped to monitoring rules, which are in turn used by a monitoring engine to monitor the SBA.

Implementation: For the assumptions monitoring purpose we adopt the integrated Astro/Dynamo framework and the corresponding monitoring language [6].

The choice is motivated by the capabilities and expressive power of the framework. First, the language permits specifying properties in a declarative way using the notations similar to those used for the verification (i. e., using temporal logic constructs similar to LTL and DC). In particular, it allows for capturing timing and statistical information about process activities. For example the property that the overall interaction with the Footpath Navigation Service should not exceed 1 second may be expressed using the following formula:

$$time(RECEIVED(navigationResult)\ Since\ SEND(navigationRequest)) < 1$$

Second, the Astro/Dynamo monitoring framework allows expressing properties and measure/aggregate information over a single execution of a process instance as well as over a set of instances; these are important properties if assumptions refer to certain statistical information (e. g., certain QoS aspects, such as performance or availability). Third, the framework allows for measuring the properties in the SBA's context using external services and components as a pluggable sources of information. This is an important capability in order to evaluate context assumptions. Details on the realization of this capability can be found in [15].

Consequently, the timed assumptions, behavioral assumptions, and certain contextual assumptions may be observed using the integrated monitoring framework. As for protocol assumption monitoring, we rely on the approach presented in [29]. The underlying idea is that a monitor observes an interaction with the corresponding service and compares the sequence of messages with the one defined in the protocol. If wrong messages arrive, the protocol is violated, and the violation is reported by the monitoring framework.

4.4 Step 4: Run-Time Verification

Concepts: If a violation of an assumption is detected by monitoring, we need to determine whether this violation actually leads to a failure in the SBA. To evaluate such failures we use the verification techniques described in Section 4.2 at run-time. An adaptation is only necessary if

the verification fails. As we have monitored the individual assumptions, we know the source of the problem, e. g. the assumption that was violated. Together with our traceability links (see again Figure 3), we can determine the affected requirement(s). Having analysed the impact of the service fault, the SBA operator can decide whether an adaptation is necessary or not. It is important to note here, that the adaptation can take effect before the SBA instance terminates, as the verification has determined the violation of the SBA requirements even before the whole workflow has been executed.

Implementation: We can use the same techniques as presented in Section 4.2. For example, if the protocol of the Navigation Service has changed (e. g., an error message is sent if the location specified by the user is unknown), it is possible to verify whether the whole composition is still deadlock-free. If the requirement is still satisfied, the adaptation is not needed.

In a similar way, from the re-verification of timed properties, it is possible to evaluate whether the time requirement is violated, and which of the violated assumptions is responsible for that.

4.5 Step 5: Triggering Adaptation

Concepts: If the requirement has found to be violated, the next activity is to identify the source of the problem and trigger appropriate adaptation actions. In our approach the identification of the problem source is straightforward, because any assumption is associated to the elements of the application model that rely upon those assumptions. This relation allows us to identify the component of the system that is subject to adaptation.

Implementation: A specific element of the SBA domain, such as constituent services and the context (e. g., connectivity, or user profiles) may be further associated with the appropriate adaptation actions. For example, the service protocol and SLAs may be associated with such actions as replacement of the service, re-design/re-compose the part of the orchestrating BPEL process that interacts with that service in order to accommodate to the new version, etc.

Note that the ability to precisely identify the critical elements of the domain allows us to identify adaptation actions that are the most appropriate in a given situation and, therefore, avoid redundant or harmful adaptations.

5. RELATED WORK

The closest relations to our approach may be found in the requirements engineering domain under the label requirements monitoring. Fickas and Feather for instance, describe in [12, 13] an approach similar to ours. In line with us, the authors argue, that assumptions should be identified during the requirements engineering phase of a software development project. The authors also argue that the system only operates correctly in case of valid assumptions. They conclude that assumption monitoring is beneficial to adapt the system (called remedial action in the papers) once a violation of an assumption is detected. However, Fickas and Feather's work is descriptive only, e. g. concrete techniques for monitoring assumptions and for linking them with the system to derive adaptation triggers are missing.

More recent work of Cohen et al. and Fickas et al. concentrate on the monitoring aspect. In the [9, 11] the authors describe a flexible method how to derive monitors based on requirements and assumptions, e. g. elicited and documented by the KAOS approach [10]. Because of the strong monitoring focus, the papers do not cover any aspect related to adaptation.

The idea of requirements monitoring is also used in the domain of Web services and service compositions, where a wide range of approaches have been proposed for monitoring services and service compositions. In particular, in [4, 5] the authors propose an approach for BPEL monitoring. The monitoring properties (functional and non-functional) are expressed as pre-post conditions on service invocations within the process specification. In [24] the authors also propose a framework for BPEL composition monitoring. These approaches, however, do not differentiate between assumptions and generic requirements, and therefore, cannot be directly applied for identifying the source of the problem and for triggering appropriate adaptations.

In [29] and [23] the authors exploit assumptions for automated service composition. In the former case a composed BPEL process is automatically built using assumptions on the protocols of the component services. Moreover, runtime monitoring matches the actual behaviors of the service composition against the assumptions expressed in the composition requirements, and report violations. In the latter case an OWL-S composition is constructed exploiting assumptions on SBA context expressed as logical constraints in the Semantic Web Rule Language (SWRL) [27]. The composition obtained in such a way satisfies the requirements (composition goals) if the assumptions hold. These approaches have as main goal the service composition using assumptions. While in the first case they present also a way to monitor the composition at run-time to find possible violations, the second presents only a way to compose services without monitoring them. Neither the first nor the second approach, however, address the issue of debugging the violation of the assumptions at run-time.

Regarding the approaches that aim at identifying the causes of the occurring deviations from requirements, in [1] authors propose a framework for Web Service orchestration, which employs diagnostic services to support a fine grained identification of the causes of exceptions (occurring during the execution of a composite service) and the consequent execution of effective exception handlers. This is achieved by defining a special infrastructure with Local Diagnoser services associated to each component service. These services generate diagnostic hypotheses over exceptions from the local point of view, while Global Diagnoser service aggregates these hypotheses to provide a global diagnosis of the occurred failure. With respect to our approach, this work focuses only on specific types of exceptions and faults; it does not evaluate the implication of these exceptions on the SBA requirements, and requires heavy instrumentation also on the side of the constituent services, which reduces the flexibility and dynamicity of the SBAs.

In [18] the authors propose a framework that exploits assumptions for the design and maintenance of software systems. At design time assumptions are defined over different modules of the system. The verification is exploited (1) to check that the component satisfies the assumptions and (2) to guarantee that the whole system behaves correctly (using "assume-guarantee" reasoning [16]). During software system evolution, if the software code changes, the assumptions are re-checked and the possible violation of the system

correctness is reported or new assumptions are generated. Differently to our approach, this framework deals only with a very specific type of properties (e. g., program code assertions) and leaves open the problem of how the changes are monitored.

6. DISCUSSION AND PERSPECTIVES

In this paper we have demonstrated how monitoring techniques can be beneficially augmented with verification techniques to support the adaptation of service-based applications. The basic idea of our approach is to start from explicitly documented requirements and assumptions. Assumptions address functional and quality properties of third-party services (e. g., as documented in service-level agreements). A verification step at design time ensures that the SBA fulfils its requirements under specified assumptions. During run-time, monitoring the assumptions allows detecting violations (e. g., service failures). A violation of SBA's requirements can then be determined by re-verifying the SBA given the violated set of assumptions. If that verification fails, an adaptation, to compensate for the violation of the assumptions, may be triggered.

It is important to note that this paper does not discuss whether an adaptation of the SBA needs to be realised. We only provide adaptation triggers. Each adaptation trigger indicates that a requirement of the running SBA instance will be violated. Wether this violation is worth to be addressed by an adpation and how this adaptation should be realised is subject to further research.

Our approach exploits formal verification techniques. By doing so, we limit our approach to those requirements and assumptions, which can be formally expressed. In addition, the verification of complex systems may take considerable resources so that it may not be feasible to use these techniques at run-time. Both issues are subject to future work.

The current discussion of our approach is limited to the service composition layer—in particular we do not address the infrastructure layer of the service-based application. However, this infrastructure is important for any quality of service attribute related to time. Especially, if the execution times for individual services are low, the time needed for the communication between services need to be taken into account. The design of our approach does not take this communication time into account and is, therefore, based on the assumption that these communication times are minimal compared to the execution time of the SBA's services.

Our approach foresees a verification at run-time to determine whether a service failure may lead to a violation of the SBA's requirements. However, even if an assumption is violated, this might not lead to a requirements violation and thus the verification might not fail. To save computational resources at run-time it would, thus, be desirable to define a minimal set of assumptions such that each violation of an assumption will lead to a failure in the SBA and, thus, eliminates the costly verification step at run-time. In the future we are going to investigate whether this approach for various types of assumptions and models would be feasible.

Furthermore, we argued that due to the identification of the problematic part in the SBA, the adaptation could be better tailored to the failure situation and is, thus, more efficient. Since the approach is based on the general concept of assumptions, it should also be possible to extend it to other types of assumptions, e. g., assumptions about users, devices, locations and other context factors and, therefore, to trigger an adaptation based on factors, which are outside the SBA's boundaries. In future work we plan to substantiate both claims by investigating the interplay between our approach and current adaptation strategies.

Lastly, we argued that assumptions are engineered during the design step. In reality, however, assumptions may also be derived during the verification step. If a verification fails at design time, this failure may be due to the fact that some assumptions were missing. Consequently, it would be very interesting to understand the interplay between requirements engineering and verification in order to derive assumptions, which fit the need of both techniques.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

7. REFERENCES

[1] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Fault tolerant web service orchestration by means of diagnosis. In V. Gruhn and F. Oquendo, editors, *EWSA*, volume 4344 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2006.

[2] M. Autili, P. Inverardi, and P. Pelliccione. Graphical scenarios for specifying temporal properties: an automated approach. *Automated Software Eng.*, 14(3):293–340, 2007.

[3] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-Time Monitoring of Instances and Classes of Web Service Compositions. In *IEEE International Conference on Web Services (ICWS 2006)*, pages 63–71, 2006.

[4] L. Baresi, C. Ghezzi, and S. Guinea. Smart monitors for composed services. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 193–202, New York, NY, USA, 2004. ACM.

[5] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In *ICSOC*, pages 269–282, 2005.

[6] L. Baresi, S. Guinea, M. Trainotti, and M. Pistore. Dynamo + ASTRO: An integrated approach for bpel monitoring. In *7th International Conference on Web Services (ICWS 2009)*, 2009.

[7] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology, and Philosophy of Science*, pages 1–1. Stanford Univ. Press, 1962.

[8] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2:2000, 2000.

[9] D. Cohen, M. S. Feather, K. Narayanaswamy, and S. S. Fickas. Automatic monitoring of software requirements. In *ICSE '97: Proceedings of the 19th international conference on Software engineering*, pages 602–603, New York, NY, USA, 1997. ACM.

[10] A. Dardenne, A. V. Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In *Science of Computer Programming*, pages 3–50, 1993.

[11] S. Fickas, T. Beauchamp, and N. A. R. Mamy. Monitoring requirements: A case study. In *ASE '02: Proceedings of the 17th IEEE international conference on Automated software engineering*, page 299, Washington, DC, USA, 2002. IEEE Computer Society.

[12] S. Fickas and M. S. Feather. Requirements monitoring in distributed environments. In *SDNE '95: Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments*, page 93, Washington, DC, USA, 1995. IEEE Computer Society.

[13] S. Fickas and M. S. Feather. Requirements monitoring in dynamic environments. In *RE '95: Proceedings of the Second IEEE International Symposium on Requirements Engineering*, page 140, Washington, DC, USA, 1995. IEEE Computer Society.

[14] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based verification of web service compositions. In *ASE '03*, pages 152–161. IEEE, 2003.

[15] C. Ghezzi and S. Guinea. Run-time monitoring in service-oriented architectures. In *Test and Analysis of Web Services*, pages 237–264, 2007.

[16] O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16, 1991.

[17] M. R. Hansen and Z. Chaochen. Duration calculus: Logical foundations. *Formal Asp. Comput.*, 9(3):283–330, 1997.

[18] P. Inverardi, P. Pelliccione, and M. Tivoli. Towards an assume-guarantee theory for adaptable systems. *Software Engineering for Adaptive and Self-Managing Systems, International Workshop on*, 0:106–115, 2009.

[19] M. Jackson and P. Zave. Deriving specifications from requirements: an example. In *ICSE '95: Proceedings of the 17th international conference on Software engineering*, pages 15–24, New York, NY, USA, 1995. ACM.

[20] R. Kazhamiakin, A. Metzger, and M. Pistore. Towards correctness assurance in adaptive service-based applications. In *ServiceWave 2008*, number 5377 in LNCS. Springer, 10-13 December 2008.

[21] R. Kazhamiakin, P. Pandya, and M. Pistore. Representation, verification, and computation of timed properties. *International Conference on Web Services*, pages 497–504, 2006.

[22] R. Kazhamiakin and M. Pistore. Static verification of control and data in web service compositions. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services*, pages 83–90, Washington, DC, USA, 2006. IEEE Computer Society.

[23] Z. Lu, S. Li, A. Ghose, and P. Hyland. Extending semantic web service description by service assumption. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 637–643, Washington, DC, USA, 2006. IEEE Computer Society.

[24] K. Mahbub and G. Spanoudakis. Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 257–265, Washington, DC, USA, 2005. IEEE Computer Society.

[25] S. Nakajima. Model-checking verification for reliable web service. *OOPSLA Workshop on Object-Oriented Web Services (OOWS 2002)*, 2002.

[26] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM.

[27] G. Newton, J. Pollock, and D. L. McGuinness. Semantic web rule language (SWRL), 2004.

[28] E. D. Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl. A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3-4):257–402, 2008.

[29] M. Pistore and P. Traverso. Assumption-based composition and monitoring of web services. In *Test and Analysis of Web Services*, pages 307–335, 2007.

[30] A. van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 5–19, New York, NY, USA, 2000. ACM.

[31] A. van Lamsweerde, E. Letier, and R. Darimont. Managing conflicts in goal-driven requirements engineering. *IEEE Trans. Softw. Eng.*, 24(11):908–926, 1998.

10 Gu, Q. ; Cuadrado, F. ; Dueñas J. C., Lago, P.: Architecture Views illustrating the Service Automation Aspect of SOA. In: Service Based Systems: Surveys by the S-Cube Project, Springer-Verlag, 2010, Ch. – accepted for publication

Architecture Views illustrating the Service Automation Aspect of SOA

Qing Gu[1], Félix Cuadrado[2], Patricia Lago[1], and Juan C. Dueñas[2]

[1] Dept. of Computer Science
VU University Amsterdam, The Netherlands
[2] Dept. de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Service-Oriented architecture (SOA) as an emerging architecture style has been widely adopted in the industry. Due to the heterogeneous, dynamic and open nature of services, architecting Service-Based Applications (SBAs) poses additional concerns as compared to traditional software applications. Hence, for SOA architects it is of great importance that their concerns are appropriately addressed in the architecture description. However, an effective and systematic way of documenting SOA design is currently missing. In this work, we focus on the service automation aspect of SOA. We carried out two large case studies to learn the industrial needs in illustrating services deployment and configuration automation, from now on service automation. As a result, we broke down service automation into three important sub-aspects, and we developed a corresponding set of architecture views (automation decision view, degree of service automation view and service automation related data view) that expresses the different concerns of stakeholders who share interest in service automation. This set of views adds to the more traditional notations like UML, the visual power of attracting the attention of their users to the addressed concerns, and assist them in their work. This is especially crucial in service oriented architecting where service automation is highly demanded.

Key words: Service-oriented Architecture;Service-based Application;Architecture view;Automation;Service management;Service deployment

1 Introduction

Service-oriented architecture (SOA) as an architectural style has drawn the attention from both industry and academia. SOA-based systems (i.e., Service-Based Applications or SBA) are constructed by integrating heterogeneous services that are developed using various programming languages and running on different operating systems from a range of service providers. Services are loosely coupled entities, often designed under open-world assumptions, distributed across organizational boundaries and executed remotely at their service providers' environment. They require a smoother transition from development to operation than traditional applications.

Consequently, the architecting of SBAs pose additional concerns as compared to traditional software applications. Some examples of these concerns include how to reason about a SOA design and how to represent the characteristics of SOA that the design delivers, how to architect the SBA to operate in an unknown environment, or how business processes can be supported by means of the collaboration of multiple services.

Clearly, traditional software engineering and architecting techniques, methods and tools are no longer sufficient to deliver SBAs, as they do not take into account specificities of services, such as the need for smooth transition from development to operation, the need to integrate third-party components, or the possibility to be hosted by a different organization. Therefore, it is necessary to propose new techniques that supplement the traditional models, enabling the capture at design time of all the relevant information about those new concerns and improving the usefulness of the architecture description.

We have chosen as the aspect under study the degree of automation of SBAs. The current situation is that the design decisions on whether a service can be possibility automated, its benefits and limitations, or the degree of automation are often left implicit in the architectural design and its description. Our goal in this paper is to make them explicit and to find a notation useful for this purpose, able to be understood for most stakeholders (including the user if relevant to the domain).

We carried out two large case studies to learn the industrial needs in illustrating services deployment and configuration automation, from now on service automation. As a result, we broke down service automation into three important sub-aspects, and we developed a corresponding set of architecture views (automation decision view, degree of service automation view and service automation related data view) that expresses the different concerns of stakeholders who share interest in service automation.

The first one (the decision view) conveys the decisions about service automation by making explicit which architecture constraints may impact the degrees of automation, and which services are affected by each constraint. The degree view -second one- shows the degree of service automation that the service flow is expected to achieve, but not the details on how to get it. Last, the automation-related data view contains explicit information about the generation, management and provision of additional input that are required from either human actors or policies.

In addition to constructing these views, we highlighted the added-value of the graphic notations we used. We argue that this set of views adds to the more traditional notations like UML, the visual power of attracting the attention of their users to the addressed concerns, and assist them in their work. Moreover, we also reflected on the relationship between the degree of automation and the granularity of services and the applicability of these views to SOA in general.

The reminder of the chapter is organized as follows. In Sec. 2, we provide some background information on architecture views and management systems for SBAs. In Sec. 3, we discuss the need of documenting SOA design decisions

and rationale in effective illustrations and present a set of concerns that we elicited from the case studies, which points out what needs to be illustrated in SOA architecture description. With these requirements in mind, we present the three service automation views in Sec. 4, 5 and 6 respectively. We highlight the power of visualization in Sec. 7 and we discuss our observations in Sec. 8. We conclude the chapter in Sec. 9.

2 Background information

2.1 Architecture views

The architecture of a software system should be documented with the purpose of capturing early design decisions, providing re-useable abstractions of software systems and enabling communication of the software architecture among stakeholders [1]. To produce relevant documentation for a software system, one has to decide what information needs to be documented and which notations or diagrams are suitable for representing this information. These decisions heavily depend on who is the target reader of the documentation.

A software system typically involves multiple *stakeholders* that have different concerns. For instance, the architect is concerned about the structure of the system; the project manager is concerned about the resources (e.g., cost, time, number of developers) needed for developing the system; and the developer has concerns about the implementation of the system. The architectural design of the system therefore should be documented in such a way that the concerns of each stakeholder are addressed.

Following the *separation of concerns* principle, software architects have already been using multiple views for years to represent the software systems of interest from multiple perspectives. These views facilitate the management of the complexity of software engineering artifacts. For instance, a structure view can be used to describe the construction of a software system (including e.g. components and connectors); while a data view can be used to describe the data flow between the components. By representing the architecture of the system in these two separate views, the software architect may focus on the construction design of the system by using the structure view, while the data manager may concentrate on the management of data by using the data view.

One of the original goals behind IEEE 1471 and ISO/IEC 42010 was to "establish a frame of reference of terms and concepts for architectural description" [2]. This frame of reference provides a basis for the community to develop a collection of views which addresses concerns that occur commonly across software projects. Practitioners may directly benefit from the application of these viewpoints in that they enable an effective architecture description.

However, the existing reusable views are limited in the sense that they address concerns that often appear in traditional software architectures. With the wide adoption of recently emerged software architecture styles (like SOA), additional concerns (often specific to the architecture styles) challenge the reusability

of the existing viewpoints. The lack of available views make practitioners face difficulties to find an effective way to illustrate any new characteristics introduced by any architecture style. As a result, views that enable the illustration of specific concerns introduced by modern software architecture styles are needed.

2.2 Management system for SBAs

Hegering [3] described the management of networked systems as the set of measures necessary to ensure the effective and efficient operation of a system and its resources, according to an organization's goals. In service-based applications the functionality is not provided by individual, monolithic elements, but is achieved by collaboration between multiple services. In order to get this collaboration, the management system must be aware of the participating elements, deploy and configure them if necessary. This is a complex process, because as the number of services grows, the possible combinations that must be considered by the management system increase exponentially. On top of that, the distribution of the participating elements over a computing network further complicates the process. Those are common characteristics for every SBA. However, they are not the only relevant factors. The domain-specific characteristics of each SBA, such as the characteristics of the services, the capabilities of the runtime resources, or the organization's business aspects must also be supported by the management system. It is clear that the potential variation of all the factors complicates defining a general solution for SBAs deployment and configuration.

In the field of systems management, there are two opposite approaches for controlling the deployment and configuration process: traditional management processes and autonomic management [4]. In traditional processes, a human administrator is continuously in control of the change process. He / She diagnoses the system, manually defines the required changes and controls every aspect of the execution. This approach is very costly and cumbersome, because it implies that every activity executed by the architecture must be performed or at least validated by a human actor. On the other hand, autonomic computing promotes to automate as much as possible the operation of the system. Ideally, completely automated closed control loops are implemented, where the system reacts automatically to a change in the environment, diagnoses its severity and implications and applies the required corrections in order to restore the environment functionality. This approach eliminates the bottleneck inherent to human operation, consequently improving scalability and efficiency of the management system.

Although autonomic control would be the most desirable approach, it is not always feasible to achieve it, because of either technical factors (e.g., a monitoring interface from a managed server does not provide information about service faults so a human administrator has to manually diagnose the incidences by inspecting the server and system logs) or, organizational aspects (e.g., manual control is preferred because the service update process is considered critical for the organization, so an automated system cannot have complete control over the process). For most cases an adequate balance between traditional and autonomic management will be the right approach. Management systems should pursue

the autonomic approach to the greatest extent possible, while respecting the requirements derived from the domain of application.

Supporting the diversity of managed services, operation environments and organizational aspects with the same management architecture demands a high level of flexibility, which is pushed forward adopting a service-oriented approach. Service orientation can offer great flexibility and agility so that the architecture can easily adapt to the characteristics of different environments with reduced required configuration. The Service Deployment and Configuration Architecture (from this point onwards SDCA) [5] is an example of such an approach, which is further described in Sec. 3.1.

3 The requirements for illustrating the automation aspect of SBAs

The SOA paradigm promotes creating new functionality from the dynamic combination of services provided by different stakeholders. A SBA can be viewed as a set of dynamically combined services.

While automation in a traditional software system refers to the degree to which the execution of the *software process* can be automated without human intervention, automation in SOA systems refers to the degree to which *services*, comprising the system of interest, can be executed automatically without any human intervention. While the two definitions are quite similar, due to a set of characteristics that differentiate SBAs from traditional software systems [6], in service-oriented development the decision on the degree of automation of each service is heavily influenced by (and has impact on) at least two quality attributes.

The first quality attribute is trust, i.e. confidence (especially from the users perspective) on the truth of what delivered or promised. SBAs are typically not fully controlled by the company: some integrated services execute in the domain of remote, dynamically determined service providers, and can be discovered and integrated at runtime. This means that if something goes wrong, malfunctions might decrease the satisfaction of ones customers, and hence influence the overall company business. Especially in traditional business domains (like banking and services to the public) the tendency is to develop applications with service-oriented technologies, but with the properties of old fashion software systems: low level of automation, static integration of services, no dynamic discovery and no dynamic composition. In the case of required interaction with third-party services, the requirements -both functional as non functional- of these and the penalties for failure, are governed by Business Level Agreements (BLA) or Services Level Agreements (SLA).

The second quality attributes is reliability, i.e. the ability of a software system to perform its required functions under stated conditions for a specified period of time [7]. By automatically integrating services during execution, reliability of the whole execution depends on various unpredictable factors, like the correct specification of the requirements of the services to be dynamically integrated,

availability of such services, or their correct execution. If third-party service discovery & composition is automated, the company does not have anymore full control on the software products delivered to its customers.

It is often claimed that SBAs have the agility to adapt to customer needs by automatically reacting to continuous changes in business processes. Consequently, the more services in a SOA system can be automated (i.e., do not need humans to make decisions for execution), the higher agility a SBA can achieve.

Users of highly automated SBAs clearly benefit from less human intervention and thus less labor costs. However, automating the execution of services and delivering agile and reliable SBAs is not always possible (as we explained above in the examples of trust and reliability) and poses additional concerns. Some of the concerns relate to the decisions on the degree of service automation; while some of the concerns are related to the realization of these decisions.

However, design decisions and their associated rationale on whether a service can be possibility automated, the benefits and limitations of automating a service, or how to automate a service are often either ignored or left implicit in the architectural design and its description. In spite of the evidence for the need of documenting design decisions and rationale in effective illustrations [8, 9] little work exists so far in the area of SOA [6]. This need has been further highlighted in the S-Cube analysis of the state of the art in [10], where one major challenge is in identifying and representing relevant concerns in SBA engineering, like monitoring, self-organization and adaptation. Viewpoints are mentioned as means to capture multiple perspectives on a given SBA. Though, they are meant to aid engineering of specific systems, whereas the corresponding architecture descriptions have not been sufficiently addressed yet. This motivated us to investigate what the stakeholders are concerned about with respect to service automation and how to address these concerns in the architecture description.

To answer this question, we analyzed the service automation aspect of the SDCA as well as two concrete industrial case studies where the SCDA has been applied to. The first case study (BankFutura) describes the deployment and configuration system of a banking organization. As for most enterprise systems, in this case the non-functional requirements such as the criticality of the delivered services, guaranteed performance levels, and organizational aspects are the dominating factors for driving the decisions on the degree of automation of the service execution flow. In the second case (HomeFutura) the implementation of SDCA provides the services of multiple third-party providers, which are presented to the end users through a service catalog, allowing them to select the functionality they require. While the same service execution flow is adopted in both cases, there are significant variations in the automation related aspects, due to the impact of their domain-specific and organization-specific constraints.

In the remaining of this section, we present an overview of the SDCA and its industrial case studies (BankFutura and HomeFutura), focusing on the concerns related to service automation that we have elicited from the cases. These concerns serve as the requirements for illustrating the automation aspect of SOA in the architecture description.

3.1 The Service Deployment and Configuration Architecture

The Service Deployment and Configuration Architecture (SDCA) is a flexible, service-oriented management architecture that can address the requirements of distributed, heterogeneous SBAs (a.o. dynamic discovery, dynamic composition, adaptation, runtime evolution). The management functions are provided by a set of services, which collaborate to identify the required changes to the environment in order to fulfill the SBA business objectives (a.o. service availability). SDCA Services are automated, reasoning over models representing the characteristics of the managed services and the runtime environment. Finally, in order to adapt to the domain-specific characteristics, the specific behavior of the services can be customized through the definition of policies that govern the decisions taken over the process.

The objective of the SCDA is to provision new functionality (in the form of services) by identifying and applying a set of changes to the managed environment. This function is achieved by an execution flow consisting of the combined invocation of nine deployment services, as shown in Fig.1.

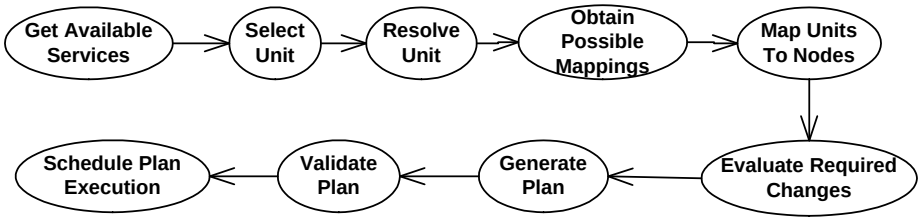


Fig. 1. The SDCA deployment service execution flow

A typical execution starts when an external change to the system is triggered (e.g. an updated version of a service has been released and must be deployed, or a hardware malfunction caused a server to stop working, and the affected services must be redeployed at another node). After it is decided that a change is necessary, the deployment service Get Available Units is invoked. This first step retrieves the complete list of units and services currently available. The second step is the deployment service Select Unit, where one of those available units is selected, in order to be deployed to the environment. The *unit selection criteria* will be provided to the service as an external input. After that, the deployment service Resolve Unit is invoked, where the deployment unit containing the service is analyzed, in order to find a closed set of units satisfying all their dependencies. There might be multiple candidate units satisfying one dependency (e.g. multiple units with minor, compatible versions) and for those cases a *criteria for selecting among them* must be provided as external input. Once the complete set of units that will participate in the operation has been identified, the deployment service Obtain Possible Mappings evaluates the available resources

from the container, and returns for each unit the potential nodes of the environment where those can be deployed. Starting with that input, the deployment service Map Units To Nodes decides on the final destination for each one of the involved units, according to external *distribution criteria*. After those mappings have been established, the deployment service Evaluate Required Changes compares the current environment status with the desired changes in order to obtain the set of required changes that must be applied to it (e.g. install selected deployment units if they are not currently running at the environment). Those changes are packed and sorted into a deployment plan in the deployment service Generate Plan, whose purpose is to ensure a correct execution of the list of changes, by adding restrictions to their execution order. Defined plans can either be instantly applied to change the environment, or can be temporarily stored at a change repository. Before being applied to the environment, plans must pass through the deployment service Validate Plan. This step checks that the automatically obtained plan is coherent with the environment state, and will obtain the desired result. Finally, the deployment service Schedule Plan Execution receives the accepted plan and schedules it for execution at some point in the future, which will be also determined by an external *schedule agenda*.

Some of the deployment services can be *completely automated* as they do not require any external intervention, such as Get Available Service and Obtain Possible Mappings; whereas some others cannot be completely automated as *external input* (i.e., additional input external to the service invocation flow)(e.g., distribution criteria, unit selection criteria) are required during the deployment process. The architect of SDCA provided services requiring external input with certain degree of flexibility, supporting two alternatives for their implementation. One of the solutions is to create a user interface so that a dedicated human actor can provide the required input to those services. Another solution is to formalize the necessary knowledge for providing the required input in terms of policies, which can be automatically consumed bt the deployment service. SDCA services with the format approach are called *semi-automated services* whereas the latter are called *policy-driven automated services*.

The stakeholders of the SDCA that are concerned about service automation include *the SOA architect*, who is responsible for defining a deployment service flow that can support the automated provisioning of services, adapting to the hardware characteristics of each environment; *the SOA manager*, who governs the design and implementation of the SDCA, and *the users of the SDCA*, who are often the SOA architects that intend to apply the SDCA to specific domains.

Being the designer of the SDCA, the SOA architect is mainly concerned about how to provide enough flexibility with the degree of automation, in order to allow adaptation of the flow to specific domain requirements. Additionally, the SOA architect is concerned about how to support the other stakeholders in terms of service automation. SDCA users are mainly concerned about how to customize the SDCA in such a way that domain specific constraints are fulfilled.

The complete list of concerns of each stakeholder is presented in Tab. 1, where each concern is described by its associated stakeholder, a description, and a concern ID.

Table 1. Concerns relevant to service automation in the SDCA

Stakeholder	Concern ID	Concern description
SOA architect	SDCACon1	Justify whether their decisions on the degree of service automation are reasonable.
	SDCACon2	Provide enough flexibility with the degree of automation, in order to allow adaptation of the deployment service flow to specific domain requirements.
	SDCACon3	Suggest policies that are required for assisting the deployment service flow.
SOA manager	SDCACon4	Trace, verify and control the decisions on the degree of service automation.
	SDCACon5	Gain an overview of the degree of service automation supported by the SDCA.
	SDCACon6	Gain an overview of the re-configurability of the SDCA.
SDCA Users	SDCACon7	Be aware of which deployment services are domain specific (hence customization is needed) and which ones are domain independent (hence no customization is necessary).

3.2 BankFutura: An application of the SDCA to an enterprise domaina

A Spanish banking company, called from this point on BankFutura, with several millions of clients over the world, and more than two thousand branches, renovates its services portfolio, which includes client services (internet banking, cashiers), internal services (for company workers at the bank offices) and B2B services for inter-bank transactions. As those services capture the company knowledge, they are internally developed and provided, with no third party dependencies. This is understandable, as they constitute the core of the company business and consequently must be under full control of the company. The company services have been architected following the SOA / BPM paradigm, in order to cope with the complexity.

The services runtime infrastructure that will replace the legacy systems and mainframes is composed by artifacts such as relational databases, JEE application servers, and BRM (Business Rule Managers) systems. Each artifact of the system is presented as a banking service, hiding its implementation details and providing a uniform high-level view. Banking services are published in directories and connected through an ESB (Enterprise Service Bus). The complete runtime infrastructure is dimensioned and defined beforehand, in order to support the strict non-functional requirements for the service operation, as well as adequately support the types of services that will provide the core banking functionality.

In BankFutura, every deployed banking service must be always available, respecting the requirements defined at the SLA, while dispatching the requests

from a potentially enormous number of consumers. Neither hardware and software malfunction, or denial of service attacks from ill-intended actors, should be able to disrupt the service operation, as service downtime would imply huge monetary costs. The stability of the banking system becomes one of the most important non-functional requirements.

Another non-functional requirement for the banking system is security. The exchanged information of banking services is very sensible, as it contains the financial status and personal data of the clients, so it is not only critical for their trust but also legally protected by the personal data confidentiality regulations. Because of that, it is fundamental to safeguard the security of the underlying systems, and provide complete logging and traceability of the performed operations. This way, change initiation, approval and execution must be registered and supported by the change management architecture, including a responsibility chain for any identified incidences.

In order to respect all these restrictions and facilitate at the same time system evolution, the BankFutura infrastructure is replicated into several, tiered environments (integration, pre-production and production) which present a balance between agility of changes and criticality. The complexity and pre-defined structure of the deployment and configuration architecture justifies that BankFutura employs specialized staff, such as Environment administrators, for watching over the runtime health, diagnosing malfunctions and controlling the execution of planned changes to the environment, despite the costs the bank is incurring by keeping this staff.

The stakeholders who are concerned about service automation in BankFutura include the SOA architect, the banking deployment plan creator, the environment administrator, and the service deployment manager.

The SOA architect is responsible for applying and customizing the SDCA so that the resulting deployment service flow can support the complete provisioning of the released services to the several tiered environments of the infrastructure of the company. He/She is mainly concerned about automating the deployment services as much as possible (moving away from handcrafted scripts) while at the same time integrating human control and responsibility over the complete process.

The banking deployment plan creator and *the environment administrator* are both deployment actors in the banking deployment service flow. The former is responsible for creating a deployment plan which will provide the desired functionality when applied to the environment; while the latter is responsible for the correct configuration of the managed infrastructure and the selection of the right physical node for each newly deployed service, taking into account the additional resources consumption by each new service. Both of them are mainly concerned about how to perform their roles in the banking deployment service flow.

The deployment manager is in charge of supervising the execution of banking deployment service flow and ensuring that the deployed banking system is aligned with the business objectives of BankFutura.

The detailed concerns of each stakeholder are listed in Tab. 2, where each concern is presented with its associated stakeholder, a description, and a concern ID.

Table 2. Concerns relevant to service automation in BankFutura

Stakeholder	Concern ID	Concern description
SOA architect	BankCon1	Justify whether their decisions on the degree of service automation are reasonable given the specific constraints in the BankFutura.
	BankCon2	Understand what specific constraints affect each deployment services and how each constraint influences the degree of service automation.
	BankCon3	Analyze the possible alternatives on the degree of service automation in order to evaluate how the deployment service flow can react to changing requirements or constraints.
The deployment manager	BankCon4	Trace, verify and control the decisions on the degree of service automation.
	BankCon5	Gain an overview of the degree of service automation in the BankFutura deployment service flow.
	BankCon6	Ensure that the environment administrator and banking deployment plan creator carry out the assigned tasks as expected and are able to trace responsibility in case an error occurs.
	BankCon7	Ensure the availability of required policies that are required for the deployment services in time.
	BankCon8	Ensure that the required policies for the deployment process are aligned with the organizational goals and regulations.
Environment administrator	BankCon9	Ensure the stability of the managed environment after executing the deployment services.
	BankCon10	Define the role and responsibility in preparing policies.
	BankCon11	Define the role and responsibility in the deployment service flow.
Banking deployment plan creator	BankCon12	Select the right physical node for each newly deployed service, taking into account the reasons that led to the initial definition of the environment topology
	BankCon13	Know which services (and what version of the service) must be made available in each environment.
	BankCon14	Define the role and responsibility in the deployment process.

3.3 HomeFutura - An application of the SDCA to a personal domain

The service aggregator of this case study (called from this point on HomeFutura) wants to offer subscribers a large catalog of services that can be consumed from the devices available at the digital home. The digital home is the house of the near future, an always connected entity, provided with network and devices to access Internet resources. It allows users to consume a wide range of services; multimedia entertainment services, surveillance services, e-learning services or consumer electronics control, just to mention a few. Services are provisioned over the Internet and accessed through multiple home devices. The specific hardware elements that will be available are controlled by the end users, which can dynamically decide to acquire additional equipment.

The ultimate goal is to create an environment that benefits end users, service providers, and service aggregators. End users should be able to browse all available services and subscribe to those they are interested in, automatically accessing them without technical skills. Service providers develop and offer services to be consumed by the end users. Service aggregators are the point of contact with the users, managing their subscription, interacting with the service providers, and ensuring correct and seamless service provisioning.

In this case study the deployment architecture plays a fundamental role. In order for those services to be available, it is necessary to execute deployment and configuration activities over the home infrastructure. The general characteristics of the environment are similar to the previous case, with the required operations consisting of managing services running over a distributed, heterogeneous infrastructure. However, the specific characteristics of this scenario lead to a different solution. In contrast with the case study of BankFutura, environment stability is not the dominating constraint. This is because the domain is personal, and the services are consumed and used without warrantee of performance nor agreed Quality of Service expressed through BLAs or SLAs. On top of that, guaranteeing stability is much harder, because of the high degree of uncertainty about the specific equipment that will be available at every moment.

Instead, the fundamental goal in HomeFutura is being able to provide the end user with a seamless experience in the process of acquiring new services. The user is not concerned about the technical details behind the services or the installation process. Those aspects must be correctly managed by the architecture, while the user is only informed about the relevant information, like functionality, or pricing.

The stakeholders who are concerned about service automation in HomeFutura include the SOA architect, the service aggregator, and the end user.

The SOA architect is responsible for defining the architecture of the digital home service deployment system by applying the SDCA. The main concern consists of how to provide a flexible deployment system that is able to adapt to the available infrastructure at each home while at the same time hiding all the technical details from the end user.

The role of a *service aggregator* is to manage the service catalog available to the different users and handle the signed contracts with service providers, ensuring that the portfolio of services offered to the users can have all their technical dependencies correctly satisfied. The service aggregator is also responsible for providing selection policies which determine what providers / versions for the services can be accessed by each different client.

The end user consumes the available services offered by HomeFutura, demanding as much variety in the services catalog as possible. The end user is mainly concerned about the simplicity of the process of accessing the desired functionality.

The detailed concerns of each stakeholder are listed in Tab. 3, where each concern is presented with its associated stakeholder, a description, and a concern ID.

Table 3. Concerns relevant to service automation in HomeFutura

Stakeholder	Concern ID	Concern description
SOA architect	HomeCon1	Justify whether the decisions on the degree of service automation are reasonable given the specific constraints in HomeFutura.
	HomeCon2	Understand what specific constraints affect each deployment service and how each constraint influences the degree of service automation.
	HomeCon3	Analyze the possible alternatives on the degree of service automation in order to evaluate how the deployment service flow can react to changing requirements or constraints.
	HomeCon4	Design a highly automated deployment process, with a minimal requirement on human intervention
The deployment manager	HomeCon5	Trace, verify and control the decisions on the degree of service automation
	HomeCon6	Gain an overview of the degree of service automation in the HomeFutura deployment service flow.
	HomeCon7	Ensure the policies that are required in the deployment process are ready in time
	HomeCon8	Ensure the policies that are required in the deployment process are aligned with the organizational goals and regulations
Service aggregator	HomeCon9	Define the role and responsibility in preparing policies.
End user	HomeCon10	Participate in the deployment process the simplest way possible

3.4 Summary

From the analysis of the SDCA and two industrial case studies, we observed that service automation is especially important during design and is relevant to multiple stakeholders in that we identified a considerable number of service automation related concerns. Being considered, designed and implemented, however, those concerns have not been explicitly addressed in the architecture description.

Instead, the current architecture description of the SDCA (as well as the two case studies) addresses the service automation related concerns in a very abstract way. For instance, it is stated that the SDCA provides a flexible solution that can be easily customized in various domain applications. However, the information about how flexible the solution is, how easy the solution can be applied, and how to customize the SDCA in specific domains is lacking. Hence, there is a need to find an effective way to illustrate how the concerns related to service automation are addressed in the architecture description.

In other words, we face the questions of *what information should be documented in the architecture description* and *how to document it in an effective way so that the stakeholders can easily understand it*. To answer the first question, we synthesized the concerns listed in Tab. 1, Tab. 2, and Tab. 3. The reason for doing so is that we noticed that a reasonable numbers of concerns are overlapping and demanding for the same type information. For instance, the concerns with ID SDCACon1, BankCon1 and HomeCon1 are all about justifying the decisions on service automation but in different cases (hence overlapping); and concerns with ID SDCACon3, BankCon7, BankCon10, BankCon13, HomeCon7, Home-

Con9 all demand for illustrating the information that is related to generate and access policies.

After the synthesis, we identified eight main concerns that are representative for the complete set elicited from the SDCA and its two case studies. *Decision on the degree of automation* covers all the concerns that are related to the decisions on service automation and their justification ; *Reconfigurability in terms of automation* covers all the concerns related to alternatives on the degree of service automation; *The impact of architecture constraints on the degree of automation* covers all the concerns related to domain-specific constraints; *Degree of automation* covers all the concerns related to the degree of automation a SBA can achieve; *Accountability* covers all the concerns related to the responsibility of stakeholders; *The preparation of policies* covers all the concerns related to the readiness of policies; *The specification of policies* covers all the concerns related to the content of policies; and *Human participation* covers all the concerns related to human actors with regards to their involvement in the deployment service flow.

Further, we noticed that some of these main concerns are inter-related. More specifically, the first three main concerns are about decisions, alternatives and constraints, which form a cause-effect-rationale relation. As such, we decided to address all these concerns using a **decision view**. The next two main concerns are about the degree of automation resulted from the design and the impact of such a degree on the execution of the deployment service flow. As such, we decided to address these concerns using a **degree view**. The last three main concerns are about the policies and human participation for enabling different degrees of service automation. Since policies and input from human actors can both be considered as data, we decided to use a **data view** to address the concerns. Hence the automation decision view, degree of service automation view and service automation related data view illustrate the service automation aspect for the architecting of SDCA.

The mapping between the elicited concerns, synthesized concerns and views for addressing these concern is presented in Tab. 4.

4 The automation decision view

The automation decision view is designed to illustrate all the decisions that have been made on the degree of service automation, the rationale behind them, and the impact of domain specific constraints on the decisions. With these requirements in mind, we created a set of graphic notations for constructing the automation decision view, as no other notation in the literature fits to our purposes. These graphic notations are presented in Fig. 2.

In this figure, services are represented by ovals; the three ones in the first column represent the three different degrees of service automation that have been decided. Moreover, they also indicate that alternative degrees of service automation are not feasible or reasonable. The services in the second column represent a decision has been made or left open among alternative degrees of

Table 4. Mapping between the elicited concerns, synthesized concerns and views

View	Main concern	Concerns in the SDCA	Concerns in BankFutura	Concerns in HomeFutura
Automation decision view	Decision on the degree of automation	SDCACon1, SDCACon4	BankCon1, BankCon4	HomeCon1, HomeCon5
	Reconfigurability in terms of automation	SDCACon2, SDCACon6	BankCon3	HomeCon3
	Impact of architecture constraints on the degree of automation	SDCACon7	BankCon2	HomeCon2
Degree of service automation view	Degree of automation	SDCACon5	BankCon5	HomeCon4, HomeCon6
	Accountability	-	BankCon6, BankCon11, BankCon14	HomeCon10
Service automation related data view	The preparation of policies	-	BankCon10, BankCon7	HomeCon7, HomeCon9
	The specification of policies	SDCACon3	BankCon8, BankCon9, BankCon11, BankCon13	HomeCon8
	Human participation	-	BankCon11, BankCon14	HomeCon10

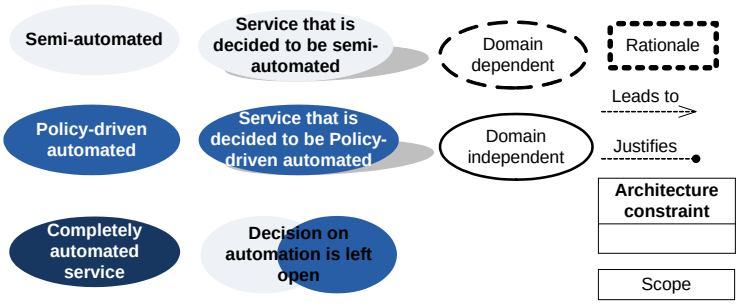


Fig. 2. The graphic notations for the automation decision view

service automation. These services indicate that they can be re-configured to an alternative degree of service automation if necessary. These two sets of notations are meant to address the concerns of *decision on the degree of automation* and reconfigurability in terms of automation.

The two notations in the third column indicate the dependency between a degree of service automation and a specific domain. The notations in the last column are used to illustrate the relation between decisions, architecture constraints, and associated rationale, as well as the scope of services where architecture constraints may have impact on. These two set of notations are meant to address the concerns of *the impact of architecture constraints on the degree of automation*.

4.1 The automation decision view for the SDCA

The automation decision view for the SDCA is presented in Fig. 3. This view aids the SOA architect in taking design *decisions on service automation* by making explicit which architecture constraints may impact the degrees of automation, and which services are affected by each constraint.

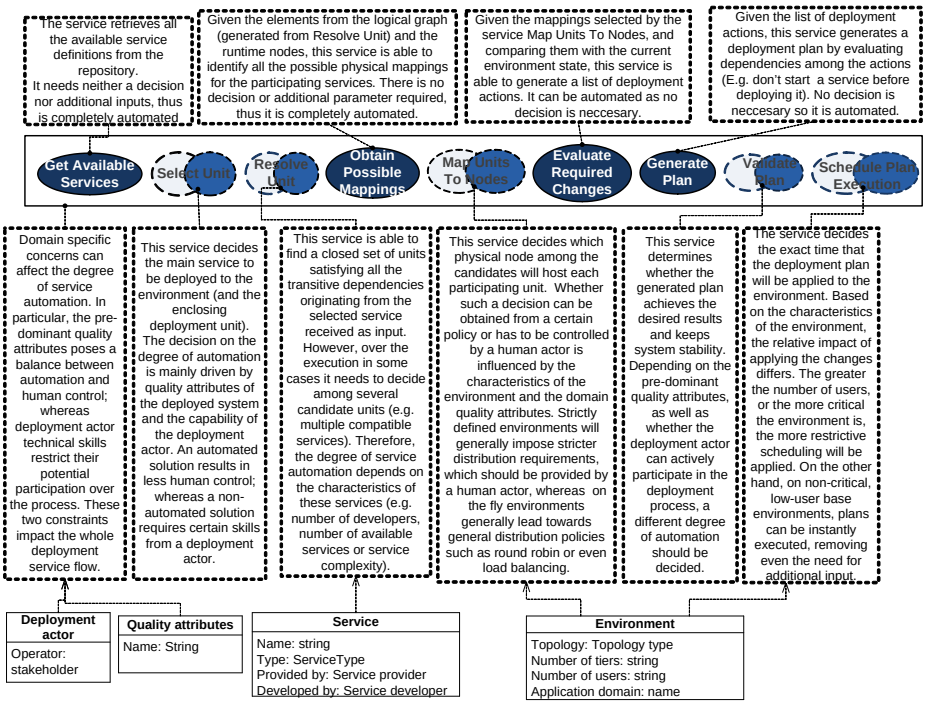


Fig. 3. The automation decision view for the SDCA

This view differentiates the degrees of service automation that are domain-dependent from the ones that are domain-independent. More specifically, the four services that are completely automated are circled with a straight edge line, indicating that they are domain-independent in terms of automation (SDCACon7). The other five services, however, are designed being both semi-automated and policy-automated. As such, the SDCA offers its users the flexibility to decide on which degree of service automation to be configured for specific domains, based on domain-specific constraints, such as quality attributes or characteristics of the execution environment (SDCACon7).

As an example of domain-independent decision, the deployment service Generate Plan automatically sorts a list of operations, ensuring they are executed in a correct order. The execution of this service only requires the input provided by the previous deployment service Evaluate Required Changes. Hence, the deployment service Generate Plan can be completely automated, independent with any domain specific constraints.

As an example of domain-dependent decision, the service Map Units to Nodes decides the physical distribution of the participating services, among a list of potential mappings provided by the service Obtain Possible Mappings. Depending on the specific domain characteristics, the criteria for making those decisions will be different, as well as the relevance of this decision (ranging from any solution is acceptable to only one distribution is correct). Because of those factors, this service has been designed with flexibility on its degree of automation.

From these examples, we can see that the SOA architect can use this view to explain why some of the services have been designed to be completely automated while others have been designed for both semi-automated services and policy-driven automated services (SDCACon1) and the SOA manager is able to use this view to trace, verify and control these decisions (SDCACon4).

Highlighting the links between the architecture constraints and the decisions, this view facilitates the SOA architect to show the flexibility of adapting the SDCA to specific domain applications. It is obvious from the view that the services whose decisions on service automation are left open, require further re-configuration when architecture constraints become specific (SDCACon2, SDCACon6). The users of the SDCA also benefit from this view by being aware of the impact of certain architecture constraints on the degree of service automation.

For instance, the deployment service Select Unit aims at selecting the main service to be deployed to the environment (and the enclosing deployment unit). The decision on the degree of automation is mainly driven by quality attributes of the deployed system and the technical capabilities of the deployment actor. An automated solution results in less human control; whereas a non-automated solution requires certain skills from a deployment actor.

To give another example, the deployment service Map Units To Nodes decides which physical node among the candidates will host each participating unit. Whether such a decision can be obtained from a certain policy or has to be controlled by a human actor is influenced by the characteristics of the envi-

ronment and the domain quality attributes. Strictly defined environments will generally impose stricter distribution requirements, which need to be provided by a human actor, whereas the environments defined on-the-fly generally lead towards programmatic distribution policies such as round robin or even load balancing.

4.2 The automation decision view for BankFutura

When applying the SDCA to BankFutura, the four completely-automated services whose automation state is domain-independent require no further decisions and hence remain being completely-automated. On the other hand, the five services implemented as both semi-automated and policy-driven automated in the SDCA require further decisions on re-configuration based on the BankFutura specific architecture constraints. The outcome of those decisions is illustrated in the automation decision view for BankFutura, presented in Fig. 4.

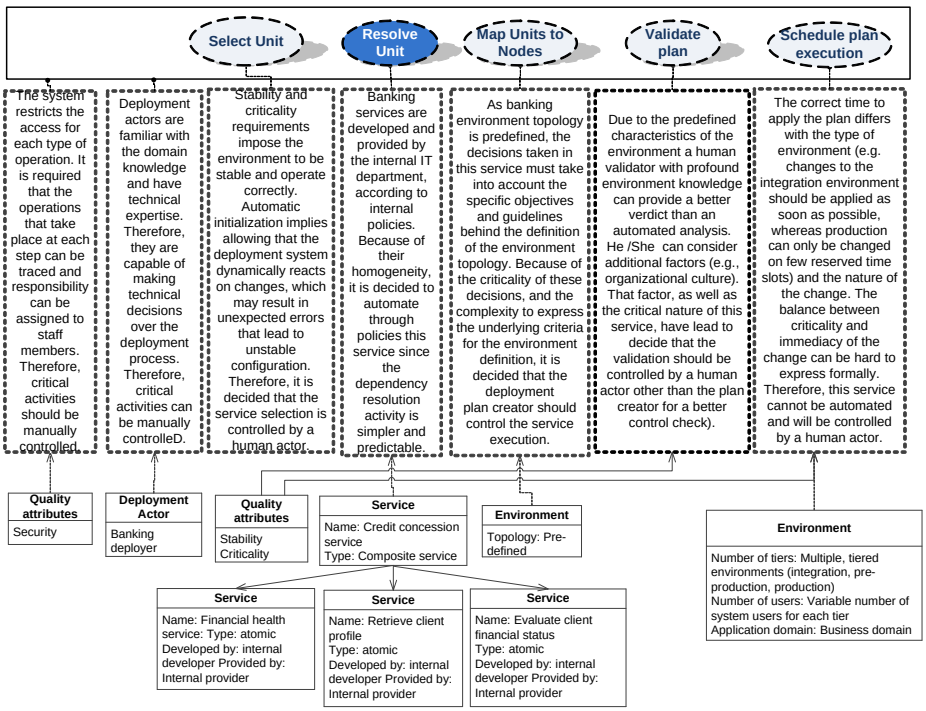


Fig. 4. The automation decision view for BankFutura

The view shows how the domain non-functional requirements such as criticality or reliability limit the degrees of service automation that BankFutura actually can operate with, in spite of the advantages of a completely automated service

execution flow. As a result, the SOA architect decided to semi-automate most of the services to guarantee a certain degree of control over the deployment process (`BankCon1, BankCon2`). The only service that was decided be policy-driven automated is the deployment service Resolve Unit. An exception was made in that case because BankFutura services are developed and provided by the internal IT department, satisfying the business needs of the organization and are developed according to internal policies. This suggests a simpler and predictable dependency resolution activity and hence it was decided to be automatically driven by policies rather than human actors.

As we can see the requirements of criticality or reliability as well as their impact on the degree of service automation are highlighted in the view (`BankCon2`). Not only the SOA architect can use this view to justify their decisions on the degree of service automation satisfying the requirements of criticality or reliability, but also the deployment manager can use it to trace, verify and control the decisions that the SOA architect made (`BankCon4`).

Explicitly documenting the rationale for the decisions on the degrees of service automation also enables the analysis of the possible alternatives on the degree of service automation, allowing to evaluate how the deployment service flow can react to changing requirements or constraints (`BankCon3`). If BankFutura intends to reconfigure the degrees of service automation, it will be useful to know the automation alternatives and the trade-off among them. For instance, the services presented in Fig. 4 are marked with a shadow if they can be either semi-automated or policy-driven. Although the services have been decided to implemented with either of the degrees of service automation, they could be reconfigured to another degree if the organization deemed it necessary (e.g. after a time of operation the organization increased its trust in the automation capabilities of the BankFutura, and opted to increase the degree of automation for a more efficient operation).

4.3 The automation decision view for HomeFutura

Specialized from the degree of automation view for SDCA (presented in Fig. 5), this view presents the implemented degree of automation for HomeFutura services.

Similar to BankFutura, the degree of automation view for HomeFutura (presented in Fig. 5) does not display the four completely automated services that are domain independent and need no further decisions. This view emphasizes the decisions on the degree of automation for the other five services that are domain-dependent, as well as the domain specific constraints that lead to these decisions.

Whereas in BankFutura environment stability is the dominant constraint, HomeFutura aims at providing end users the flexibility to experience new services available on the network. Consequently, HomeFutura opted for a deployment solution with higher degree of automation, not only because end users are not capable of providing technical input to the deployment process but also because

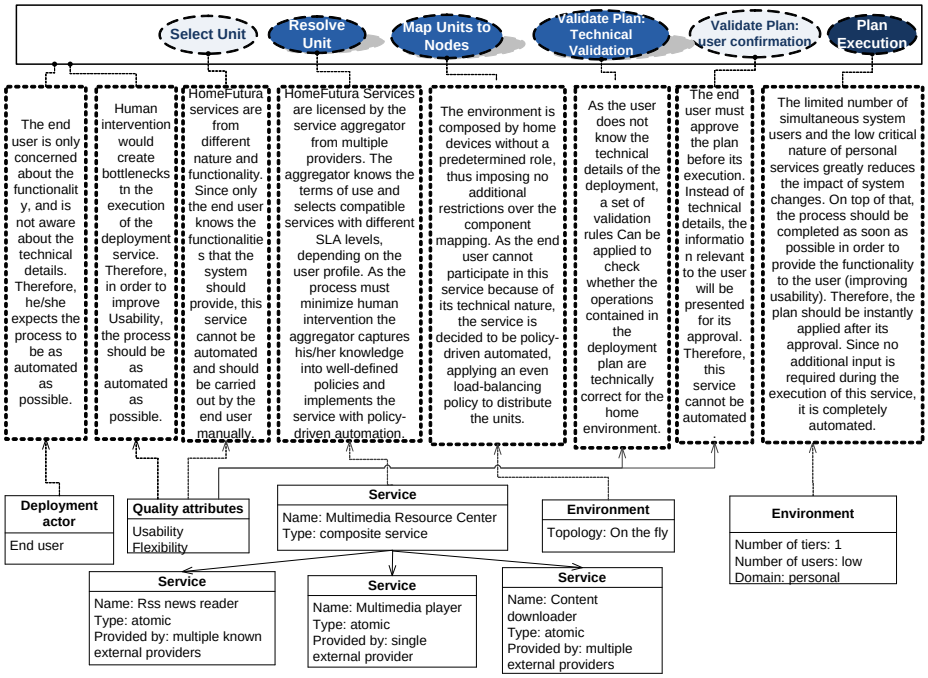


Fig. 5. The automation decision view for HomeFutura

agility in the execution is required. The view presents this rationale and shows its link with the architecture constraints(HomeCon2 influencing the decisions.

As a result, only two services that require input from end users are semi-automated. The service Select Unit requires the end users to select the services that they would like to experience; and the other service Validate Plan requires the end users to approve the execution of the deployment plan (e.g., the cost of news service, the changes to multimedia player device). The rest of the services in the deployment process are all automated and do not require human intervention . Similar with the decision view for BankFutura, this view enables the SOA architect to justify the decisions and supports the deployment manager in governing the decisions (HomeCon1, HomeCon5).

It is worth to note that three services are marked with a shadow, which indicates that a choice has been made between semi-automation and policy-driven automation for these services. Although policy-driven automation has been chosen for the current deployment solution for HomeFutura, the shadow reminds the architect that this service could be re-configured to be semi-automated if needed (HomeCon3). For instance, after a time of operation it turns out the service Map Units To Nodes does not provide the most optimal mapping of the selected services to devices (due to e.g. too complicated dependency graph that Map Units To Nodes cannot interpret correctly or incomplete policy that is not able to provide sufficient information for the mapping), the architect could decide to let a technical expert decides the mapping and hence make Map Units To Nodes to be semi-automated. However, involving another deployment actor (other than the end user) in the process would cause that the user cannot instantly start to experience the new service, having to wait for the required input from the technical expert. This way, usability and agility of HomeFutura would be challenged.

5 The degree of service automation view

Whereas the automation decision view emphasises the domain-specific constraints that lead to the decisions on the degree of service automation, the degree view focuses on the decided *degrees of service automation* for the service execution flow. As a result, only the degree of service automation that the service flow is expected to achieve is relevant in this view, while the details of how the decisions are made are irrelevant and should not be presented in this view.

As denoted by the graphic notations presented in Fig. 6, the degrees of automation are graphically rendered by the darkness of the color assigned to each service: the darker is the color, the higher is the degree of automation. In addition, human actors are associated to semi-automated services with the purpose of highlighting who are expected to provide input to which services. The sequence between services indicates the order in which the deployment services are invoked. With this additional information, the period during which human intervention is (and is not) required becomes explicit. By illustrating that ex-

ternal inputs are expected to be provided by *whom* and *when*, this view also addresses the *accountability*.

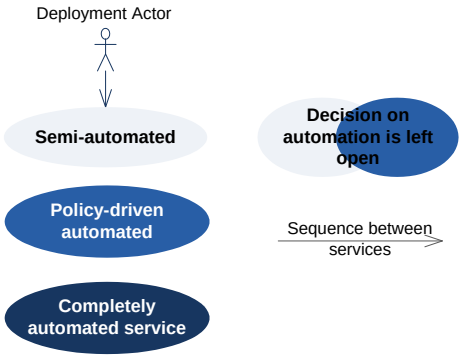


Fig. 6. The graphic notation for constructing the degree of automation view

5.1 The degree of service automation view for the SDCA

Applying the graphic notations presented in Fig. 6, we constructed the degree of service automation view for the SDCA shown in Fig. 7.

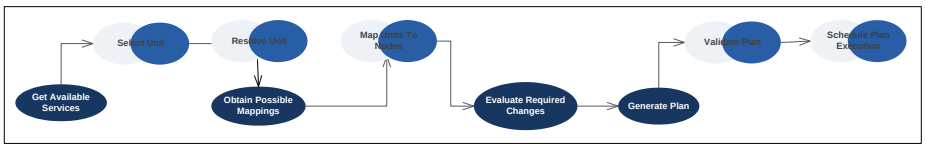


Fig. 7. The degree of automation view for the SDCA

Using this view, the SOA manager can gain an overview of the degree of service automation supported by the SDCA, as a result of the decisions illustrated in Fig. 3. More specifically, two different degrees of automation are designed for the SDCA (`SDCACon5`). Deployment services that do not require additional information are completely automated, while those needing external information are designed to retrieve the external information either from human actors or from policies.

5.2 The degree of automation view for BankFutura

Analogous to the SDCA, we also used the described notation to construct the degree of service automation view for BankFutura, as it can be seen in Fig. 8.

The deployment manager can see from this view the three degrees of service automation designed for the BankFutura (`BankCon5`). More specifically, four services are completely automated, one is policy-driven automated and four are semi-automated. In other words, this view shows that nearly half of the services require human intervention, meaning that the automation degree of the deployment process for BankFutura is relatively low.

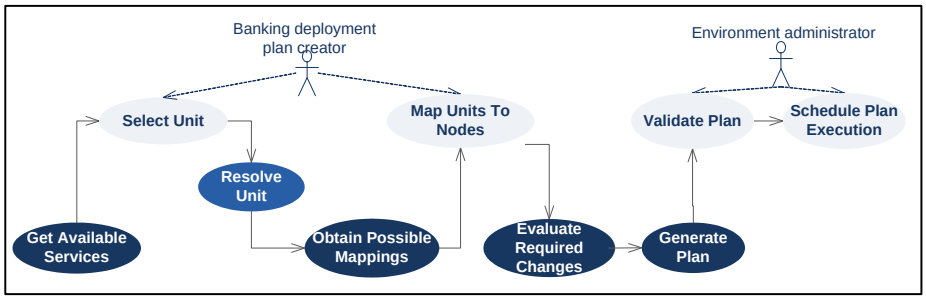


Fig. 8. The degree of automation view for BankFutura

In addition, as the semi-automated services are associated with a banking deployment plan creator and an environment administrator, the manager may use this view to trace the responsibility of these two human actors who are expected to provide input during the services execution (`BankCon6`).

Similarly, this view points out directly for the banking deployment plan creator and environment administrator which services are expecting their input. As shown in Fig. 8, the role of the banking deployment plan creator is associated to the deployment services Select Unit and Map Units To Nodes, indicating that as soon as the deployment service flow is initiated and the banking deployment creator should be prepared to first make a selection on the available units and later on to establish mapping between selected units and physical nodes (`BankCon14`). On the other hand, the environment administrator is only involved in the validation and execution of the deployment plan (`BankCon11`).

5.3 The degree of automation view for HomeFutura

Similar with the degree of automation view created for BankFutura, Fig. 9 shows the designed degree of automation for each service in the deployment process of HomeFutura.

The view visually highlights the fact that a higher degree of automation has been designed for HomeFutura as compared to BankFutura (`HomeCon4, HomeCon6`). As shown in Fig. 9, most of the services are illustrated with dark color (indicating that the services can execute without any human intervention); while two services are in light color (indicating that human intervention is needed).

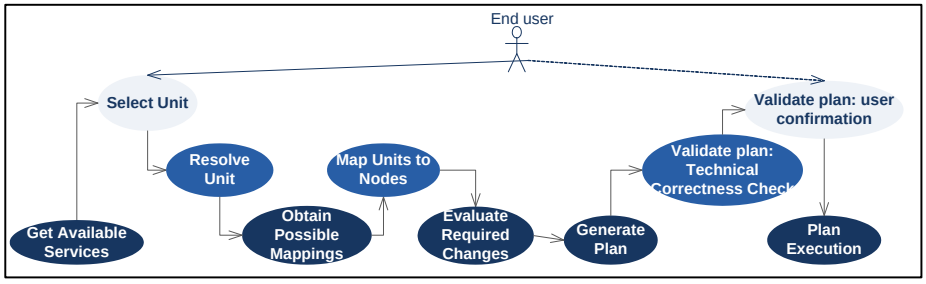


Fig. 9. The degree of automation view for the HomeFutura

The degree of service automation view for HomeFutura can also be used to explain to the end users how they are expected to participate in the deployment process. End users can see hwo their participation consists of selecting the home services that they would like to experience using service Select Units and eventually to agree on the corresponding costs of consuming these services by using service Validate Plan. This way, it can be seen how the end users are presented only the relevant information for them, while the low level, technical details are hidden(`HomeCon10`).

6 The automation-related data view

While the *degree of service automation view* highlights the degree of service automation designed for the deployment service flow, the *automation-related data view* details the design from the data perspective. This way, the questions related to the generation, management and provision of additional input (from either human actors or policies) can be answered by the automation-related data view.

The graphic notations that we created to construct the automation-related data view is presented in Fig. 10. Besides the notations for the three degrees of service automation, we distinguish the guidelines/rules from formalized policies. While both guidelines/rules and formalized policies are relevant to service automation, the former are used by the deployment actors to drive the decision and the latter are directly accessed by deployment services to achieve policy-driven automation.

The most right hand side of Fig. 10 shows the graphic notation denoting the relationships between elements in the automation-related data view. More specifically, a deployment actor *is responsible for* providing an input; such input *assists the execution* of semi-automated services; guidelines / rules guide the provision of such an input; formalized policies directly *assists the execution* of policy-driven automated services; and *sequence between services* is also denoted. Given these details on the relationships between policies, deployment actors, and deployment services, the deployment actors can tell which services are expecting

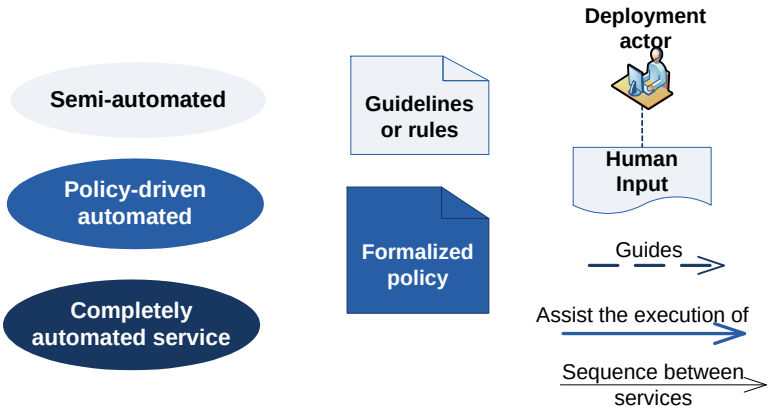


Fig. 10. The graphic notation for constructing the automation-related data view

what information from them. Moreover, it explicitly points out which organizational guidelines or rules should this information comply with. In this way, the deployment actors can be prepared to transform this organizational knowledge to their input to services, hence facilitating *human participation*.

In addition to the graphic notation, we also constructed a table template (shown in Tab. 5) for listing the policies that are relevant to service automation in the deployment service flow. This table aids *the specification of policies* in presenting all the information relevant to the policies in a structured manner. As such, this table also aids the *preparation of the policies*.

Table 5. The template for automation-related policy table

Policy ID	Policy name	Policy Description	Associated service	Controlled by	Type of format

6.1 The automation-related data view for the SDCA

As the SDCA is a reference deployment management system, it does not define concrete deployment actors or policies, as they will be specialized in specific applications. However, in order to guide the application of the SDCA, the SOA architect is concerned about identifying the required policies for providing the additional input to the deployment service flow. For this reason, we constructed the automation-related policy table, using the template presented in Tab. 5.

The *automation-related policy table* (presented in Table 6) provides detailed information about all the policies, including the ones for guiding human actors and the ones for policy-driven automated services. Using this table, the SOA architects in specific domains can gain an overview on what type of policies

might be relevant and should be prepared, as well as which format should they be expressed when applied to the SDCA (`SDCACon3`).

Table 6. The automation-related policy model for the SDCA

Policy ID	Policy name	Policy Description	Associated service	Controlled by	Type of format
P01	System requirements	Describes the functionality (services) that the target environment must provide	Select Unit	-	Formal/Text
P02	Unit selection policy	Provides criteria for selecting among multiple candidate units that satisfy the same dependency	Resolve Unit	-	Formal/Text
P03	Unit distribution policy	Provides criteria for selecting the physical place of the environment where each deployment unit will be installed	Map Units To Nodes	-	Formal/Text
P04	Plan Validation Rules	Defines a set of checks that can identify fatal errors in the plan definition or potential risks expressed as warnings	Validate Plan	-	Formal/Text
P05	Environment update policy	Controls at what periods of time plans can be applied to the environment	Schedule Plan Execution	-	Formal/Text

6.2 The automation-related data view for BankFutura

Applying the graphic notation presented in Fig. 10, we constructed the automation-related data view for BankFutura (shown in Fig. 11). This information is complemented with the BankFutura policy table, presented in 7)

From this view the two human actors, the *banking deployment plan creator* and the *environment administrator*, can see what types of information they are expected to provide to which services during the service execution flow (`BankCon11, BankCon14`). In addition, they can see which organization policies (or guidancerules) can be referred in order to provide the required information (`BankCon9, BankCon11, BankCon12, BankCon13`).

We will use an example scenario to illustrate how `BankCon10` is supported by the data view, guiding the participation of the *deployment plan creator* in the service *Map Units to Nodes*. In an specific environment, the environment design document mentioned in the data view informs that only one server from the environment infrastructure is configured in the network firewall to be remotely accessible from the outside; the remaining elements being protected from outer clients. In this case, by analyzing that information, the human will decide to assign units containing final services (which must be remotely accessible) to the visible server, whereas the remaining elements will be distributed over the other elements, regardless of whether those servers might also be technically capable of hosting the same types of services.

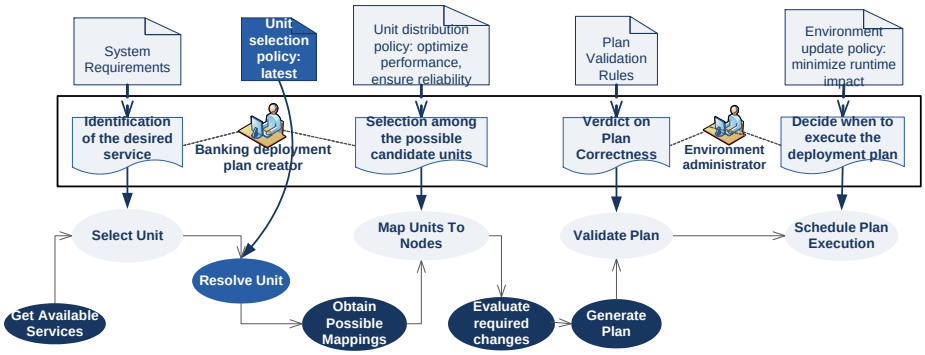


Fig. 11. The automation-related data flow view for BankFutura

Fig. 11 shows that the policy-driven automated service is explicitly linked to the corresponding policy. For instance, service Resolve Unit is linked to *Unit selection policy* indicating that the criteria for selecting among multiple candidate units defined by the policy directly influences the output of Resolve Unit. Explicitly visualizing the dependency between the services and their corresponding policies, the automation-related data flow model aids the deployment manager in obtaining an overview on when certain policies are required and which services require them during the deployment service flow (`BankCon7`).

Complementary to the data view, the automation-related policy table further details each policy presented in Fig. 11 by noting its description, the role that is in charge of it and the type of format for documenting the policy. This table aids the preparation of policies as it can be used as a check list for the deployment manager to assign tasks to some specific personnel, making sure that the policies are in place and are expressed in the right format before the execution of the deployment process (`BankCon7`).

6.3 The automation-related data view for HomeFutura

Applying the graphic notation shown in Fig. 10, the automation-related data view for HomeFutura is presented in Fig. 12. From this view, the end users can see that they are only required to initiate the deployment process when they want to experience new services and confirm the operation when the selected services are ready to be deployed. More importantly, the end users will be confident in providing this information as the data view shows that the former is based on their own functional requirements and the latter on their own non-functional requirements. With limited involvement in the deployment process, the end user has full control over the selection of new home services and the acceptance of any associated cost (`HomeCon10`).

As the deployment process for HomeFutura has much higher degree of automation than our previous case, it is more critical that the policies are in place before the deployment process starts as compared to the case of BankFutura.

Table 7. The automation-related policy model for BankFutura

Policy ID	Policy name	Policy Description	Associated service	Controlled by	Type of format
PB01	System requirements	Describes the business processes that must be supported by the environment	Select Unit	Managers	Textual Document
PB02	Unit selection policy: Latest version	Selects the unit with the most recent version among the potential candidates	Resolve Unit	SOA Architects	Formal/SQL Sorting Query
PB03	Unit distribution policy	Defines the rationale behind the environment definition and the quality levels to be sustained by the deployed services	Map Units To Nodes	SOA Architect	Textual Environment design / SLA Document
PB04	Plan Validation Rules	Checks whether the plan is coherent with the current state of the environment	Validate Plan	Environment Administrator	Formal/RETE-based rules + Guidelines Textual Document
PB05	Environment update policy: Minimize runtime impact	Controls the reserved time slows for applying changes, and avoids overlap in the execution of concurrent changes	Schedule Plan Execution	Environment Administrator	Environment Operation Guidelines Textual Document + Environment Calendar

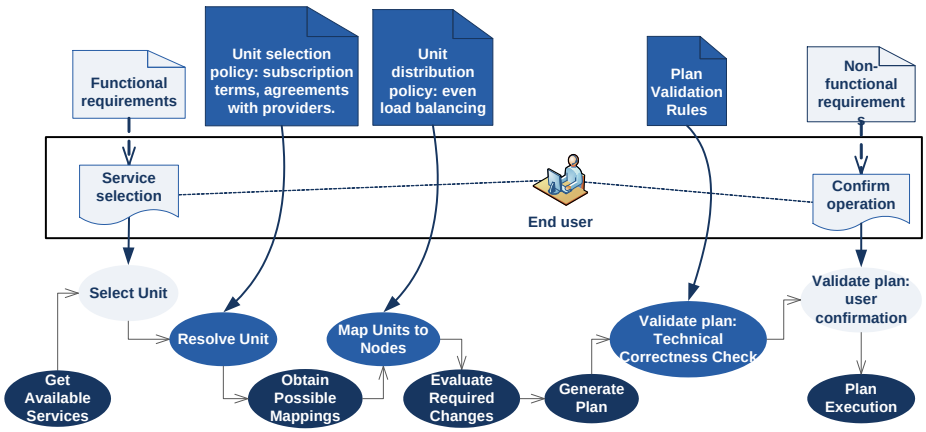


Fig. 12. The automation-related data flow view for HomeFutura

In BankFutura, there is only one policy-driven service requiring direct access to its associated policy. The other semi-automated services require the associated policies to be well documented enough so that the deployment actors are able to provide the required input. In HomeFutura, as shown in Fig. 12, there are three policy-driven services, which means that the three associated policies should all be accessible before the deployment process starts (HomeCon7).

Complementary to the data view, the automation-related policy table for HomeFutura presented in Tab. 8 provides detailed information about the policies illustrated in the data view. From this table, the service aggregator can see that he/she is the main role who is in charge of the formalized policies (`HomeCon9`).

Services available to HomeFutura come from different service providers, after being licensed by the service aggregator. The aggregator knows about their use licenses and selects compatible services with different SLA levels, depending on the user profile. With the aggregator defining policies for solving dependency conflicts based on the user profile, the technical knowledge of the aggregator is transferred to formalized documents that the three deployment services can directly access to. This not only enables these services to be policy-driven automated, but also ensures the alignment between them and technical requirements that are specific to HomeFutura (`HomeCon8`).

Table 8. The automation-related policy model for HomeFutura

Policy ID	Policy name	Policy Description	Associated service	Controlled by	Type of format
PH01	Functional Requirements	Functionality that must be provided from the HomeFutura platform to the end user	Select Unit	End User	Undocumented knowledge
PH02	Unit selection policy	Selects among the candidate units based on the agreements with service providers and the user subscription terms	Resolve Unit	Service aggregator	Subscription Terms Document, Agreements with providers document
PH03	Unit distribution policy: Even load balancing	Distributes the affected units over the home environment attempting to balance the load on the computing nodes	Map Units To Nodes	Service aggregator	Formal: Linear Programming coding
PH04	Plan Validation Rules	Checks whether the plan is coherent with the current state of the environment and the subscription terms of the end user	Validate Plan: Technical Correctness Check	Service aggregator	Subscription Terms + Formal/ RETE-based rules
PH05	Non-functional requirements	Checks whether the consumption of the selected service complies the non-functional requirements of the end user	Validate Plan: User Confirmation	End user	Undocumented knowledge
PH06	Environment update policy: Instant execution	Instantly applies the plan to the home environment	Schedule Plan Execution	Service aggregator	Formal: code

7 The power of visualization

Visualization is a common technique to convey abstract information in intuitive ways. Representing information in terms of (a set of) graphics often more easily draws readers' attention and improves understandability, as compared to pieces of text. That is why visualization has been considered as one of the most effective ways for communication. In the same vein, effectively applying the technique of visualization in architecture description improves the communication between stakeholders; in our experience the range of stakeholders involved in services engineering is broader than in the development of traditional applications, rendering the usage of visualization techniques as a key to get effective communication. On the other hand, the usage of diagrams (such as UML) well-known in the field of software architecture, while useful for technical stakeholders, results to be difficult to understand and reason with for non-technical ones (such as users in HomeFutura).

When constructing the service automation views, we consciously design the graphic notation to make the views intuitively understandable and to hold readers attention steady. Some of these graphic notation have been commonly used in architecture description, like using a symbol of person to stand for a human actor, or using a symbol of document to stand for a policy.

In addition to these commonly used notation, we created a set of color schema representing the different degrees of service automation. The motivation behind this schema is that from the human perception point of view, objects with dark color often make one feel heavy in weight (and easy to sink), whereas objects with light color make on feel light (as easy to float). As shown in the views, the degree of automation is graphically rendered by the darkness of the color assigned to each service: the darker the color, the higher s the degree of automation. This representation resembles an iceberg immersed in the sea: only the top (white is the lightest color) is visible (i.e. the user is aware of the service and manually participates in its execution), while the deeper the iceberg is sunk in water, the lesser visible it becomes (i.e. accessible by users, or in other words, increasingly automated).

In the same vein, the graphic notation for the policies inherit the same color schema. As shown in Fig. 10, the policies for providing guidance for deployment actors are in light color; whereas the policies for assisting the execution of policy-driven automated services are in dark color. As such, from the color of the policies shown in Fig. 11, and 12, the reader can have the perception of the correspondence between the degree of service automation and policies. In addition, in both the degree of service automation view (shown in Fig. 7, 8, and 9) and automation-related data view (shown in Fig. 11, and 12, the completely-automated services with dark color "sink" at the bottom, implying that they are loosely coupled with human actors. Whereas the semi-automated services with light color "float" at the top, implying that they are tightly coupled with human actors. As such, these visualization techniques enable the views become self-explaining.

8 Observation

By studying the SDCA, and in particular its two industrial cases, we have identified a set of concerns that are particularly relevant to service automation. By constructing views to illustrate the service automation aspect, we gained insight into the way in which service automation has been designed under different contexts. In addition, during the course of this work, we made several observations.

First, we noticed that the degree of service automation is also relevant to the level of service granularity, which was not foreseen in this study. In the design of SBAs, the appropriate level of granularity of services is often regarded of great importance and challenging. The alignment between business and IT is often the (only) main driver for service identification due to the benefits it might bring [11], such as the ease of comprehension of the design, the increase of potential reuse, just to name a few. Since service granularity in nature does not share common interests with service automation, it was not identified as one of the concerns to be addressed by the views. However, in this work we noticed that the service granularity, to certain extent, is also influenced by service automation, which again makes the relevance of service automation to SOA more evident.

In the case of HomeFutura, we noticed that the deployment service Validate Plan was decomposed to two services, one is Validate plan: Technical Correctness Check and another is Validate plan: user confirmation. The main driver for this decomposition is that Validate Plan consists of two types of validation that can be implemented with two different degrees of automation. The technical validation aims to check whether the operations contained in the deployment plan are technically correct for the home environment. This can be done automatically, provided that the policy *Plan Validation Rules* is available. Whereas the user validation aims to get approval from the end users if they agree with the non-functional attributes associated to the select home service. Since the end users are the only ones that have the knowledge on their own preferences and these preferences vary from person to person, it is not feasible to embrace this knowledge into a policy and it has to be controlled directly by the end users. As a result, the deployment service Validate plan: Technical Correctness Check validates the deployment plan from a technical perspective and is designed to be policy-driven automated; whereas the deployment service Validate plan: user confirmation validates the deployment plan from a user perspective and is designed to be semi-automated.

In the case of BankFutura, the deployment service Validate Plan was not decomposed although it also consists of the technical validation and user (system) validation. Similar with HomeFutura, *Plan Validation Rules* can be formalized as a policy that Validate Plan can directly access. The difference lies in the fact that the non-functional requirement of the system is known by the environment administrator and hence can also be formalized as a policy. In this way, Validate Plan can be designed as policy-driven automated, accessing *Plan Validation Rules* that consists of both the rules for technical validation and the non-functional requirements.

From these two examples, we can see that the identification of the deployment services or the level of service granularity is not only driven by the business functionalities that they represent, but also influenced by the degree of service automation. Despite the benefits that the business-IT alignment may achieve, an architect sometime would decompose a coarse-grained service into multiple fine-grained services due to different service automation requirements. The result of decomposition might lead to a SBA with higher maintainability and adaptability in terms of service automation but tightly-coupled services and decreased reusabiliy. As such, in the design of SBAs, an SOA architect has to make a trade-off between the alignment with business functionalities and the level of service automation.

The second observation we made is on the applicability of the views to architecture descriptions of SBA in general. As explained by Shull et. al [12], the sources of variability may influence the result, such as the types of projects for which a technique is effective. For this reason, we did an analysis on the variability of the domain and the type of architecture that we studied.

More specifically, the design of the SDCA aims at providing a reference architecture for service configuration and management while the same time focuses on its applicability in industrial domains. We also studied the application of the SDCA in an enterprise domain and a personal domain with completely different characteristics. The difference between these domains contributes to the variability of this study. Although the concerns elicited from the SDCA and its two case studies are somehow different and represent domain-specific interests, addressing these concerns in architecture description demands for similar types of information. When illustrating all these types of information in terms of the same set of graphic pictures and tables (or views), we are able to show that all the concerns identified from each case have been addressed in the corresponding architecture design. As a result, we are confirmed that the views can be applied to three different domains.

However, the concerns that we identified are all related to the SDCA, both its own design and its applications in different industrial domains. The lack of variability in terms of the type of architecture that we studied might threat the validity of the views in illustrating the service automation aspect of SOA in general. For this reason, we plan to replicate the study by analyzing the service automation aspect of various types of SOA in our future work.

9 Conclusion

In this chapter we have studied the different degrees of automation suitable for services configuration and deployment on different domains (business and home). While the initial goal was just the development of a system able to perform these functions -a Services Deployment and Configuration Architecture or SDCA- we discovered that the architectural concerns were affected by the specific domain of application it was to be used for; in fact there are several quality attributes

that must be covered, but the balance between trust and reliability for example, is specific to the domain.

The key contribution of this paper is the identification of three views that structure and ease elicitation and documentation of stakeholders' concerns. The application of these three views onto the bank and the home domain case studies clearly reflects the differences on the degree of automation for a similar set of basic functions (provided by the services); with a lower degree of automation at the bank domain when compared to the home domain. The notation we have used for the description of views and decisions is simplified with respect to available notations. This allows for a better representation of the concepts involved in the architectural decision making by stakeholders, while remaining intuitive even for non-technical ones. The expression of usually implicit architectural knowledge allowed us getting a hint on the relationship between the degree of automation and the granularity of services. Also, the usage of the same description technique across domains revealed commonalities between them.

The results obtained seem promising, but in order to better capture the wide variability of service automation we plan as future work to to apply the same process to additional Service-Based Applications, as well as applying the approach to several more unconnected domains. This way, it would also be interesting to validate whether different SBAs belonging to the same domain share specific constraints, which affect their decisions on the degree of automation the same way.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley (2003)
2. Emery, D., Hilliard, R.: Updating ieee 1471: architecture frameworks and other topics. In: WICSA '08: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), IEEE Computer Society (2008) 303–306
3. Heinz-Gerd Hegering, Sebastian Abeck, B.N.: Integrated management of networked systems: concepts, architectures, and their operational application. Morgan Kaufmann Publishers Inc (1998)
4. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer Magazine, IEEE **36**(1) (January 2003) 40–49
5. Ruiz., J.L., Dueñas, J.C., Cuadrado, F.: Model-based context-aware deployment of distributed systems. Communications Magazine, IEEE **47**(6) (June 2009) 164–171
6. Gu, Q., Lago, P.: On service-oriented architectural concerns and viewpoints. In: 8th Working IEEE/IFIP Conference on Software Architecture (WICSA), Cambridge, UK, IEEE (2009)

7. ANSI/IEEE: Standard glossary of software engineering terminology, std-729-1991. ANSI/IEEE (1991)
8. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In: 2nd International Conference on the Quality of Software Architectures (QoSA). (2006)
9. Ali Babar, M., Dingsoyr, T., Lago, P., van Vliet, H., eds.: Software Architecture Knowledge Management: Theory and Practice. Springer (jul 2009)
10. Andrikopoulos, V., Bertoli, P., Bindelli, S., Nitto, E.D., Gehlert, A., Germanovich, L., Kazhamiakin, R., Kounkou, A., Pernici, B., Plebani, P., Weyer, T.: State of the art report on software engineering design knowledge and survey of HCI and contextual knowledge (2008)
11. Heuvel, W.J.v.d., Yang, J., Papazoglou, M.P.: Service representation, discovery, and composition for e-marketplaces. In: Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS 2001). Volume Lecture Notes In Computer Science; Vol. 2172., Springer-Verlag London, UK (2001) 270284
12. Shull, F., Carver, J., Vegas, S., Juristo, N.: The role of replications in empirical software engineering. Empirical Software Engineering **13**(2) (2008) 211218