Grant Agreement N° 215483

Title:	*Knowledge extraction of service usage*
Authors:	*CNR, TUW, SZTAKI*
Editor:	*Fabrizio Silvestri (CNR)*
Reviewers:	*Osama Sammodi (UniDuE)*
	Elisabetta di Nitto (PoliMi)
Identifier:	*Deliverable #PO-JRA-2.3.7*
Type:	*Deliverable*
Version:	*1.0*
Date:	*30 June 2011*
Status:	*Final*
Class:	*External*

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).

Management Summary

This deliverable is aimed at summarizing the joint research in WP-JRA-2.3. related to knowledge extraction from service usage. The work is focused on methodologies to extract knowledge from service/Web logs and possible applications of them in order to enhance service/Web applications. Results are presented in four published papers and one technical report that constitute the core contribution of this deliverable. The work is positioned within the Integrated Research Framework (IRF, WP-IA-3.1), internal WP-JRA-2.3 research architecture and overall WP-JRA-2.3 goals and visions.

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
Universit Claude Bernard Lyon	France
University of Crete	Greece
Universidad Politcnica de Madrid	Spain
University of Stuttgart	Germany
University of Hamburg	Germany
Vrije Universiteit 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/results/deliverables/`

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.

- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with todays limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-Cube materials are available from URL: `http://www.s-cube-network.eu/`

Contents

List of Figures

Chapter 1

Foreword

According to the Description of Work (Grant Agreement for: Network of Excellence Annex I Description of Work) this deliverable is aimed at documenting the ways a service/web log can be mined in order to derive useful knowledge. In particular, this deliverable presents different ways of using knowledge extracted from Service Oriented Architectures (SOAs) event logs and Web search engines log and different applications of such knowledge. The rationale of focusing our interest on these techniques within the SOA context is motivated by the fact that they have proven to be effective in other domains (e.g. Web search engines).

The joint research work in WP-JRA-2.3 is composed of two tasks: T-JRA-2.3.1: Infrastructure Mechanisms for the Run-Time Adaptation of Services and T-JRA-2.3.2 Service Registration and Search. The current deliverable is related to T-JRA-2.3.2. As stated in the Description of Work, this deliverable "will report on studies about the knowledge extracted from the activities in online services. The deliverable will also report on the techniques studied and used to perform the knowledge extraction task. As one of the case studies we will use logs from search engines activities to extract knowledge regarding users activity". Therefore, it aims at summarizing the investigations related to extracting knowledge from event logs collected by complex software systems.

The joint work is focused on applying several state-of-the-art data mining algorithms to event logs provided by S-Cube partners. In order to fully present the possible outcomes resulting from the application of these techniques in the SOA domain we describe, as our running case study, how log mining has proven to be effective in enhancing the overall performances of complex software systems such as Web search engines. Results of the research work are presented in four published papers and one technical report and constitute the core contribution of this deliverable. The work is positioned within the Integrated Research Framework (IRF, WP-IA-3.1), internal WP-JRA-2.3 research architecture and overall WP-JRA-2.3 goals and visions.

Chapter 2

Introduction

Data is everywhere. Computer systems keep track of activities of users in the form of log files. Ranging from system logs on Web servers to logs collected by large-scale service based applications, this type of data represents a goldmine of knowledge that, once extracted, can help the stakeholders of the whole system to understand better if, and how, the application can be improved. To this aim, data mining consist of a set of techniques aiming at extracting patterns from large data sets by combining methods from *statistics* and *artificial intelligence* with *database management*. With recent tremendous technical advances in processing power, storage capacity, and inter-connectivity of computer technology, data mining is seen as an increasingly important tool by modern business to transform unprecedented quantities of digital data into business intelligence giving an informational advantage.

Service-centric systems are said to be flexible and dynamic. To support this flexibility, event processing mechanisms can be used to record which events occur within the system. This includes both basic "service events" (e.g., service is created) and complex events regarding QoS (e.g., average response time of service X has changed) and invocations (e.g., service X has been invoked), supporting complex event processing. Users can subscribe to various events of interest, and get notified either via email or Web service notifications (e.g., WS-Eventing). Such notifications may trigger adaptive behavior (e.g., rebinding to other services). Service Oriented Architectures (SOAs) are thus complex infrastructures consisting of thousands or millions of service interacting together in order to achieve complex operations (tasks). Service invocation logs are file tracing the interactions between services. As in other contexts, data mining techniques can be thus applied in order to derive useful knowledge. Such knowledge can be spent in order to enhance both effectiveness and efficiency of the overall infrastructure.

The same approach within other fields like, for example, the Web domain is proven to be effective. The knowledge extracted by means of data mining techniques from query logs (files containing the interactions of the users with the search engine) is the first way a search engine improve its performances in terms of effectiveness and efficiency. In this deliverable we thus investigate how useful knowledge can be extracted from service logs and possible ways of applications within the SOA context. In order to do that as one of the case studies we will use logs from search engines activities to extract knowledge regarding users activity.

2.1 Connections to the Integrated Research Framework

The Integrated Research Framework (IRF) [1] defines four views (other aspects of the IRF, like research challenges, questions and results are being developed simultaneously with this deliverable and omitted.)

The *Conceptual Research Framework* organizes the joint research activities by providing a high-level conceptual architecture for the principles and methods for engineering service based applications, as well as for the technologies and mechanisms which are used to realize those applications. The work presented in this deliverable is clearly related to the "Service Composition and Coordination" and "Service Infrastructure" domains in the horizontal classification whereas addresses issues of "Adaptation",

"Monitoring" and "Quality Definition, Negotiation and Assurance" of the cross-cutting vertical categorization.

The *Reference life-cycle* complements the static view of the conceptual research framework. It is composed of two main cycles: one corresponds to the classical application design, deployment and provisioning while the second one corresponds to the run-time perspective, including monitoring and adaptation. The work presented in this deliverable corresponds to both of the two cycles. The classical analysis and design could benefit of the techniques proposed here for adaptation cycle by investigating techniques aiming at enhancing the analysis and design phase. Furthermore, the run-time perspective exploits these techniques for detecting problems and changes. The techniques presented in this deliverable could also be used for identifying possible adaptation/monitoring strategies and enacting them.

The *Logical Run-Time Architecture* unifies all runtime mechanisms into a coherent framework. The work reported in this deliverable is related to nearly all components of the logical architecture: Monitoring Engine, Adaptation Engine, Negotiation Engine, Runtime QA Engine and Resource Broker.

The *Logical Design Environment* is complementary to the run-time architecture and its purpose is to provide a context where to place the envisioned techniques and mechanisms that support the analyst and designer during the whole SBA's lifecycle. Our work is related to modeling and verification. The techniques presented in this work could help SBA designers as they could reveal possible frequent patterns contained in historical data that could be exploited within new activities.

2.2 Connections to the JRA-WP-2.3 research architecture

Research work in WP-JRA-2.3 is driven by the Work Package vision that structures the research work internally. Figure 2.1 illustrates the overall research architecture of WP-JRA-2.3: research on service infrastructures is comprised in three threads, Service Discovery, Service Registries and Service Execution. Orthogonally different approaches are separated in three layers.

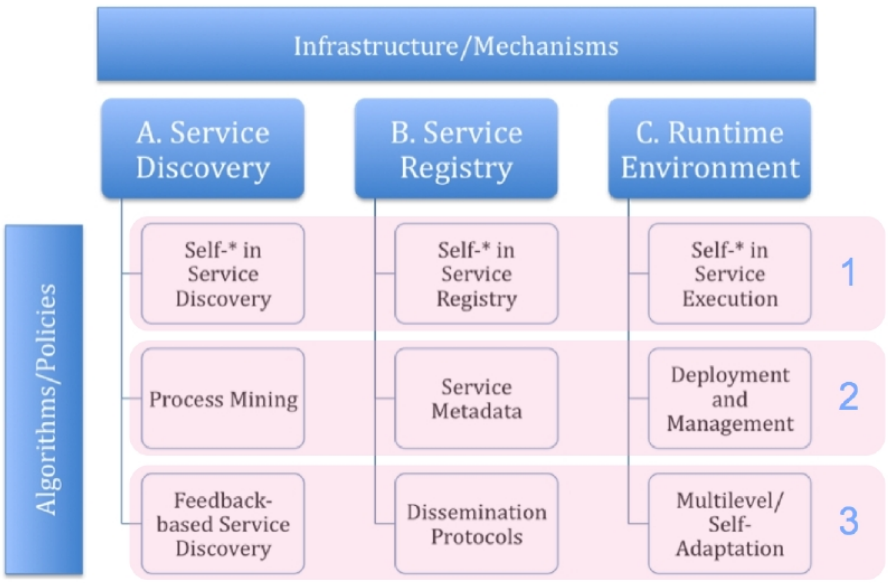

Figure 2.1: WP-JRA-2.3 Research Architecture.

- Service Discovery Thread (A) - Service discovery is a fundamental element of service oriented architectures, services heavily rely on it to enable the execution of service-based applications. Novel discovery mechanisms must be able to deal with millions of services. Additionally, these discovery mechanisms need to consider new constraints, which are not prevalent today, such as

Quality of Experience requirements and expectations of users, geographical constraints, pricing and contractual issues, or invocability.

- Service Registry Research Thread (B) - Service registries are tools for the implementation of loosely-coupled service-based systems. The next generation of registries for Internet-scale service ecosystems are emerging, where fault tolerance and scalability of registries is of eminent importance. Autonomic registries need to be able to form loose federations, which are able to work in spite of heavy load or faults. Additionally, a richer set of metadata is needed in order to capture novel aspects such as self-adaptation, user feedback evaluation, or Internet-scale process discovery. Another research topic is the dissemination of metadata: the distributed and heterogeneous nature of these ecosystems asks for new dissemination methods between physically and logically disjoint registry entities, which work in spite of missing, untrusted, inconsistent and wrong metadata.
- Runtime Environment Research Thread (C) - There is an obvious need for automatic, autonomic approaches at run-time. As opposed to current approaches we envision an infrastructure that is able to adapt autonomously and dynamically to changing conditions. Such adaptation should be supported by past experience, should be able to take into consideration a complex set of conditions and their correlations, act proactively to avoid problems before they can occur and have a long lasting, stabilizing effect.

In alignment with the lifecycle aspect of the Integrated Research Framework, the presented work – due to its nature -, is related mostly to runtime activities hence, to research topics in thread A in Figure 1. While topic A2 is directly covered topic C1 also presented progress. Service registries and runtime environments could be enhanced by the techniques presented in the following. In particular, some aspects of B2 and C2 are covered.

Chapter 3

Description of Data Analyzed

Modern complex software systems save traces of their activities to proper files called "logs". Such files could store "events" regarding the whole infrastructure (like in Service Oriented environments) or real "activities" performed by users interacting with the infrastructure. A possible example of the latter case could be represented by logs of queries coming from Web search engines.

In this deliverable we focus on these two different types of logs: "event logs" coming from Service Oriented Architectures and "query logs" coming from Web search engines. We now describe in more details the main characteristics of an "event log" and a "query log".

3.1 SOA Event Logs

Events and complex event processing [11] (CEP) are frequently used tools to document and track the lifecycle of applications in various domains. For instance, in the business domain the idea of business activity monitoring [9] (BAM) uses events to monitor business process performance. Analogously, technical implementations of business processes on top of SOAs (service compositions) are often monitored using CEP. To this end, many service composition engines can be configured to track their current state in event logs. For instance, the Apache ODE WS-BPEL engine triggers a rich model of execution events[1]. Similarly, service compositions implemented using Windows Workflow Foundation can use the .NET tracking service[2] to persist event logs. However, tracking system state via event logs in SOA is not confined to composition engines. For instance, The Vienna Runtime Environment for Service-Oriented Computing (VRESCo) [12] uses events to track not only service compositions, but all entities and interactions in a SOA (services, users, compositions, metadata and interactions).

In its most general form, an event log $\mathcal{E}$ consists of a sequence of n recorded events, i.e., $\mathcal{E} = \langle e_1, e_2, \ldots, e_n \rangle$. Each event $e_i \in \mathcal{E}$ usually contains at least an unique identifier, an event timestamp, the publisher of the event (e.g., the BPEL engine), the subject of the event (e.g., the composition instance that triggered the event), and the event type. Depending on the concrete event type, more detailed information is available. This type-specific information cannot be described generally, i.e., it is different from event type to event type as well as from system to system. In the following we describe the event types triggered by the VRESCo system as an example of the possibilities provided by event logs. The VRESCo event log will also constitute our running data set all along with this deliverable.

VRESCo is an experimental runtime environment developed at Vienna University of Technology. VRESCo is being developed under an open source license, and can be accessed via the project Web page[3]. The project aims at solving some of the research problems identified in [14], e.g., dynamic selection of services based on Quality-of-Service (QoS), dynamic rebinding and service composition, service metadata and event-based services computing.

[1] `http://ode.apache.org/ode-execution-events.html`
[2] `http://msdn.microsoft.com/en-us/library/ms735887(v=vs.85).aspx`
[3] `http://www.infosys.tuwien.ac.at/prototypes/VRESCo/`

In the following we focus on the latter aspect. The foundations of event-based service-oriented computing have been discussed in [2, 3]. In a nutshell, the goals of this earlier work were to track what is going on in a service-based application by constantly triggering events and using CEP to construct meaningful information from those events.

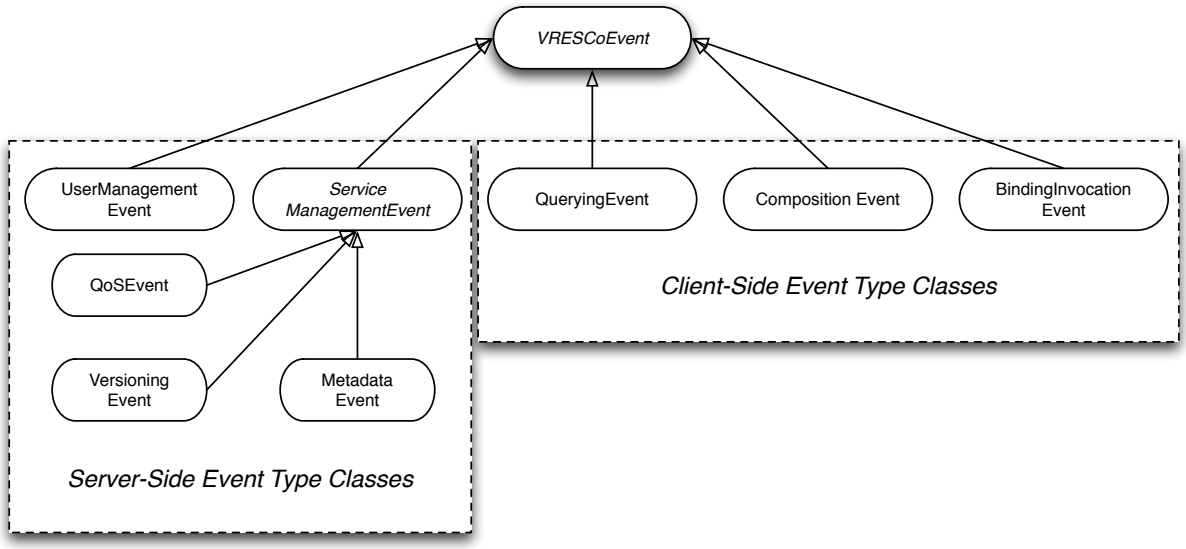

Figure 3.1: VRESCo Event Type Hierarchy

In a VRESCo system, events of various types are triggered. A simplified taxonomy of event type classes is depicted in Figure 3.1. As can be seen, events are triggered when services are queried, bound and invoked. Additionally, events indicate if the data or metadata about services changes (e.g., the QoS is changed, new operations are available). Each of the concrete event type classes (those with non-italic name) in turn contains a number of concrete event types that can be triggered. For full details on all events refer to [2].

Events in VRESCo can be triggered either on client- or server-side. While events concerning metadata are triggered by the VRESCo server, all querying and invocation events are triggered by clients and only processed throughout the VRESCo event engine. These client-triggered events are listed in more detail in Table 3.1. In the table, we provide the condition that triggers each event along with the event type and event type class of the event. All of these events provide the basic information discussed above (sequence number, timestamp, ...). In addition, events generally provide some type-specific additional information, which we also summarize in the table. For reasons of brevity, we have omitted composition events. Composition events in VRESCo are of comparable expressiveness as the events triggered by Apache ODE.

Event Type: *BindingInvocationEvent*		
Event Name	**Event Condition**	**Event Payload**
ServiceInvokedEvent	Specific service is invoked	Message sent to service
ServiceInvocationFailedEvent	Service invocation failed	Message sent to service, fault
ProxyRebindingEvent	Service proxy is (re-)bound to a specific service	Selected service
Event Type: *QueryingEvent*		
Event Name	**Event Condition**	**Event Payload**
RegistryQueriedEvent	Registry is queried	Query string
ServiceFoundEvent	Specific service is found by a query	Query string, query results
NoServiceFoundEvent	No services are found by a query	Query string

Table 3.1: Client-Triggered VRESCo Events.

The VRESCo event engine stores triggered events in an event log. Therein, events are serialized as XML and can be accessed and analyzed via a RESTful service interface. In Listing 3.1 we provide an example event serialized to XML. Evidently, the event reflects a service invocation with a very simple input message (<order>) as payload.

Listing 3.1: Serialized Invocation Event

```
<ServiceInvokedEvent xmlns="http://www.vitalab.tuwien.ac.at/vresco/usertypes">
  <Priority>0</Priority>
  <Publisher>guest</Publisher>
  <PublisherGroup>GuestGroup</PublisherGroup>
  <UserName>a007b09b-8c23-4fac-af30-0142a61f3795</UserName>
  <SeqNum>74006756-64f1-40cb-858e-565d4bc6a94c:24</SeqNum>
  <Timestamp>2010-11-09T09:58:49</Timestamp>
  <CurrentRevisionId>180</CurrentRevisionId>
  <CurrentRevisionWsdl>
    http://localhost:60000/AssemblyAtomicServices/IAssemblingPlanningService?wsdl
  </CurrentRevisionWsdl>
  <FeatureName>GetPartFeature</FeatureName>
  <InvocationInfo>
    <service_input>
      <order><part1>text</part1></order>
    </service_input>
  </InvocationInfo>
</ServiceInvokedEvent>
```

3.2 Web Search Engine Query Logs

Query logs keep track of information regarding interaction between users and the Web search engine. They record the queries issued to a search engine and also a lot of additional information such as the user submitting the query, the pages viewed and clicked in the result set, the ranking of each result, the exact time at which a particular action was done, etc. In general, a query log is comprised by a large number of records $\langle q_i, u_i, t_i, V_i, C_i \rangle$ where for each submitted query q_i, the following information is recorded: i) the anonymized identifier of the user u_i, ii) the timestamp t_i, iii) the set V_i of documents returned by the WSE, and iv) the set C_i of documents clicked by u_i.

From query log information it is possible to derive *Search Sessions*, sets of user actions recorded in a limited period of time. The concept can be further refined into: (i) *Physical Sessions*, (ii) *Logical Sessions*, and (iii) *Supersessions*.

Physical Sessions: a physical session is defined as the sequence of queries issued by the same user before a predefined period of inactivity. A typical timeout threshold used in web log analysis is $t_0 = 30$ minutes. [16, 21].

Logical Sessions: a logical session [5] or *chain* [18] is a topically coherent sequence of queries. A logical session is not strictly related to timeout constraints but collects all the queries that are motivated by the same information need (i.e., planning an holiday in a foreign country, gathering information about a car to buy and so on). A physical session can contain one or more logical session. Jones *et al.* [8] introduced the concepts of *mission* and *goal* to consider coherent information needs at different level of granularity, being a goal a sub-task of a mission (i.e., booking the flight is one of the goal in the more general mission of organizing an holiday).

Supersessions: we refer to the sequence of all queries of a user in the query log, ordered by timestamp, as a supersession. Thus, a supersession is a concatenation of sessions.

Sessions are, thus, sequences of queries submitted by the same user in the same period of time. This data can be used to devise typical query patterns, used to enable advanced query processing techniques. Click-through data (representing a sort of implicit relevance feedback information) is another piece of information that is generally mined by search engines. In particular, every single kind of user action (also, for instance, the action of not clicking on a query result) can be exploited to derive aggregate statistics which are very useful for the optimization of search engine effectiveness. How query logs interact with search engines has been studied in many papers. Good starting point references are [20, 4, 19].

Query Log Name	Public	Period	# Queries	# Sessions	# Users
Excite (1997)	Y	Sep 1997	1,025,908	211,063	∼410,360
Excite Small (1997)	Y	Sep 1997	51,473	–	∼18,113
Altavista	N	Aug 2, 1998 Sep 13, 1998	993,208,159	285,474,117	–
Excite (1999)	Y	Dec 1999	1,025,910	325,711	∼540,000
Excite (2001)	Y	May 2001	1,025,910	262,025	∼446,000
Altavista (public)	Y	Sep 2001	7,175,648	–	–
Tiscali	N	Apr 2002	3,278,211	–	–
TodoBR	Y	Jan 2003 Oct 2003	22,589,568	–	–
TodoCL	N	May 2003 Nov 2003	–	–	–
AOL (big)	N	Dec 26, 2003 Jan 01, 2004	∼100,000,000	–	∼50,000,000
Yahoo!	N	Nov 2005 Nov 2006	–	–	–
AOL (small)	Y	Mar 1, 2006 May 31, 2006	∼20,000,000	–	∼650,000
Microsoft RFP 2006	Y	Spring 2006 (one month)	∼15,000,000	–	–

Table 3.2: Features of the most important query logs that have been studied in the latest years. The dash sign (–) means that the feature in the relative column was non-disclosed.

An important key issue in query log mining is the pre-processing of logs in order to produce a good basis of data to be mined. An important step in usage analysis is thus the *data preparation*. This step includes: data cleaning, session identification, merging logs from several applications and removing requests for robots. This techniques aims to remove irrelevant items, so that the resulting associations and statistics reflects accurately the interactions of users with the search engine.

Very few query logs have been released to the public community in the last years due to their commercial importance and to privacy issues. Starting from 1997 the query logs that have been released to the public are: Excite (1997), AltaVista (1998–1999), AOL (2003–2004), AOL (2006), MSN (2006). Table 3.2 resumes the most important features of the query logs that have been examined in the latest years.

The most famous query log is undoubtedly AOL and it is also the data set we will refer to in the following of this document. The AOL data set contains about 20 million queries issued by about $650,000$ different users, submitted to the AOL search portal over a period of three months from 1st March, 2006 to 31st May, 2006. After the controversial discussion related to users' privacy issues followed to its initial public delivery, AOL has withdrawn the query log from their servers and is not offering it for download anymore.

Figure 3.2 shows a fragment of the AOL query log. Each row of this query log consist of records collecting five fields: i) the ID referring to the user issuing the query, ii) the issued query, iii) the time the

```
507  kbb.com            2006-03-01 16:45:19  1   http://www.kbb.com
507  kbb.com            2006-03-01 16:55:46  1   http://www.kbb.com
507  autotrader         2006-03-02 14:48:05
507  ebay               2006-03-05 10:50:35
507  ebay               2006-03-05 10:50:52
507  ebay               2006-03-05 10:51:24
507  ebay               2006-03-05 10:52:04
507  ebay               2006-03-05 10:52:36  69  http://antiques.ebay.com
507  ebay               2006-03-05 10:58:00
507  ebay               2006-03-05 10:58:21
507  ebay electronics   2006-03-05 10:59:26  5   http://www.internetretailer.com
507  ebay electronics   2006-03-05 11:00:21  20  http://www.amazon.com
507  ebay electronics   2006-03-05 11:00:21  22  http://gizmodo.com
507  ebay electronics   2006-03-05 11:00:21  22  http://gizmodo.com
507  ebay electronics   2006-03-05 11:18:56
507  ebay electronics   2006-03-05 11:20:59
507  ebay electronics   2006-03-05 11:21:53  66  http://portals.ebay.com
507  ebay electronics   2006-03-05 11:25:35
```

Figure 3.2: An example of the AOL query log [15].

query was issued to the search engine, iii) the position of the clicked result in the results page, and iv) the host of the clicked document.

Topics covered by queries contained in search engines are the most disparate. Figure 3.3 highlights the 250 most frequent terms in the AOL query log by means of a tag cloud. The dimension of each term in the tag cloud is directly related to its frequency in the log. The larger the term in the tag cloud, the more frequent it is in the log. As an example,"google" and "free" are the two most frequent terms in the log. Other very frequent words are: "yahoo", "new", "county", "pictures", "http".

Figure 3.3: A tag cloud of the 250 most frequent words in the AOL query log [15]. Picture has been generated using wordle.net. From [20].

Furthemore, the queries submitted by users to search engines and thus contained in query logs are sometimes incredible. Just to give an idea, from the AOL query log user #2386968 submitted the query "why is my husband so talkative with my female friends". Another funny example is the query submitted by the user #427326 looking for "where is my computer?".

Chapter 4

Different Types of Knowledge Extracted and Possible Applications

The knowledge extracted from service logs can be fruitfully applied for different purposes within SOAs or Web domains. While the use of this type of knowledge within Web domains is useful for enhancing both the *effectiveness* (i.e., quality of results returned to the users) and the *efficiency* (i.e., response time) of the overall search infrastructure, SOAs could exploit this knowledge during different steps of the life-cycle of their compliant applications. In particular, at design-time it is possible to provide service designers with tools aiming at simplifying/suggesting possible (and actually used) interactions between groups of services. These tools could thus simplify the whole design phase, by giving service designers hints on possible associations between services, or possible workflows of activities (tasks) carried on by groups of services. Furthermore, at run-time phase it is also possible to study adaptation strategies that exploit this historical knowledge. In this deliverable we present five applications of different types of knowledge derived from logs coming from SOA or Web domains.

The first contribution (see Appendix A) refers to the use of knowledge extracted from service logs in order to perform a semantic resource allocation within Cloud Computing environments [6].
In this work authors introduce a generic framework for prediction and adaptation, and describe its application in a concrete scenario (Cloud resource scheduling). The basic requirements for such an environment are:

- to provide generic capability of collecting log data about internal events;
- to unify the collected data so that global coherences can be revealed;
- to provide customizable methods for getting predictions based on the collected data;
- to feed back predictions into the realization of adaptation mechanisms.

This framework combines the prediction techniques with the semantic technologies which introduces semantic knowledge to the data evaluated by predictors. Furthermore, it also exploits a multi-agent system to introduce proactiveness and distributed problem solving for increasing the scalability, adaptability and self management of the system. The predictions extracted from the semantic data are taken into account by a group of agents for allocating the different costumer jobs in the most reliable resources for each case.
Authors emphasize on the importance of using historical data by service and cloud providers. They present a generic approach and a re-usable solution for the collection and exploitation of historical log data produced by services and demonstrate its usability and usefulness in a concrete resource scheduling scenario. The heterogeneity of log data arriving from various resources calls for a semantic data representation, which can facilitate the unification of these data and a query mechanism supported by

inferencing. The paper belongs to work package JRA-2.3 as the presented solution is mostly effective on the infrastructure layer, however the methodology and the semantic approach addresses work package JRA-1.3 as well. Regarding WP-JRA-2.3 research challenges the paper is related to "Supporting adaptation of service-based applications" and "Runtime SLA Violation Prevention". The presented solution provides a generic technique to predict the quality attributes of services and resources, which serves as an enabler to various self-adaptation techniques. The paper is more focused on the resource scheduling task, and provides a semantic model for the representation of QoS measurements and requirements used in the scheduling process.
As the decision making is distributed in the suggested generic framework, there is a possibility to extend standard decision making with additional behaviors representing strategies and policies on different levels; on the level of resource providers and on the level of service providers.

The second contribution (App. B) consists of an application of process mining techniques on a real-world service log coming from the VRESCo runtime environment [13].
The vast majority of nowadays software-based systems, ranging from the simplest, i.e., small-scale, to the most complex, i.e., large-scale, record massive amounts of data in the form of *logs*. Such logs could either refer to the functioning of the system as well as keep trace of any possible software or human interaction with the system itself. For this reason, logs represent a valuable source of hidden knowledge that can be exploited in order to enhance the overall performances of any software-based system.
Well-known examples of systems that have started trying to improve their performances by analyzing event logs are surely Web Search Engines (SEs). Roughly, SEs are increasingly exploiting past user behaviors recorded in *query logs* in order to better understand people search intents, thus, for providing users with better search experiences. Indeed, by accurately recognizing and predicting actual user information needs, SEs are now able to offer more sophisticated functionalities (e.g., *query suggestion*) as well as better relevant result sets in response to a specific query (e.g., *query diversification*).
Moreover, there are plenty of modern enterprise software systems that need to operate in highly dynamic and distributed environments in a standardized way. Such systems implement their business logic according to the *Service-oriented Architecture* (SOA) principles, thus, assembling their business processes as the composition and orchestration of autonomous, protocol-independent, and distributed logic units, i.e., *software services*.
Service-based systems and applications (SBAs) require proper *run-time* environments where their composing services can be searched, bound, invoked, monitored and managed. Therefore, SBA's run-time support might keep track of what is going on during the whole application lifecycle by roughly recording all such events to log files, i.e., *service event logs*.
Analysis of such service event logs could reveal interesting *patterns*, which in turn might be exploited for improving the overall performances of SOA's run-time frameworks as well as supporting SBA designers during the whole application lifecycle.
The main contribution of this work concerns the application of data mining techniques to a real-life service event log collected by the VRESCo SOA run-time framework. Our aim is to analyze the historical events stored on VRESCo in order to discover software services that are frequently invoked and composed together, i.e., *process mining*.
Although traditional process mining refers to a set of techniques and methodologies whose aim is to distill a structured process description from a set of actual traces of executions recorded in event logs, here we treat it as an instance of the *sequential pattern mining problem*.
Finally, authors apply two sequential pattern mining algorithms to a real event log provided by the Vienna Runtime Environment for Service-oriented Computing, i.e., VRESCo. The obtained results show that they are able to find services that are frequently invoked together within the same sequence. Such knowledge could be useful at *design-time*, when service-based application developers could be provided with *service recommendation tools* that are able to predict and thus to suggest next services that should be included in the current service composition.

The third work (App. C) addresses the problem of interface-based test coverage for Web services [7].

In recent years, the Service-Oriented Architecture (SOA) has become a widely adopted paradigm to create loosely coupled distributed systems, and Web services are the most commonly used technology to build SOA. One of the defining characteristics of SOA is that the API (Application Programming Interface) of the available services is published to service consumers using standard, machine-readable description languages. In the case of Web services, this is achieved using the Web Services Description Language (WSDL), paired with XML Schema Definitions (XSD) of the service operations' input and output messages.
Now that the WS-* stack builds a solid technological foundation, traditional software engineering disciplines are being applied to Web services. Among these disciplines is software testing in terms of Verification and Validation (V & V). An important field in software testing is concerned with test coverage, i.e., the extent to which an implemented system has been tested or used. Classical code-based coverage metrics such as function, statement or branch coverage require access to the source code, which is not always possible for Web services. Therefore, testing methods for Web services can usually only rely on the service's API definition, and hence focus on black-box testing of single services or testing the composition of services.
API coverage is commonly expressed as the ratio of previously performed, distinct invocations to the number of (theoretically) possible invocations as defined by the API. In the simplest case, single parameter values are varied and tested, e.g., with extreme values, to verify functionality and detect failures. Since an isolated analysis of parameters is often not sufficient, input combinations need also be considered. Achieving an API coverage of (near) 100% in this case is generally subject to the problem of combinatorial explosion, because all input parameter combinations need to be considered. Hence, it is desirable to restrict the value domain of each parameter to its smallest possible size, in order to minimize the total number of possible combinations. The authors of this work see two possibilities to achieve that. Firstly, Web service developers need to provide a more precise specification of the valid operation parameters, e.g., in terms of string patterns or allowed numeric ranges. Secondly, service testers may analyze these parameters to identify similarities or combinations that seem less important to be tested. XSD already provides a solution for the first point in the form of facets, but expressing such schema-based restrictions is surprisingly hard to achieve in major Web service frameworks such as JAX-WS (Java API for XML Web Services) or Microsoft's .NET platform. Essentially, developers cannot rely on the automated XSD generation, but have to manually define the XML Schema that uses facet restrictions. Concerning the second point, there is still an evident lack for API coverage frameworks with customizable coverage metrics that can be easily plugged into (existing) service-based systems.
In a previous work, authors developed coverage metrics for data-centric dynamic service compositions. Under the same umbrella project named TeCoS (Test Coverage for Service-based systems) they now investigate coverage of service APIs. In this paper, authors apply software testing concepts to Web services and present a solution for measuring API coverage based on historical invocations. Overall, their contribution is threefold: 1) they define API coverage metrics and their instantiation for Web services, 2) they suggest the implementation of XSD facets in the JAX-WS framework, and 3) they present and evaluate our prototype in an experimental evaluation.
The proposed approach is based on domain partitioning, a technique used to narrow down the domain space of invocation parameters, and hence to reduce the number of possible invocations to obtain more meaningful coverage metrics. The prototype implementation of TeCoS intercepts Web service invocation messages and converts the XML tree representation to a relational schema for storage in a relational database management system (DBMS). The domain partitions are defined by the user in a tailor-made expression language. To compute a certain coverage metric of some Web service operation o, the user defines which partitions should be applied to which part of the elements in the schema of the input message of o. The actual computation of the coverage value is performed on the underlying DBMS. In an ongoing work, authors plan also to extend the scope of API coverage to invocation sequences and semantic input description. Moreover, they will further enhance TeCoS with support for distributed storage and coverage computation.

The fourth contribution (App. D) aims at devising effective techniques for identifying task-based sessions, i.e. sets of possibly non contiguous queries issued by the user of a Web Search Engine for carrying out a given task [10].
The *World Wide Web* (Web) was born as a *platform* to connect academic and research people, which exploit the Internet as the communication medium infrastructure. Rapidly, an increasing number of users, which were not directly involved in academia or research activities, have started to have access to the Web as well.
Nevertheless, in the first Web era there was still a clear separation of roles between *few* content providers, i.e., typically skilled workers and professionals, and *many* content consumers, i.e., common end users.
During the last years, a new trend has gained momentum: new applications that allow easy authoring and content creation have lead to an increased *democratization* and *collaborative involvement* in the Web. Somehow, this process caused the end of the first Web era, by bringing down the wall between content providers and consumers, which now can play both roles interchangeably from time to time. Therefore, information made available on the Web have started raising at a tremendous speed rate, reaching nowadays a huge and still growing number of contents, which spread over several media types (e.g., text, images, audio/video, etc.).
This great repository of data makes the Web the place of choice where people look at whenever they come up with *any* sort of information need. Indeed, there is a common belief that the Web is increasingly used not only for consulting documents but also for trying to simplify the accomplishment of various everyday activities, i.e., *tasks*.
Moreover, most of the interactions between users and the Web are often mediated by *Web search engines*, which are amongst the most important and used Web-based tools. This trend is confirmed by a rising "addiction to Web search": no matter what an information need is, user will ask it to a Web search engine that will hopefully give her the answer she expects.
Although the huge number of features which now the most popular Web search engines come with, in essence they still belongs to the category of Web documents retrieval tools. The results they provide in response to a user query are given according to the traditional "*ten blue links*" paradigm, i.e., links to Web pages that are considered *relevant* to the given user query. If results are not satisfactory, decision taken by looking at some of them, users may decide to "re-phrase" the query to try to refine the retrieved results. However, when the need behind a certain query is a task to be accomplished, this "*query-look-refine*" paradigm could not be effective. In other words, for certain (quite popular) tasks, Web search engines as we know can be considered obsolete tools.
Therefore, authors believe next-generation Web search engines should turn from mere Web documents retrieval tools to multifaceted systems, which fully support users while they are interacting with the Web. Of course, this opens up novel and exciting research challenges, in particular the ability to recognize implicit user tasks from the issued queries. Thus, authors focus on identifying *task-oriented sessions* from past issued queries, i.e., sets of possibly non-contiguous queries phrased by users for carrying out various tasks.
First, they built, by means of a manual labeling process, a *ground-truth* where the queries of a real query log have been grouped in tasks. The analysis conducted by authors on this ground-truth shows that users tend to perform more than one task at the same time, since about 75% of the submitted queries involve a multi-tasking activity.
Moreover, authors define the *Task-oriented Session Discovery Problem* (TSDP) as the problem of best approximating the above ground-truth. The TSDP deals with two aspects: *(i)* a robust measure of the *task relatedness* between any two queries, i.e., *task-based query similarity*, and *(ii)* an effective method for actually discovering task-oriented sessions by using the above measure of task relatedness. Concerning *(i)*, we propose and compare both *unsupervised* and *supervised* approaches for devising several task-based query similarity functions.
These functions also exploit the collaborative knowledge collected by *Wiktionary* and *Wikipedia* for detecting query pairs that are not similar from a lexical content point of view, but actually *semantically*

related. Therefore, authors tackle *(ii)* by introducing a set of *query clustering* methods that exploit the above similarity functions for detecting user tasks.
All the proposed solutions have been evaluated on the ground-truth, and two of them have been shown to perform better than state-of-the-art approaches.

The fifth contribution (App. E) outlines the requirements for a reliable management in crowd-computing environments.
Open Web-based and social platforms dramatically influence models of work. Today, there is an increasing interest in outsourcing tasks to crowdsourcing environments that guarantee professional processing [17].
Crowdsourcing follows the "open world" assumption allowing humans to provide their capabilities through the platform by registering themselves as members. While conventional enterprise systems rely on well-defined in-house knowledge of their processes and services, i.e., on established policies, crowdsourcing has a more loosely-coupled, dynamic, and flexible structure and depends especially on the preferences and behavior of the individual crowd members. Hence, it is necessary to monitor, log, and extract knowledge on the behavior of the crowd members to derive an adaptive, optimized usage.
Crowd behavior can lead to an incomplete and unsatisfactory task state at deadline. As a consequence, meeting promised service contracts is challenging and demands for sophisticated management techniques of a crowd platform.
The challenge is to gain the crowd customer's confidence by organizing the crowd's mixture of capabilities and structure to become reliable. The prerequisite is a monitoring infrastructure that updates a crowd-based resource model. Authors focus on an agreement model derived from the collected knowledge and combined with an adaptation approach for reliable service usage and task execution.
This model integrates crowdsourcing with the well-known concept of SLAs. For SLAs, independent from the agreed parameters of quality, the two common categories are hard- and soft-constraints. In this work, for crowdsourcing authors will distinguish between criteria that must be met, e.g., expertise area of crowd members and their principal participation interest and soft constraints that are used for ranking potential crowd members, including their capacity, reputation, and costs. These categories combined with on-line feedback data from behavior monitoring will allow to adapt and optimize assignments according to the agreements and the current crowd status. Eventually, authors show an integration of observed crowdsourcing in a SOA environment and the related enforcement of agreements by the WSLA standard.

Chapter 5

Conclusions

In this deliverable we presented five different contributions regarding the exploitation of usage data coming from Service Oriented Architectures or real world Web search engines. Such knowledge is mostly derived by using data mining techniques. Five possible applications of this derived knowledge is presented here.

First, we introduced a resource allocation mechanism that uses semantic annotated historical data to enable adaptation mechanisms of the run-time environment. The contribution shows the importance of using historical data for service and cloud providers. Furthermore, the approach demonstrates the coupling of semantic data processing with data mining as a promising novel combination.

Secondly, two well-known data mining techniques have been applied to real event logs coming from a event-based service-oriented architecture. The contribution focuses on process mining as a specific instance of the more general sequential pattern mining problem. The aim of this contribution is to detect frequent sequential patterns that might be present in actual traces of service executions recorded in event logs. To this end, two sequential pattern mining algorithms have been applied to a real event log provided by the Vienna Runtime Environment for Service-oriented Computing, (i.e., VRESCo). The results obtained show that it is possible to find services that are frequently invoked together within the same sequence. Such knowledge could be useful at design-time, when service-based application developers could be provided with service recommendation tools that are able to predict and thus to suggest next services that should be included in the current service composition.

In the third contribution we presented an efficient, novel solution to measure API coverage of service-based systems implemented with Web services. User-specified domain partitioning allows for the definition of customizable and reusable API coverage metrics. The proposed end-to-end framework TeCoS can be easily plugged into existing service execution engines to log service invocations, calculate coverage data and render the results in a Web UI. This user-friendly extension remarkably reduces the required development effort and is a step towards meaningful coverage data and schema-based validation of invocation parameters. The performance and scalability of the proposed approach are successfully evaluated within the experimentation.

The fourth contribution discussed a technique for splitting into meaningful user sessions a very large, long-term log of queries submitted to a Web Search Engine (WSE). The contribution formally introduced the Task-based Session Discovery Problem as the problem of extracting from a stream of users queries several subsequences of queries which are all related to the same search goal, i.e., a Web-mediated task. The contribution also proposed a clustering-based solution, leveraging distance measures based on query content and semantics, while query timestamps were used for a first pre-processing breaking phase. In particular, the proposed technique exploited both Wikipedia and Wiktionary to infer the semantics of a query. The novel graph-based heuristic, namely QC-htc, which is a simplification of the weighted connected components QC-wcc, significantly outperforms other heuristics in terms of F-measure, Rand and Jaccard index.

In the last contribution a framework for successfully managing task assignment in a crowd-computing

environment is presented. The framework solves the problem of managing the task assignment with an adaptive and multi-objective task scheduling. The environment is based on a large-scale SOA infrastructure. The extension to one of the existing standards for agreements (WSLA and WS-Agreement) which includes assignment identifying information and relation to different objectives, fits the requirements of our crowd-computing scenario. When deploying the assignment as independent and dependent tasks to capable members, these objectives can than be used as soft- or hard-constraints for a weighted scheduling strategy. The results highlight the advantages of an objective-aware metric ordered strategy in contrast to plain random scheduling while task loads remain in between the boundaries. Nevertheless, the results show that the effort for ordering the assignment lists induces a higher effort in scheduling.

Bibliography

[1] Integration framework baseline, s-cube deliverable cd-ia-3.1.1, 2009.

[2] Anton Michlmayr and Florian Rosenberg and Philipp Leitner and Schahram Dustdar. Advanced Event Processing and Notifications in Service Runtime Environments. In *Proceedings of the 2nd International Conference on Distributed Event-Based Systems (DEBS'08)*, 2008.

[3] Anton Michlmayr and Philipp Leitner and Florian Rosenberg and Schahram Dustdar. Publish/-Subscribe in the VRESCo SOA Runtime. In *Proceedings of the 2nd International Conference on Distributed Event-Based Systems (DEBS'08)*, 2008. invited demo paper.

[4] R. Baeza-Yates. Applications of web query mining. *Advances in Information Retrieval*, pages 7–22, 2005.

[5] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proc. KDD'07*. ACM, 2007.

[6] J. Ejarque, A. Micsik, R. Sirvent, P. Pallinger, L. Kovacs, and R. M. Badia. Semantic resource allocation with historical data based predictions. In *The First International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2010*, 2010.

[7] W. Hummer, O. Raz, and S. Dustdar. Towards efficient measuring of web services api coverage. In *3rd International Workshop on Principles of Engineering Service-Oriented Systems (PESOS), co-located with ICSE 2011*, 2011.

[8] 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.

[9] H. Kochar. Business activity monitoring and business intelligence, 2005.

[10] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 277–286, New York City, NY, USA, 2011. ACM Press.

[11] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[12] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. *IEEE Transactions on Services Computing*, 3:193–205, July 2010.

[13] F. M. Nardini, G. Tolomei, P. Leitner, F. Silvestri, and S. Dustdar. Mining lifecycle event logs for enhancing service-based applications. Technical Report N. /cnr.isti/2010-TR-017, CNR ISTI Pisa Italy, 2011.

[14] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer*, 40(11):38–45, November 2007.

[15] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, InfoScale '06, New York, NY, USA, 2006. ACM.

[16] B. Piwowarski and H. Zaragoza. Predictive user click models based on click-through history. In *Proc. CIKM'07*. ACM, 2007.

[17] H. Psaier, F. Skopik, D. Schall, and S. Dustdar. Resource and agreement management in dynamic crowdcomputing environments. In *2011 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2011)*. IEEE Computer Society, 2011.

[18] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proc. KDD'05*. ACM Press, 2005.

[19] A. Scime. *Web Mining: Applications and Techniques*. IGI Publishing Hershey, PA, USA, 2004.

[20] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, 1(1-2):1–174, 2010.

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