FP7-ICT-2009-4-248061

EnPROVE

Energy consumption prediction with building usage measurements for software-based decision support

Instrument:	Small or medium-scale focused research project
Thematic Priority:	Theme 3 – Information and Communication Technologies

D6.1 BAU-WSN AND EPDSS INTEGRATION FRAMEWORK

Due date of deliverable:	31.01.2012
Actual submission date:	31.01.2012

Start date of project: 01.02.2010 Duration: 36 months

Organisation name of lead contractor for this deliverable:	GEM
Dissemination level:	PU
Revision	1

Executive Summary

This document presents deliverable *D6.1 BAU-WSN and EPDSS Integration Framework*. This deliverable describes the integration and testing efforts related to the functional packages developed in *WP5 Implementation* and their access through the BAU-WSN and EPDSS systems.

The deliverable also describes work done as part of *Task 6.3 EnPROVE platform integration and testing*.

The first section, Introduction, presents a general overview over the module components, the integration goals and concepts of the framework, namely the communication, GUI and dashboard components, the goals of streamlining the GUI and interfaces and a general runtime independence of the two systems.

Section 2, Integration, describes the integration results, first from a user and then from a technical perspective, to show how the goals and concepts are implemented in the framework. The user perspective shows a system that presents itself a specialized GUI for all EPDSS and on site building assessment tasks and another one for the main auditing tasks. The technical communication between the two systems is implemented via web service providers on the EPDSS side (communication service) and on the BAU-WSN side (auditing results service) that each are accessed by respective clients in the other system.

Section 3, Testing, describes how testing is facilitated on the various integration levels. While unit testing already provides the methodology for initial development and regression testing and artificial testbed data is provided to take the place of unavailable real data, the needed scope of software support for validation and benchmark testing is not yet profoundly defined and will be developed in the later phases of integration.

Section 4, Deviations from previous design and deliverables, describes the deviations in the implementation from previous designs and deliverables, which is mainly the integration of the graphical GUI component into the EPDSS GUI module instead of into the auditing dashboard GUI.

Abbreviations

BAU	Building Usage and Performance Auditing
D	Deliverable
e.g.	*exempli gratia* = for example
EC	European Commission
EPDSS	EnPROVE prediction and decision support engine
etc.	*et cetera*
GUI	Graphical user interface
i.e.	*id est* = that is to say
ICT	Information and Communications Technologies
EnPROVE	Energy consumption prediction with building usage measurements for software-based decision support
M	Month
MS	Milestone
RTD	Research and Technological Development
S&T	Scientific & Technological
STREP	Small or medium-scale focused research project
w.r.t.	With respect to
WP	Workpackage
WSN	Wireless Sensor Network

Contents

1 Introduction

1.1 EnPROVE platform components

The EnPROVE platform consists of two major systems that need to communicate with each other:

- The Building Performance and Usage Auditing (BAU) includes a wireless sensor network (WSN) deployed in the building to be renovated, connected to local gateways that transmit data to the remote building performance and usage server, which processes this
- The Energy Prediction and Decision Support System (EPDSS) responsible for interacting with the technical consultant to extrapolate the data collected from the building and predict energy consumption for several possible technical solutions, and enable the investor in selecting the best renovation scenario considering tangible (e.g. return on investment) and intangible (e.g. comfort level) criteria.

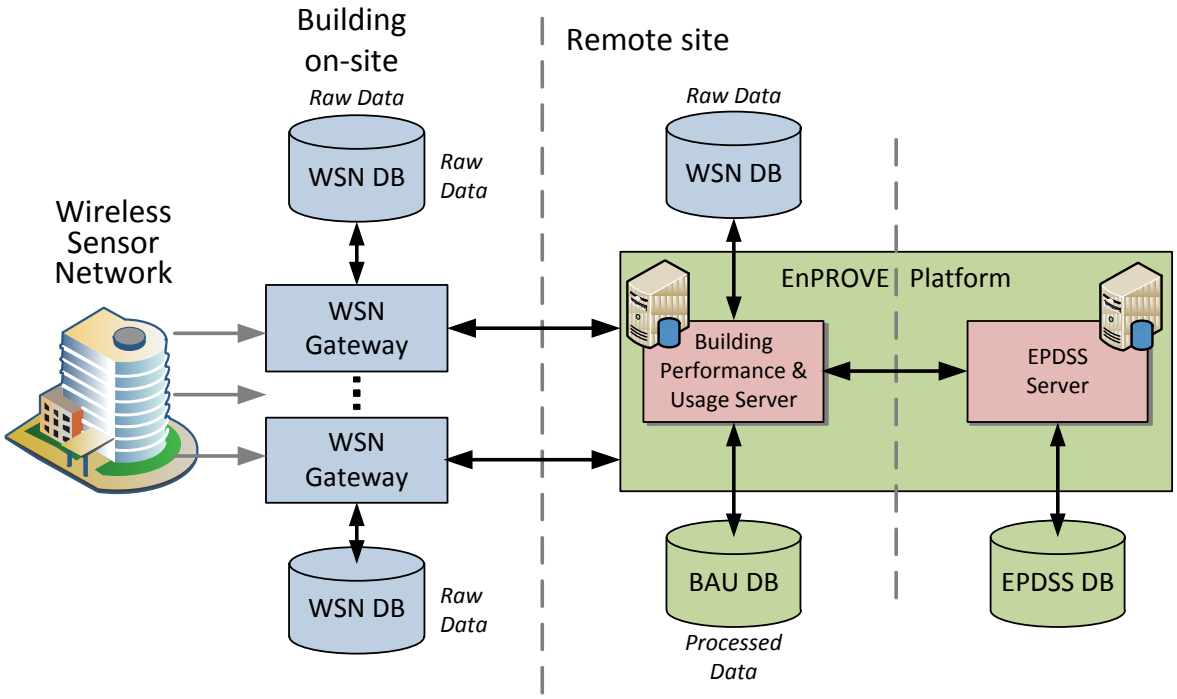

Figure 1: EnPROVE platform

Figure 1 shows the EnPROVE platform with the EPDSS on the right side and the BAU-WSN on the middle section and left side.

1.2 EnPROVE assessment projects and actors

When the term assessment project is used it refers to the EnPROVE representation of a building renovation project and the EnPROVE processes involved.

An assessment project is operated by several actors that access the system for different purposes during different phases of its execution. There are two key actors involved in the communication between the EPDSS and BAU-WSN. The technical consultant can be viewed as the assessment

project manager and the auditing contractor is in charge of the work related to the sensory building auditing on site.

The actors are involved in the execution of the following successive project steps:

1. Create an assessment project and define generic constraints or legal requirements, as well as energy/financial targets (investor and technical consultant).
2. Define or import a building model (technical consultant with help of external consultants).
3. Structure the model by partitioning the building into a functional zone hierarchy and then specify the zones current installed equipment which comprises the buildings baseline scenario (technical consultant with help of the auditing contractor).
4. Request the auditing results of a given period (technical consultant).
5. Install the auditing equipment, audit the building and return the results (auditing consultant).
6. Extrapolate building usage data for the whole year, based on auditing results (technical consultant).
7. Predict and calibrate the current building consumption and create of a set of renovation scenarios (technical consultant).
8. Evaluate scenarios and select one for implementation (technical consultant and investor).

1.3 Integration Goals

The integration and testing framework establishes a software communication system, that enables the EnPROVE users to retrieve and supply information needed in each step of the processes that are executed on the EPDSS and on the auditing side of the system.

The EPDSS and the BAU-WSN systems should work relatively independent from each other, can be located on separated hardware environments and should not need to communicate constantly or even regularly during a assessment project.

Within the EPDSS system a common data model, data access and graphical user interface concept form the core of the systems integration, that guarantees a smooth internal information flow and helps preventing general design or conception errors during the implementation and integration of the contained functional modules.

The auditing dashboard and unified results database take the same role on the BAU-WSN side of the EnPROVE platform.

The integration framework for the integration of the EPDSS and the BAU-WSN can therefore concentrate on the data exchange and mapping between two monolithic models as opposed to considering multiple loosely coordinated models on each side.

The EPDSS system uses Web-Services technology to facilitate communication between the EPDSS Server and the client modules (graphical user interface, external systems).To align this methodology with the communication between the BAU-WSN and the EPDSS system is therefore obvious.

Regarding the users perspective, integration should be as invisible as possible. Since multiple users can be working parallel on the system, an asynchronous communication calls for a separated request/answer approach that has already been sketched in deliverable *D3.2 BAU-WSN and EPDSS* Integration Framework.

2 Integration

2.1 User Perspective

The communication between the technical consultant and the auditing contractor has to be initiated by phone or preferably an email by sending some initial connection information from the technical consultant to the auditing contractor, that allows the BAU-WSN software to establish a link to the EPDSS system. All further software communication is then automatically controlled between both systems.

After an EnPROVE assessment project has been created and a graphical building model has been imported from an external (CAD-) system the technical consultant needs to prepare the building model for auditing by specifying hierarchical auditing zones (e.g. grouping spatial partitions and spaces into zones).

This 'zoning' is the elementary preparation task in order to facilitate the later deployment and mapping of sensors and sensor readings to a common building model abstraction that is interpreted by the BAU-WSN and EPDSS functional modules alike.

The initial version of this zone model helps specifying and documenting the auditing scope regarding negotiations between EPDSS technical consultant and auditing contractor.

The final zone model which also leads to the baseline Scenario will be generally established as a collaborate task by the technical consultant and the auditing contractor and has to be fixed before the auditing data collection phase starts.

To facilitate the collaboration or initial reviewing, the auditing contractor uses the EPDSS graphical interface to access the zone defining and reviewing functionality. The part of EPDSS GUI needed will be accessed through the auditing dashboard GUI and appear in a separate window, that is self-contained and not interacting with the dashboard GUI.

After the zone model has been fixed, and any manual building model changes have been realized the main auditing phase can start. The EPDSS defined zones have now to be transported from the EPDSS to the BAU-WSN software. To facilitate this, the technical consultant requests via GUI auditing to commence and the auditing software will extract the required information from the EPDSS system.

Now the auditing contractor will start the main auditing phase by installing the auditing hardware, initializing the auditing software system and starting the measuring phase.

After all auditing data has been collected and the auditing results database is built, the auditing contractor will release this information, using the GUI, to the EPDSS system, which will signal the technical consultant, that the auditing results are available for processing.

The EPDSS system will from then on automatically access the auditing data from the BAU-WSN system when needed without any explicit user action required. This also concludes the interaction between the EPDSS and BAU-WSN systems until another auditing is required.

2.2 Technical Perspective

The communication between the EPDSS and the BAU-WSN system is established via Web-

Services.

The EPDSS system hosts the center of EnPROVE communication and provides a communication service also to the BAU-WSN system. Via an initial semi-manual connection procedure, the technical consultant and auditing contractor will make the web-address of the EPDSS server known to the BAU-WSN client software, which will then connect itself to the server and listen to requests sent by it. The request will include the information needed to connect to the appropriate Web-Service and access the provided interfaces.

On the BAU-WSN system also resides a Web-Service which is responsible to supply pre-processed auditing results data to an EPDDS client. The BAU-WSN server web-address will be supplied by the BAU-WSN client as an answer to an EPDSS server request for auditing.

While the EPDSS server Web-Services will have to be available constantly while an assessment project is carried out, the BAU-WSN client to the EPDSS and the auditing results server need only be online on demand. For the client this will be during the auditing preparation phase until the main auditing phase commences and after auditing is complete until the end of the assessment project. The BAU-WSN server needs to be online from the signalling of auditing completion until the EPDSS system signals the BAU-WSN client that the service is not required anymore.

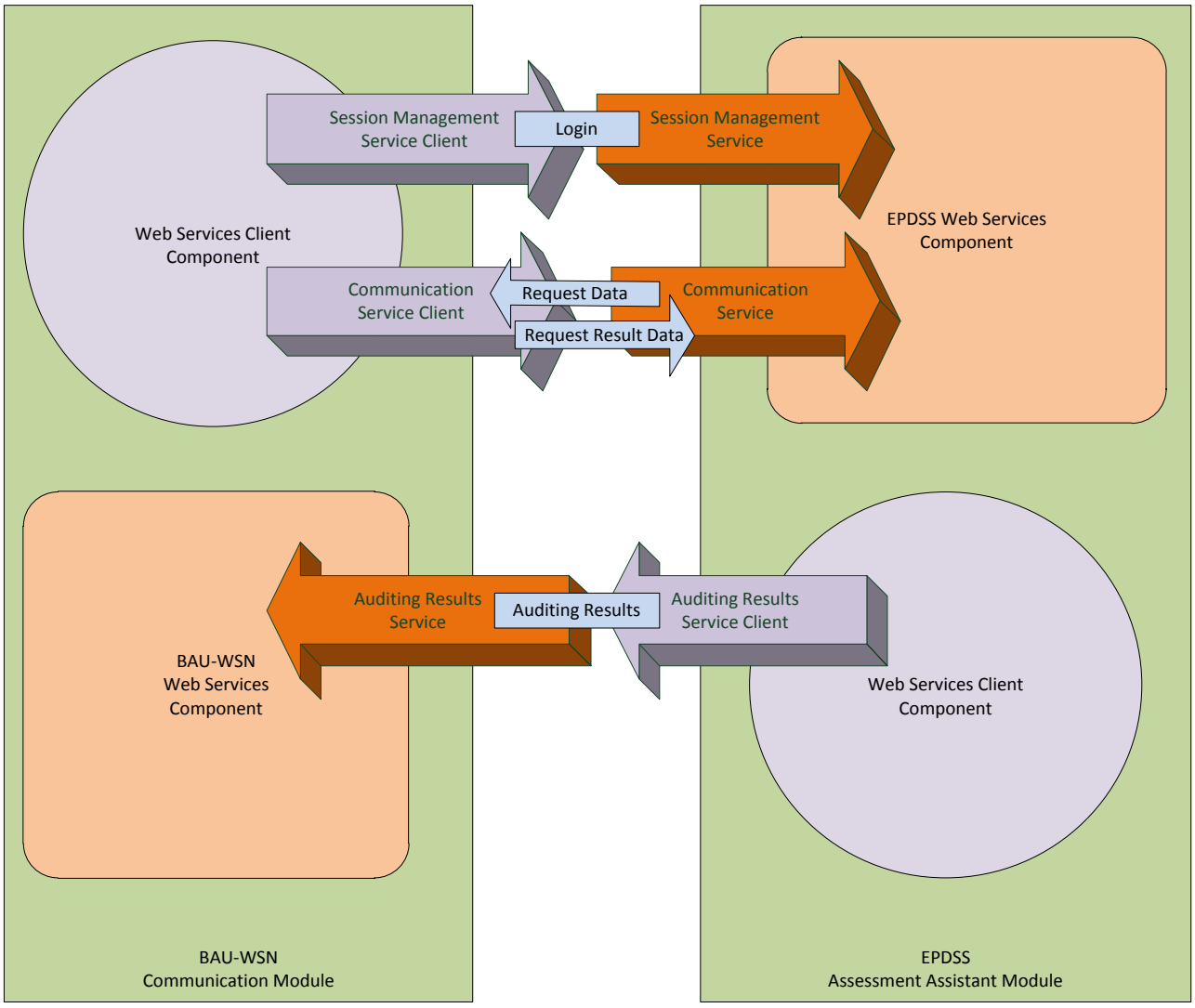

Figure 2: Web Services Framework

Figure 2 shows the web services (orange) and clients (purple) and the data transported (light blue).

All interactions are initiated by client calls. In order for the BAU-WSN client to timely receive requests from the EPDSS, the client has to poll the service regularly for requests.

2.2.1 Employed Tools and Technology

All EnPROVE web services communication is implemented with Apache Tomcat Servers using an Axis2 supported Java mapping with standard ADB data-binding to generate the WSDL defined interface. The service clients are Axis2 generated Java stubs with the exception of the .NET plugin clients used to plug into the AutoCAD and Revit platforms.

The service interface functions and the complex data types that are transported over the interfaces are all designed to be interchangeable between most Java and .NET environments. Their Java definition is using only simple types for attributes (Boolean, Long, Double, String), simple arrays (no collection classes) for interfaces and nested complex types of the same restrictions.

2.2.2 The EPDSS communication service

The communication service interfaces used for communication control between the EPDSS and the BAU-WSN systems are represented by two main interfaces and two main data types described in the following.

2.2.2.1 The interface for the session management

Method Summary	
boolean	**closeSession**(java.lang.String sessionKey) close the web services session identified by the sessionKey
java.lang.String	**createSession**(java.lang.String username, java.lang.String password, java.lang.String clientId) create a web services session for the given web services client and user and return a web service session key for session identification
boolean	**sessionValid**(java.lang.String sessionKey) check if the session, identified by the session key is valid

The session management interface is the most basic EPDSS web service, consumed by all EPDSS web clients.

After initial communication has been established by the user, the service client software logs into the EPDSS system via the `createSession` call. The returned session key identifies the user and the client module to all other EPDSS web service calls. The client communication and systems involved are therefore always identified with an actor.

2.2.2.2 The interface for the EPDSS communication web service

Method Summary	
EPDSSCommunicationStatus	**acceptOrDeclineRequest**(java.lang.String sessionKey, long requestId, boolean accepted, java.lang.String reason) mark a request as accepted (handled) by the calling client/user
boolean	**checkForRequests**(java.lang.String sessionKey) ask the service, if a request is waiting for the calling client/user (identified by the sessionKey)
EPDSSRequestData[]	**getRequests**(java.lang.String sessionKey) retrieve all unhandled requests for the calling client/user (identified by the sessionKey)
EPDSSCommunicationStatus	**sendRequestResults**(java.lang.String sessionKey, EPDSSRequestResultData data) send the results of a request to the EPDSS

The logged in client regularly polls the EPDSS system for requests, such as a request for auditing, with the `checkForRequests` call. If yet unhandled requests are available they can be retrieved by a call to `getRequests`.

All requests accepted by the client will be acknowledged by a positive call to `acceptOrDeclineRequest`. All requests that cannot be handled by the client should be declined with the same call. Acknowledged requests are immediately removed from the request queue.

Finally if the request has been accepted and successfully fulfilled, the results are returned by a call to `sendRequestResults`.

2.2.2.3 The request data type

Method Summary	
java.lang.Long	**getId**() get the Id of the EPDSSRequest
BuildingModelData	**getModelData**() get the building model data provided by the export request
AttributeListData	**getParameters**() get the variable set of parameters that describe the request
java.lang.Long	**getProjectId**() get the Id of the related assessment project
java.lang.String	**getTopic**() get a short descriptive name/topic string of the request
java.lang.String	**getType**() get the general type/target of the request ("Auditing" , "BuildingModel")
ZoneData	**getZoneData**() get the hierarchical auditing zone data provided by the auditing request
void	**setId**(java.lang.Long id) set the Id of the EPDSSRequest
void	**setModelData**(BuildingModelData modelData) set the building model data to be provided by the export request
void	**setParameters**(AttributeListData parameters) set the variable set of parameters that describe the request
void	**setProjectId**(java.lang.Long projectId) set the Id of the related assessment project
void	**setTopic**(java.lang.String topic) set a short descriptive name/topic string of the request
void	**setType**(java.lang.String type) set the general type/target of the request ("Auditing" , "BuildingModel")
void	**setZoneData**(ZoneData zoneData) set the hierarchical auditing zone data to be provided by the auditing request

The request data type defines all information to start an auditing request from the BAU-WSN system or a request for model data from the CAD system.

It provides the needed zone tree structure with zone ids to the auditing client module to establish a common zone model understanding between the EPDSS and auditing software.

The zone ids are later provided by the auditing results service of the BAU-WSN software to map back auditing measurements and results to the corresponding EPDSS zones.

2.2.2.4 The requestResult data type

Method Summary	
java.lang.String	**getAuditingWebServiceConnection**() get the address of the web service that is providing the auditing results interface
BuildingModelData	**getBuildingModel**() get the building model that is the result of the request
AttributeListData	**getParameters**() get the additional attributes that describe the request/result
java.lang.Long	**getProjectId**() get the id of the related assessment project
java.lang.Long	**getRequestId**() get the id of the related request
void	**setAuditingWebServiceConnection**(java.lang.String auditingWebServiceConnection) set the address of the web service that is providing the auditing results interface
void	**setBuildingModel**(BuildingModelData buildingModel) set the building model that is the result of the request
void	**setParameters**(AttributeListData parameters) set the additional attributes that describe the request/result
void	**setProjectId**(java.lang.Long projectId) set the id of the related assessment project
void	**setRequestId**(java.lang.Long requestId) set the id of the related request

2.2.3 The auditing results service

The auditing results service interface functionality has already been described in detail in deliverable *D4.3 Guidelines for BAU-WSN Selection and Deployment*.

The connection details necessary to access the web service are provided with the `requestResult` data that is returned to the EPDSS after the initial availability of auditing results.

3 Testing

3.1 Implementation support

The implementation of the EPDSS system is covering implementation testing strategies that support testing of the code:

- Unit functionality testing is supported with JUnit test cases which support mainly regression testing and documentation of functional usage.
- Testbed testing provides temporary dummy functionality and data to test algorithms when the full set of input information applicable to the tested functionality is not yet available. The test data and functionality will partly evolve to be part of the standard EnPROVE functionality, where data patterns and assumptions have to be generated when the respective input is missing.

3.2 Methodology

After testing the general functionality with artificial data, the algorithmic testing and acceptance testing is evolving out of the results from the first integration tests with real data. Each functional module has already described the needed input parameters, the scope handled and the results that can be expected in previous deliverables. While the unit tests will still automatically help finding bugs for not yet explored algorithmic states, the acceptance and performance testing will have to be executed manually by reviewing the results of each of the system execution phases.

3.3 On-site testing

Although the design of the EnPROVE system does not impose general restrictions or complex requirements to the system software installations and well prepared off-site testing conditions should match live testing under real conditions to a great extent, various on-site testing will not only be needed to monitor and evaluate the auditing processes but also to find shortcomings in GUI, software use cases or unit test design.

4 Deviations from previous design and deliverables

Since the BAU-WSN and the EPDSS have different demands on the capabilities of their respective GUI modules, and since from the building model only the zone object is of relevance to the auditing sensor data collection and initial processing, it has been decided to remove the graphical 2D interface functionality from the auditing dashboard and add it to the EPDSS GUI instead. This is in contrast to the previous design in deliverable *D4.1 System Dashboard*.

This eliminates the necessity to share a full blown building model between the BAU-WSN and the EPDSS while access to the building model is now supported to the auditing contractor by using the respective functionality of the EPDSS GUI.

5 Conclusion

The design of a common overall EPDSS data model consumed more resources early in the project as a module by module approach would have. However the benefits of a common data model regarding conception mismatch problems and expecting testing efforts by large outweigh an individual approach, especially during system integration.

Together with the streamlining of the two now more or less independent GUI concepts of the EPDSS and BAU-WSN, the interfacing of the two EnPROVE systems and the data exchange could be reduced to an effort mainly serving functionality without a lot of administrative or organizational overhead. As only the zone objects identification is used to map between the auditing results and the EPDSS building model and as the workflow fixes the definition of these zones prior to commencing the main auditing process, there are no model synchronization tasks to be performed between the two systems and the main auditing phase can take place without the need of additional communication between the two systems.

Since it is not yet decided if and how the integration framework will support the system validation and performance testing there might be respective functionality which will be added later to the framework.

The next development steps to be taken are to integrate the external AutoCAD, Revit and Active3D plugins into the described framework and to evaluate and tune the performances of the functional modules of the EPDSS with real data provided by the BAU-WSN.