Abstract

The paper presents a fragment of a hybrid modeling method designed for the maintenance subdomain of the ComVantage EU research project. The method as a whole addresses modeling needs for business stakeholders trying to achieve business-IT alignment with the technological specificity of the runtime prototypes of the ComVantage research project – more specifically, a front-end based on mobile apps with a back-end of Linked Data, in order to support collaborative business processes. Mobile maintenance is one of the project application areas and it has its own domain specificity, captured in the modeling method fragment described by the paper. Concepts from several layers of abstraction have been integrated in a metamodel, partially implemented in a modeling tool. Additionally, an RDF export mechanism has been designed to expose the visual models as RDF data structures, thus bridging the design-time and run-time facets of the project.

1. Introduction

The goal of this paper is to describe a hybrid modeling method designed for the domain specificity and the requirements of the ComVantage EU research project [7], with a focus on its mobile maintenance application area. Furthermore, a general takeaway message is extrapolated beyond the project scope.

As R. Maier’s comprehensive monograph on knowledge management systems [19] indicates, knowledge can be captured, represented and handled with many approaches, depending on goals and available sources for acquisition. We position our work in the areas of process-oriented knowledge management (the work at hand is process-centric) and knowledge modeling (the work supports knowledge externalization through models that are both human-oriented and machine-readable). The data model employed here to serialize the machine-readable format is RDF [34], as promoted by the Linked Data paradigm [38].

Domain-specific modeling is generally favored by business stakeholders against general purpose, IT-oriented modeling languages such as UML [25] or BPMN [24]. Although these standards are abstract enough to support the design of non-IT models (e.g. activity diagrams may depict business steps, BPMN may depict non-automated collaborative processes), they are most often employed in software engineering contexts, aiming for code generation and workflow automation (via BPEL [23], XMI [26] or XPDL [36] serializations). Hence, they are often blamed for being cryptic, excessively technical or syntactically bloated, at least notation-wise and relative to the goals of describing a business view [8] [21] [32].

This paper brings forth the notion of a modeling method, to be distinguished from modeling languages/notations such as UML/BPMN. These standards aim to express semantics visually, through the notation variability. While the visual variation determined by such explicit semantics (at least by the typing of the artifacts) is highly relevant, in the case of a modeling method we are just as concerned with non-visual (explicit) semantics, implied (non-explicit) semantics and hybrid semantics - notions that will be explained in Section 2.

Methodologically, we subsume this work under the metamodeling paradigm, concerned with the design of instruments for modeling targeted problem classes and systems. We use the multifaceted domain given by the ComVantage research project as a requirements source and we believe that modeling method engineering must embrace the same agility-oriented principles as software engineering. This means that emphasis must be placed on requirements and customization requests that come from the domain specificity. The method must adapt as these requirements change or evolve. Standardized approaches are rather rigid in this respect, acting on the generic abstraction level needed to guarantee automation and domain-independent reusability.
The research questions tackled by the paper are driven by the ComVantage requirements (to be summarized in Section 3):

RQ1. How can the business view of the mobile maintenance domain be modeled in a holistic manner that is a) process-centric, b) embeds domain specificity and c) enables communication of requirements according to the technological specificity of ComVantage?

RQ2. How can these models become resources for the ComVantage prototypes?

The paper is structured as follows: Section 2 provides clarifications on some of the working terms employed in this work. Section 3 describes the context of the ComVantage research project and formulates requirements that complement the research questions raised here. Section 4 refers to relevant related works that have been investigated and, in some cases, integrated in the proposed hybrid method. Section 5 offers an overview on the proposed modeling language and delimits its maintenance-related aspects. Section 6 discusses model samples covering the following integrated facets: motivators, processes, resources and resource access. It also provides insight regarding a model serialization component aimed to answer RQ2. The paper ends with a conclusive SWOT analysis and a statement of the intended takeaway message.

2. Working Terms

Mobile maintenance refers to one of the application areas of the ComVantage research project, where maintenance is performed on site, with mobile apps supporting the maintenance processes. The processes can be triggered remotely, by thresholds of various sensors embedded in the maintained machines.

According to the definition from [17], which also provides the framework for this paper, a modeling method is composed of three building blocks: 1) A modeling language comprising the modeling constructs grouped in language subsets called model types, each type being dedicated to a different subproblem or abstraction layer; 2) A modeling procedure comprising the modeling steps and conventions that guide the user towards his/her goals; 3) Modeling functionality comprising mechanisms and algorithms that transform or evaluate models, to be implemented in a modeling tool.

By nonvisual explicit semantics, we understand the set of attributes (defined in the modeling language) that are not visually communicated (through the notation), but are editable in the method implementation (a modeling tool) and are structured enough to be processed by the modeling functionality. In our case, this type of semantics is exposed in the model serialization as RDF data properties. By hybrid semantics we refer to the links between the different model types of the language (“inter-model links”). These links have: a) a usability function, of allowing navigation between related models in the tool (such as the hyperlinks), and b) the role of expressing semantic relations between model types. In our serialization these are exposed as RDF object properties. Implied (non-explicit) semantics can be present in the modeling procedure, in the form of conventions that make the difference between “bad modeling” and “good modeling”. An initial modeling procedure skeleton serves for method requirements elicitation (see Table 1), and a final procedure can be provided as a knowledge product: modeling guidelines. Implied semantics are also handled in the modeling functionality, in the way of model transformations described algorithmically (or through production rules running on, for example, the RDF representation of the models). Just as inference rules generate new interpretations on existing facts (in knowledge bases), model transformations can be driven by graph rewriting rules producing new interpretations (models) from source models. This is another argument that brings the notion of modeling method in the domain of knowledge systems, where the fact base is given by a model repository, and the rules are either a) prescribed in the modeling language grammar or b) described as algorithms in the modeling functionality. The RDF serialization of models, as supported by our modeling method, aims to bridge the worlds of knowledge engineering (with a focus on machine-readability) and knowledge management (seen as a type of intangible asset management, promoted in works such as [12] [14]).

Domain specificity can be captured by a modeling method in all its aspects: in the notation or the semantics of the language, in the procedure (as modeling conventions) or in the functionality. For example, what UML considers an abstract “Activity” construct will have to be enriched in maintenance process modeling with links to the types of resources it requires, with data properties relevant for users (times, costs etc.); in other words, through the nonvisual or hybrid semantics. Functionality can be built to derive new models (implied semantics) from such information.

3. Project Context and Requirements

The ComVantage research project as a whole aims to define an IT architecture to support collaborative processes in supply chains (including those supporting the delivery of maintenance services). Its technological specificity is given by: a) a front-end based on mobile
apps and b) a back-end relying on Linked Data shared between collaboration partners [22]. This must be complemented by a design-time component that includes the ComVantage modeling method. A bridge between the design-time and the run-time components is the app orchestration mechanism presented in [37], based on a model driven approach: it takes input from models with the goal of deploying an app flow driven by the designed business processes. This bridge gives to the run-time component the quality of a process-aware information system (as defined by [1]). However, this is not the case of a fully automated workflow automation system: instead of orchestrating web services in the spirit proposed by BPEL, ComVantage orchestrates mobile app support at the front-end, by extending app chaining techniques to a business process-driven approach [22][37].

The current paper highlights a fragment of the ComVantage modeling method, relevant to the mobile maintenance use cases. A key challenge for the method development is the multifaceted nature of this domain, which requires a hybrid method integrating best practices with customized aspects, multiple abstraction layers (from a high level business model down to business processes mapped on their resources) and multiple technical facets (mobile app requirements, access control etc.).

Table 1 supports the research questions raised in the Introduction: a modeling procedure skeleton describes the knowledge required to be externalized in models, from which an initial set of concepts are derived for the mobile maintenance area. Additionally, the run-time component of ComVantage has the extra requirement for model interoperability, to support the process-awareness for the mobile app orchestration mentioned earlier in this section.

**Table 1. Concerns of the mobile maintenance application area (as extrapolated from ComVantage requirements)**

<table>
<thead>
<tr>
<th>Required steps of a modeling procedure skeleton</th>
<th>Required concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. to know what are the domain-specific motivators (triggers) of the business processes</td>
<td>Configurations of sensor values for a machine with a valid service level agreement</td>
</tr>
<tr>
<td>2. to know what are the business processes</td>
<td>High level process describing how the maintenance service is delivered and what business actors are involved</td>
</tr>
<tr>
<td>3. to know how the business processes run</td>
<td>Operational maintenance processes</td>
</tr>
<tr>
<td>4. to know what the business processes need in order to run</td>
<td>Apps, data, humans</td>
</tr>
<tr>
<td>5. to limit the access to resources</td>
<td>Access policies</td>
</tr>
</tbody>
</table>

4. Related Works

General-purpose notations like UML and BPMN have been considered insufficient for our goals, with respect to a) the requirements for domain-specificity; b) the approach of agile modeling method engineering; c) the requirement for model serializations, making the models themselves a resource to be queried and linked in a Linked Data environment.

Modeling tools taking more domain-specific approaches have been investigated, mainly from the supply chain modeling domain (a listing of such tools is maintained on the Supply Chain Council website [31]) and from business process management. Most of these tools deal with the integration of generic business process designs in the high level business views (the business model, process map, the supply chain). Some of them (ADONIS [3], ADOLog [18], Metastorm Provision [20], ARIS [30]) cover also resource allocation but details are only supported for the human resources (through organizational models) while the other resource types are quite generic (e.g. systems, information) and lacking nonvisual semantics. TIBCO Business Studio [33] employs UML for customizing the resource vocabulary, thus acting on the meta level.

ComVantage deals with two particular types of resources: mobile apps (modeled at interaction level) and Linked Data information resources (for which access control requirements must be communicated). Additional domain specificity and integration with the upper (business context) and lower (resources) levels are needed for the mobile maintenance subdomain.

Some modeling approaches have been assimilated in the hybrid modeling language: a model type relies on the e3 value language [11] to describe the value exchanges of the business model; feature-oriented modeling [16] inspired the product/service structuring approach; the app requirements modeling takes inspiration from the Cameleon framework [5] repurposed for process-based requirements engineering; the access control approach uses concepts from role-based access control [9].

Regarding interoperability, most tools provide exports targeted to commercial products (e.g. MS Visio) or aiming code generation with standard XML-based serializations for business processes (XPDL, BPEL) or generic models/metamodels (XMI). We promote a serialization format that is Linked Data-compliant (hence, query- and linking-oriented) and model type-independent. It relies on the meta-metamodel of the ADOxx metamodeling platform [4] and its FDMM formalism [10], therefore it is customized in synch with the modeling method itself.

3392
5. Modeling Language Overview

The modeling language can be described as a **modeling stack** (with layers corresponding to different abstractions or delimited subproblems) – Fig. 1, and as a **structured metamodel** – Fig. 2. The stack gives a layered view on the set of model types:

The highest two layers provide means for describing the business model, the product portfolio, the logistic scope and the high level supply chain. Such models act as hubs enabling selection and navigation to lower levels through the inter-model links (e.g. from a scope model to a particular organizational unit, from a supply chain down to its operational processes). Discussions on these are not within the scope of this paper.

The third layer enables the modeling of “motivators” for the core business process execution, specific to each ComVantage application area: for mobile maintenance they are **machine defects/states** with **service level agreement contracts** (to be detailed in Section 6.1). The fourth layer is the method core, comprising the actual **maintenance processes**, the derived **resource orchestrations** and the **app interaction processes** (to be discussed in Sections 6.2-6.3). The lowest layer deals with resource requirements – what is needed to perform business processes (to be discussed in Sections 6.3-6.4).

The modeling stack can be presented with the concepts as follows:

- **Business model**: This layer covers the high-level view of the system, including the business strategies, market characteristics, and the overall scope of the project. It serves as the starting point for more detailed modeling.
- **Product portfolio**: This layer focuses on the different product offerings and their characteristics. It includes models of products and their features.
- **Scope configuration**: This layer zooms into specific configurations, focusing on the configuration options and settings for different aspects of the product.
- **Machine state configuration**: This layer deals with the state configurations of machines, including their current state and the history of state transitions.
- **Process flow**: This layer represents the flow of processes within the system, including the activities and decisions involved.
- **Resource orchestrations**: This layer focuses on the orchestration of resources, including scheduling, allocation, and management of resources across different processes.
- **App interaction processes**: This layer deals with the interaction of mobile apps with the system, including user interactions and operational processes.
- **Resource requirements**: This layer addresses the requirements for resources needed to perform business processes.

**Figure 1. A Modeling Stack for ComVantage (underlined are the model types relevant to this paper)**

For the metamodel, we use a customized language (Fig. 2) that depicts (as suggested by the legend):

- **a)** the modeling concepts grouped in model types – for example, the **Process model** type grouping the concepts **Activity, Decision** etc.;
- **b)** the relations, of two types: 1) **visual modeling relations**, for example the arrows between two subsequent activities in the same process flow; 2) **inter-model links** acting as hyperlinks, for example to navigate from a process activity to its required resources (e.g. the responsible role from an organizational model).

While a standard UML class diagram could be misused to visually capture a similar structure, its semantics would fundamentally different – we do not deal here with OO classes and associations, but with visual connectors and navigable hyperlinks between concepts. Additionally, we need customized functionality for controlling the evolution of the metamodel, for versioning, querying and serializing it (by using the same RDF export mechanism), hence a UML tool does not suffice.

The metamodel presents a deviation from the modeling stack: The **Process** model type is used with two different interpretations in the stack: a) for maintenance processes, linked through the **supported_by** relation to resources and b) for human-app interaction processes, linked through the **interacts_with** relation to user interface elements (POI, meaning “point of interaction”) described in the **Mobile IT support model type**.

The metamodel also provides the **schema for the RDF vocabulary** used in the model serialization mechanism. A full description of the mapping from metamodel constructs to the RDF Schema resources (and their URIs) is available in one of the project public deliverables ([7, deliverable D311] and will not be fully specified here.

However, we indicate the basic principles:

1. Each model type becomes a named graph (in one of the quads syntaxes [2] [6]);
2. Each modeling concept becomes an RDF class;
3. Each modeling relation becomes an RDF class if it has editable properties (visual connectors can have their own editable attributes, making each occurrence distinguishable);
4. Each modeling relation without attributes becomes a data property;
5. Each modeling attribute with atomic values becomes a data property;
6. Each complex attribute value (table) becomes an RDF list;
7. Each inter-model link becomes an RDF object property bridging different named graphs.

Serializing the models in RDF enables the use of Linked Data technologies to manipulate the knowledge captured in models. SPARQL [35] queries are able to search along the relations specified by the metamodel (e.g. “list all app features with landscape display capability required for the technician role in a particular maintenance process”). More concrete examples of what the RDF serialization looks like are given on Section 6 for concrete examples.
6. Detailed Views on the Model Types

6.1. Modeling the Maintenance Motivators

The maintenance processes are triggered by service requests that involve two aspects (Fig. 3): a) a defect occurrence and b) a service level agreement. Defects are modeled with the \textit{Machine state} model type, where a machine is hierarchically decomposed into parts and each part can have one or more sensor variables color-coded with three possible states (\textit{safe}, \textit{at risk}, \textit{critical}). A model expressing such a configuration is linked (through the \textit{recommended approach} relation visible in the metamodel) to a maintenance process prescribed for that particular sensor state combination.

Additionally: a) The maintenance process has mappings on required human roles, further linked to required skill profiles for those roles; b) The machine state model is linked to a Service Level Agreement (SLA) model, which is further linked to a particular business entity (customer). This linking contributes to positioning the work in the field of knowledge systems, as models fulfill the requirement of knowledge reuse and the knowledge management success criteria discussed in [13] and [15] (having a knowledge strategy that identifies users, processes, sources and their links).

The explicit semantics used in a machine state model include properties such as:

a) \textit{For the machine components}: type (machine or part, to enable the decomposition of complex machines), lifetime and production year, price, reliability, actuator table (to indicate the settings under which the state is described), defect history. Additionally, visual relations describe the decomposition (a “part-of” hierarchy of machines and
machine parts) and the assignment of sensors to each part.

b) For the sensor variables: measurement unit, status (safe, at risk, critical), threshold limits assigned to each status, URI (in case the current sensor value is directly retrievable, meaning that the model depicts the current state and not a hypothetical configuration).

For the sensor variables:
- measurement unit,
- status (safe, at risk, critical),
- threshold limits assigned to each status,
- URI (in case the current sensor value is directly retrievable, meaning that the model depicts the current state and not a hypothetical configuration).

Figure 3. Inter-model linking for maintenance motivators

The explicit semantics used in service level agreement models are reflected by properties such as:

a) Service level agreements are also decomposed in reusable “terms” of two types: quantifiable and non-quantifiable. The properties of an SLA object capture those terms that are usually not reused: validity interval, price, termination, and transition conditions. A visual relation indicates the reuse of terms from other SLAs;

b) For a reusable quantitative term, the relevant properties are: property (what the term refers to, e.g. “Response time”), condition (what the term promises, e.g. “<5ms”), quantitative success goal (e.g. the term holds for >99% of service requests), priority (for the service provider) and compensation value (if the service provider does not fulfill it). A non-quantifiable term is any term that does not have the quantitative success goal specified.

6.2. Modeling the Maintenance Processes

The maintenance process models have domain specificity captured in the nonvisual semantics (editable properties like costs, times, wastes), hybrid semantics (links to other domain-specific models, such as the machine state/SLA models and resources). They rely on swim lane modeling conventions to express the following patterns identified in the maintenance use cases (Fig. 4): a) Delegation of work (role switch in control flows); b) Waiting for results after delegation (role switch with an absence of subsequent arrow on the same swim lane); c) Delegation followed by parallel work and resynchronization (parallel splits and joins).

These patterns are easily detectable in the RDF export of the model via SPARQL queries, thus enabling process awareness for ComVantage run-time components. For example the delegation pattern would generate the following code (here roles are represented by companies):

\[
\text{Company1} :\text{contains} :A1. :A1 :\text{Activity}.
\text{Company2} :\text{contains} :A2. :A2 :\text{Activity}.
\text{Arrow1} :\text{from} :A1; :\text{to} :A2.
\]

A run-time component stepping through the process and arriving at activity :A1, will use the following query to get the next flow steps, their types (activity, decision, parallel split) and swim lanes:

\[
\text{SELECT} \ ?\text{step} \ ?\text{swimlane} \ ?\text{type} \\
\{ ?r :\text{from} :A1; \text{to} ?\text{step}. \\
?\text{swimlane} :\text{contains} ?\text{step}. \\
?\text{step} \ ?\text{type} \}
\]

Similar queries can further traverse the inter-model links to other model types (e.g. motivators, resources).
Figure 5. Resource orchestrations derived from business process requirements

Figure 6. Policy example for a particular resource (and its RDF serialization)
6.3. Modeling the Resource Requirements

Each activity of a maintenance process model is further linked to: a) roles (from organizational models); b) information resources; c) app features and d) hardware resources (from the Object repository).

For each resource type, a resource orchestration model can be derived automatically using an implementation of several graph rewriting rules. The results indicate the order of involving resources along a maintenance process. Fig. 5 shows examples of role, information and app orchestration models, as derived automatically from a simple maintenance process. (the vertical swim lanes suggest what was initially linked to each of the process steps). The graph rewriting rules have been specified in [7, public deliverable D311].

For information resources, the orchestration can justify the use of Linked Data, highlighting when data is needed from different URIs or organizations, a typical use case for SPARQL queries.

For apps, the orchestration is additionally filtered by role, in order to derive the flow of apps required by a particular role during a particular process. Assuming that a person cannot perform two tasks in parallel, a split in the app orchestration models should be interpreted as an OR. An app supporting the OR split will display the list of encountered paths, allowing the user to choose one or more. The RDF serialization of this model type is the input to the orchestrated deployment of run-time apps described by [17].

As it is reflected by the metamodel (Fig. 2), the mobile app requirements can be modeled on three levels of detail: a) as mobile app/feature objects (from the Object Repository) whose nonvisual semantics include device type, operating systems and a selection of capabilities (portrait, landscape, night/day switching, voice/touch interaction modality); b) as Mobile IT support models, which are navigation maps describing screens and how they trigger each other; c) as user-app interaction processes describing the flow of interactions with the screens, hence suggesting initial rudimentary mockups; for this, the same process notation is employed, with different hybrid semantics (links to features/UI triggers instead of resources).

6.4. Modeling the Access Control

Access control semantics are attached to information resource objects from the resource pool (Object Repository), to state who can access them.

The explicit semantics of information resources include a type (Linked Data, Message, Physical, Peripheral) suggested with visual cues (see Fig. 5), a source URI and a SPARQL query (if they can be retrieved from a SPARQL endpoint).

In addition, the semantics include a table (see top of Fig. 6) that allows the linking of roles (from the organizational models), actions (read, write or both) and additional constraint descriptions in natural language. In the example from Fig. 6, a service technician role (SvTn) gets Read access to the current modeling object (the information resource “Machine sensor values”) with the additional comment that only machines assigned to him are available. Such data can be seen in the RDF graph derived from the Resource Pool (bottom of Fig. 6) and can be used by the run-time policy implementers.

The next step is to assign the information resources to the process steps, thus indicating where they are required. Since the roles should already be assigned to the same process, a cross-check can be performed along the inter-model links between a) what role was assigned to an activity and b) what roles are allowed to use the resources required for that activity. This check is relevant since it is expected that Resource pools are modeled by resource owners, while the maintenance processes are designed by a central maintenance coordinator, using the same modeling tool and sharing the models. This is enabled by the current implementation on the Open Model Initiative portal [27], where partners access remotely the same tool and can develop the models in a collaborative fashion.

7. SWOT Evaluation and Conclusions

A partial implementation of the method is available in the Open Model Initiative [27], for experimentation and validation purposes. It currently covers the maintenance processes, their resource and access requirements (including the resource orchestration derivation) and the RDF export mechanism. The run-time component of ComVantage is process-aware through the app orchestration framework that takes its input from these models (as described in [17]).

The rest of the method is still evolving as domain semantics are being refined and periodical change requests are integrated. A SWOT evaluation performed on the method design and its partial implementation produced the following conclusions:

Strengths: Compared with the baseline given by the investigated tools and methods, the work of this paper is more integrative (across multiple facets) and more specific (deeper specializations of resource types, process motivators and object properties). The integration is further transported in the Linked Data environment, where inter-model links are turned into semantic RDF relations, a feature unavailable in the investigated base.

The work promotes the approach of modeling method engineering to capture domain specificity,
bringing modeling instruments closer to the business view (compared to general purpose standard notations) and evolving them iteratively.

**Weaknesses:** Due to the emphasis on domain specificity, there is a risk that the method is perceived as having narrow scope and limited use (within the boundaries of ComVantage). However, it integrates multiple abstraction layers and aspects that can be used independently, either by ignoring or by using selectively the model linking (the ComVantage domain is not narrow, but multifaceted). The hybrid method has the reusability of a controlled vocabulary, shared within a specialized community with a relatively wide scope.

The limitations currently tackled involve more detailed modeling of mobile app requirements, with respect to usage patterns. There are no plans to generate app code; the final aim is to support process-driven app selection.

**Opportunities:** The RDF export mechanism opens opportunities for bridging the metamodeling and the Semantic Web worlds - for example, an ontology skeleton could be derived from the RDF version of the metamodel; or, model transformations (such as the resource orchestration) can be performed outside the modeling tool (with rule engines, CONSTRUCT SPARQL queries etc.).

**Threats:** Users accustomed with standardized notations, especially from the field of software engineering, may prefer standard modeling approaches with wider tooling support. However, we promote here an agile evolution of modeling approaches driven by change requests/requirements and a gradual understanding of the domain semantics. For this, we rely on the Open Model Initiative and Laboratory [28], where customized modeling methods are open to interested communities that further develop them and share models beyond the project goals.

The research questions raised in the Introduction have been answered as follows:

**RQ1.** The requirements expressed in Table 1 can be satisfied with the presented modeling method. The method reflects domain specificity in the explicit semantics (properties), the hybrid semantics (links to what triggers and what is required to perform maintenance processes) and in the semantics implied by the resource orchestration (as derived from the resource requirements). The orchestration models cross the gap, as requirements, from the business view to run-time implementations.

**RQ2.** The RDF export mechanism turns models in Linked Data resources that are shared for queries, searches, transformations and linking outside the modeling tool. Model-awareness (and, specifically, process-awareness) is thus enabled for prototypes such as the app orchestration deployment engine.

The work draws a distinction between general purpose modeling languages / notations and the more complex notion of modeling method. The generalized takeaway message is that agile engineering of modeling methods is a key necessity for bringing a modeling method closer to the business view, as new insight about the domain specificity emerges. Therefore we promote an alternative to rigid abstract standards, emphasizing the importance of specialized communities with evolving requirements assimilated in the model semantics.

### 8. Acknowledgment

The research leading to these results was funded by the European Community's Seventh Framework Programme under grant agreement no. FP7-284928 ComVantage.

### 9. References


[38] ***, Linked Data resources - http://linkeddata.org/.