Technische Universiteit
Eindhoven
University of Technology

Semantic Concepts in Product Usage Monitoring and Analysis

Mathias Funk, Anne Rozinat, Ana Karla Alves de Medeiros,
Piet van der Putten, Henk Corporaal and Wil van der Aalst

ES Reports
ISSN 1574-9517

ESR-2008-10
29 September 2008

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems

http://www.es.ele.tue.nl/esreports
esreports@es.ele.tue.nl

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems
PO Box 513
NL-5600 MB Eindhoven
The Netherlands

Semantic Concepts in Product Usage Monitoring and Analysis

Mathias Funk[1], Anne Rozinat[2], Ana Karla Alves de Medeiros[2], Piet van der Putten[1], Henk Corporaal[1], and Wil van der Aalst[2]

[1] Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands
{m.funk,p.h.a.v.d.putten,h.corporaal}@tue.nl
[2] Information Systems Group, Eindhoven University of Technology, The Netherlands
{a.rozinat,a.k.medeiros,w.m.p.v.d.aalst}@tue.nl

Abstract. Nowadays, complex electronic products, such as DVD players or mobile phones, offer a huge number of functions. As a consequence of the complexity of the devices, customers often have problems to use such products effectively. For example, it has been observed that an increasing number of technically sound products is returned due to, e.g., interaction problems. One possible root cause of this problem is that most product development processes are still too technology-driven, i.e., potential users are brought into contact with the product only at a very late stage. If early consumer tests are carried out, then these typically aim at abstract market evaluations rather than formulating concrete requirements towards the functionality of the product. As a result, products often have little meaning or relevance to the customers. Therefore, we need better ways to involve users in the development of such products. This can be achieved by observing product usage in the field and incorporating the gained knowledge in the product creation process. This paper proposes *an approach to build automatic observation modules into products, collect usage data, and analyze these data by means of process mining techniques exploiting a novel semantic link between observation and analysis*. This link yields two main benefits: (i) it adds focus to the potential mass of captured data items; and (ii) it reduces the need for extensive post-processing of the collected data. Together, these two benefits speed up the information feedback cycle towards development.

1 Introduction

Complex electronic products, both for private consumers and professional users, are hard to specify and design as no real information is available about the potential customers' expectations and needs. Meeting these expectations is, however, crucial as nowadays customers can choose among a wide variety of products, and will more easily reject products that do not suit their needs. A symptom of this problem is, for example, that an increasing number of technically sound products is being returned [1]. At the same time, it is not possible to perform lengthy user studies as there is a strong pressure on 'time to market'. Moreover, it is difficult to gradually improve products by incorporating user feedback from the field as often only very few generations of the same product are made (to be replaced by new, more innovative products). In short, customers

are becoming more demanding, while product development must be done with fewer iterations.

One way to ensure that products will suit the needs of their potential customers is to involve them as early as possible in the development process. This can be achieved by letting potential users test early prototypes containing important functions, and to incrementally incorporate the gained knowledge into the product under development. However, to make this approach applicable in practice, two conditions need to be fulfilled.

1. It needs to be *feasible* to perform the tests in the first place, i.e., it should fit into today's challenging development cycles.
2. The collected test data needs to be *useful*, i.e., valid ("Does this reflect our potential customers?") and relevant ("Is this what we want to know?").

To address the first condition, the test data needs to be collected and fed back to the development team as automatically as possible. As we will demonstrate later, *our approach is supported by a tool chain that allows for seamless data collection, processing and analysis with a high degree of automation*. Addressing the second condition is more difficult as it depends on a variety of parameters. For example, to obtain valid data one needs to choose test users that reflect the actual target group. However, one common problem is that early user tests are often performed in non-representative environments, and that people do not behave normally as they feel observed. *Our approach allows for data collection from testers using the product in their habitual environment*. For example, test products are given to users who unpack, install and use the devices in their home environment. The products themselves record usage information and automatically deliver it to the respective development unit in the company. This way, tests can easily run several weeks, and thus cover different phases of use [2]. For example, the long-term usage behavior is often quite different from the behavior in the first few hours after unpacking the product. Finally, to ensure that the right data is collected, *we allow the observation logic to be changed dynamically by the development team, i.e., while the test is running*. This way, truly iterative data collection and analysis becomes possible. Furthermore, a visual approach to specifying the observation logic is taken to make it accessible to the (mostly non-technical) people that have an interest in the data collection process. These are, for example, product managers, quality engineers, interaction designers, or user interface developers.

With the aim to further increase both the feasibility and the usefulness of product usage observation, we extend the above-described approach by an important aspect: in this paper, we establish a semantic link between the observation and analysis phase. More precisely, we allow to semantically annotate the logged data during the specification of the observation logic, and these semantic annotations are preserved and actively leveraged in the analysis phase (by *semantic process mining techniques*). So-called *ontologies* [3], which are representations of a set of concepts within a domain and the relationships between those concepts, is used to define these semantic aspects. To allow different views on the data, multiple ontologies can be used to "tag" the observed data with orthogonal concepts at the same time. As a result of the logged data being preprocessed and structured using high-level concepts, there is no need for extensive and

time-consuming post-processing of raw data. Instead, the data can be analyzed directly and in a more efficient way. An ongoing case-study within a large Dutch electronics company shows the applicability of the approach. First results indicate that, on the one hand, semantic information can indeed reduce the need for extensive post-processing and, on the other hand, improve the quality of product usage information.

The remainder of this paper is organized as follows. After pointing at related work (Section 2), we show an overview of the approach and describe how ontologies are used to link the different system components (Section 3). Then, we describe the observation system (Section 4) and potential analysis approaches for semantic data (Section 5) in more detail. Afterwards, we introduce an industrial case study that is currently performed (Section 6). The paper ends with an outline of possible future steps and a conclusion (Section 7).

2 Related work

Uses of remote product monitoring have been reported before as described in [4–8]. However, these approaches assume information stakeholders capable of programming and willing to use programming paradigms to achieve the sought-after data. Also, our approach towards "product observation", as we call it, differs from [4–8] in the sense that the integration of observation functionality into the target system is tackled and described in a software engineering process which is, in our opinion, necessary for widespread use. On the technical level we rely on the proven model-driven engineering approach, but also try to apply more agile modeling techniques like model interpretation [9] that allow for dynamic adaptation of runtime systems without the need for dynamic compilation support. While previous work describes our product observation approach in more detail [25–27], this paper focuses on the novel semantic link between observation and analysis, and we explain the extension of visually-defined observation specifications by semantic concepts. In this sense, the paper is also related to the areas of visual languages [10] and domain-driven design [11].

The idea of using semantics to perform analysis of processes is not new [12–16]. In 2002, Casati et al. [12] introduced the *HPPM intelligent Process Data Warehouse (PDD)*, in which taxonomies are used to add semantics to process execution data and, therefore, support more business-like analysis for the provided reports. The work in [13] is a follow-up of the work in [12]. It presents a complete architecture for the analysis, prediction, monitoring, control and optimization of process executions in Business Process Management Systems (BPMSs). This set of tools is called *Business Process Intelligence (BPI)* suite. The main difference of these two approaches with respect to ours is that (i) taxonomies are used to capture the semantic aspects (in our case, ontologies are used), and (ii) these taxonomies are flat (i.e., no subsumption relations between concepts are supported). Hepp et al. [14] proposes merging Semantic Web, Semantic Web Services, and Business Process Management (BPM) techniques to build Semantic BPM systems. This visionary paper pinpoints the role of ontologies (and reasoners) while performing semantic analysis. The papers by Sell et al. [16] and O'Riain et al. [15] are also related, because the authors use ontologies to provide for the semantic analysis of systems. The main difference is the kind of analysis supported, since their work

can be seen as the extension of OLAP tools with semantics. The work in [15] shows how to use semantics to enhance the business analysis function of detecting the core business of companies. This analysis is based on the so-called Q10 forms. The work in [17] explains how the analysis performed by process mining techniques can benefit from semantic information. Process mining techniques can provide a wide variety of feedback information about the usage of a system. This feedback is based on the data registered in event logs (like the performed tasks, the users, timestamps etc.) [18]. Actually, our approach is based on the semantic process mining techniques discussed in [17, 19]. However, the works in [17, 19] do not present any real-life application of the semantic process mining tools. In this sense, our paper is the first one to use semantic process mining techniques to analyze processes based on product usage.

3 Approach

Direct product information (i.e. the recording of the actual usage of a system) is potentially of use to a large group of professions, not even only in the industrial context: knowledge engineers, product managers, requirements engineers, developers, interaction designers, and other information stakeholders can benefit from such information. In the following we will use the term *domain expert* to refer to a representative of this group. Note that the members of this group are traditionally not so much involved in the actual development process of a product, but have a rather modest influence during some phases of the product creation process only. Especially in the development of innovative products, this strict separation is partially released as the expertise of domain experts is needed in the product creation process. These experts are the target users for our approach: initially, they might have a vague understanding about what should be observed in the product to answer open quesitons, but iteratively it is possible to map issues to observable items within the product, and finally, to obtain comprehensible and reliable information.

In the following, first the general approach is explained (Section 3.1) and it is shown how an automatic chaining of observation and analysis supports iterations. Then, we describe how ontologies are used in the context of product observation and analysis (Section 3.2) to further improve the data quality.

3.1 Overview

The system we propose is a *combination of a logging framework and a process mining tool*. On top of that, one or more ontologies are used to link collected data items, hence, to connect observation and analysis on the information level. Figure 1 shows an overview of the system infrastructure, depicting both the flow of tasks and the necessary tools. The figure shows that ontologies, as the main semantic linking of data, are connected to three steps of the flow. Therefore the definition and maintenance of one or more ontologies should be a concurrent task that accompanies the depicted flow. What is captured in this semantic layer shall be described in the second part of the approach section.

The *first step* of the actual flow is the observation specification: domain experts visually define what information should be observed in the product and how this information should be represented in the central data storage. Then, concepts from the ontology are intergrated into the basic information processing flow. This task is done within an easy, but formal visual language. The outcome are observation specifications which are used to automatically and remotely instruct observation modules situated in products given to testers at home. These modules start collecting field data in the *second step*: observation. The semantic annotations of the observation specifications enable the observation module to categorize the captured data accordingly on-the-fly. This results in log data with an inherent semantic structure. The *third step* of the flow (data conversion) deals with an extraction of log data stored in a data base. A special tool called ProM*import* [20] condenses the log data into an XML file that is directly usable in the final analysis step. The conversion makes use of a newly developed plug-in specialized to the needs of the semantically annotated product usage log data. In the *final step* (data analysis) the data is processed using various (semantic) process mining plug-ins which provide different views on the aggregated data. These plug-ins are part of the ProM tool [21]. This last step offers the possibility to extract the essence out of a potentially huge data set. Furthermore, it helps to present this information in a comprehensive and directly usable way.

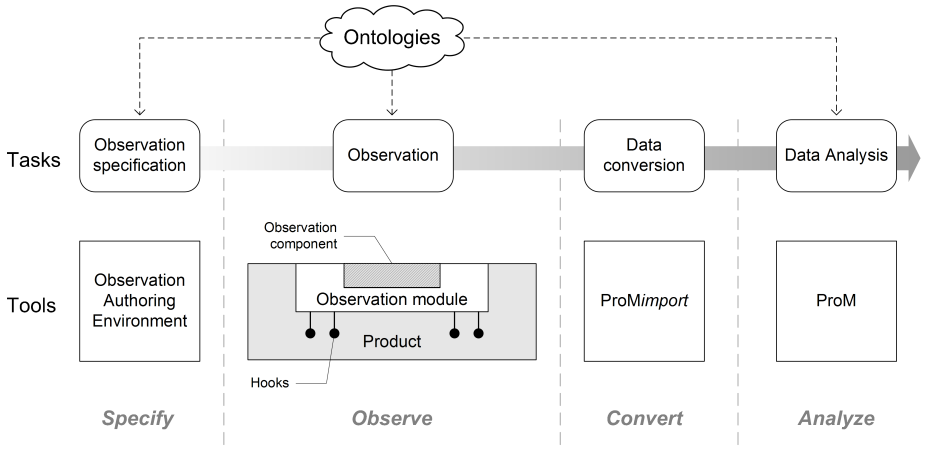

Fig. 1. System overview

Although the automatic processing chain from observation to analysis consists of several independent parts, it now becomes clear that a common connection is feasible by using ontologies for a semantic content structure. The whole process is of a strongly iterative nature. Cycles, especially between definition of ontology, observation specification, observation, and analysis, are not only expected but encouraged to achieve the most reliable and accurate final picture of product usage.

For instance, during the observation phase, the domain expert might come across unexpected information that needs a special treatment and the extension of the connected ontology with new concepts. These changes can be carried out directly and lead to an immediate improvement of the quality of collected data.

3.2 Ontologies

Ontologies [3] define the set of shared concepts necessary for the analysis, and formalize their relationships and properties. Ontology elements are organized in a directed graph and there are several formalisms to build ontologies such as OWL [22] and WSML [23]. In our approach, we are using WSML to formally define concepts because this is the language supported by the semantic process mining tools. Currently, we employ ontologies in a rather syntactic manner, whereas the full semantic power of ontologies can leverage even better analysis and will be part of a future version.

In principle, there are two connections between both systems: ontologies on the conceptual level and log data on the information level. Ontology concepts appear in the log data whenever a semantically annotated event is logged. Indeed, this link between conceptual level and information level is the goal of the approach. Obviously, a careful specification of one or more ontologies is crucial, also for the communication between the people involved in product monitoring.

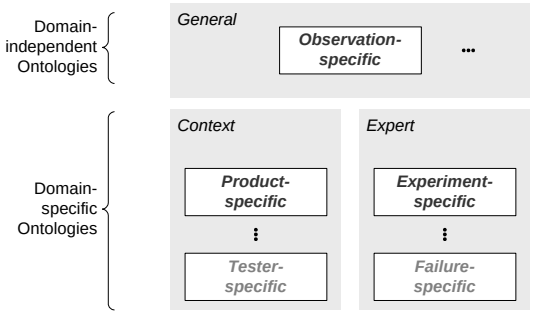

Fig. 2. Different types of ontologies can be identified for product usage observation

In our product usage observation approach we identify three types of ontologies: *general*, *context* and *expert* (cf. Figure 2). These types can be characterized as follows.

General General ontologies are domain-independent and they are used to capture concepts that are neither product not experiment related. An example for this type of ontology is an *observation-specific* ontology that contains concepts relating to the data collection process, but does not say anything about the product that is observed, or the kind of analysis that should be performed. General ontologies are expected to be highly re-usable for a couple of experiments without changes.

Context Context ontologies provide information about the setting of an experiment. In other words, they might characterize certain aspects of the product to be observed (i.e., *product-specific*), the habitual context of actual product use, or the people that perform the tests (i.e., *tester-specific*). The applicability of these ontologies may be limited to a certain domain or product group, but they can be re-used across different experiments within that scope.

Expert Expert ontologies are related to specific analysis purposes and are bound to expert views on a product, a single study or specific categories of collected data. They may be needed to answer particular questions for a certain experiment (i.e., *experiment-specific*). However, we can also think of certain domain-expert views, such as the quality engineer focusing on product failures (i.e., *failure-specific*). In principle, expert ontologies could be re-used across different product groups.

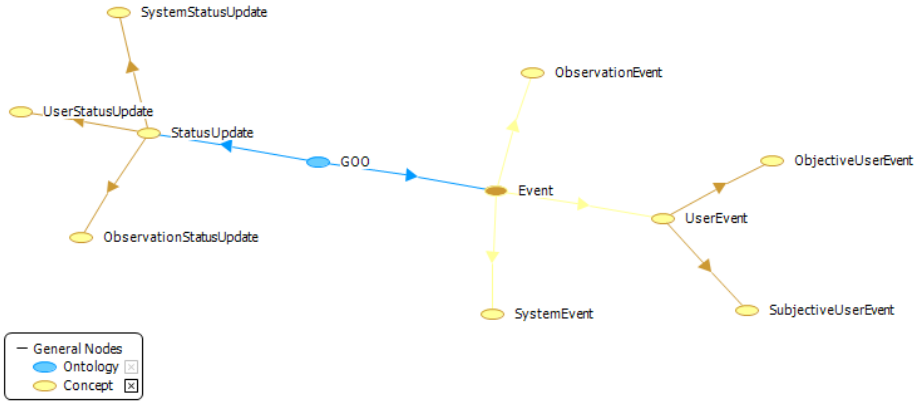

Fig. 3. Graphical representation of the Generic Observation Ontology (GOO) in the WSMT editor [24]

The differences between these three categories can be clarified by an example. Consider the *observation-specific* ontology depicted in Figure 3, which concentrates on the process of information collection itself. It contains concepts that help to distinguish between 'events' and 'status updates' in the observation. The observation-specific ontology also captures the origin of such an event or status update, i.e., whether it is user-initiated, system-initiated, or triggered by the observation system.

Events are atomic occurrences which may transport data. They are triggered by the observed system, the user, or even by the observation system. Examples of events are user interface actions, such as button clicks or menu selections (`ObjectiveUserEvent`), or system notifications (`SystemEvent`). Other examples are notifications about an update of the observation specification (`ObservationEvent`) and feedback actions initiated by the user (`SubjectiveUserEvent`). Note that with respect to user events in this paper we mostly consider objective user actions (i.e., related to actual product usage).

However, in the observation-specific ontology we already distinguish subjective user events, which relate to perceptions and feelings of the user orthogonally collected to the actual usage process itself. We briefly outline the purpose and opportunities by combining these objective and subjective data in Section 6.2.
In contrast, status updates are samples of a continuous stream of ever changing data. Status updates have in most cases the distinctive nature of periodicity. For example, the observation system triggers a certain hook to report the network activity or memory usage every 10 seconds, or when it changed by a pre-determined amount. Thus, the originally continuous stream of data is quantized. In addition, status updates can be taken on demand, e.g. only when a certain system function is used, its performance is observed.

While an observation-specific ontology formalizes properties of the data collection process, the product- and experiment-specific ontologies both describe the content of the collected data, i.e., their formalisms are dependent on the (product) context or expert domain, respectively. In the case of the *product-specific* ontology, details of the product context like product features, hardware and software components, or interaction capabilities are formalized. An excerpt of such an ontology is depicted in Figure 5. *Experiment-specific* ontologies contain semantic concepts that are just relevant in the scope of the experiment. For example, the product manager might want to find out to which extent certain categories of functions are being used throughout the test period, and for this she would create a custom view that groups these (product-specific) functions into the desired classes.

Note that multiple ontologies are used because the semantic observation and analysis is not done by one person alone. A team of domain experts should be able to work together, and to benefit from each other's insight into product usage. Therefore, many (potentially orthogonal) views on the topic have to be combined in an efficient way.

4 Product usage observation

Product observation consists of two parts: (i) instrumentation of products to collect data, and (ii) the possibility for remote and on-the-fly changes of data collection and processing. The latter aspect emphasizes also the difference between *logging* and *observation*: while logging collects as much information as possible for offline analysis, observation is an interactive process that happens mainly online. Observation tools enable a continuous revision of collected information by means of real-time visualizations, and accordingly, allow to constantly adapt observation mechanisms in order to refine the resulting data, whereas logging has to rely on the initial guess on the amount of required information. The consequence of traditional logging approaches are uncertainties that do not match the idea of fast and reliable data.

Product observation tackles the first of the two introductory conditions (cf. Section 1): observation *enables* the collection of product usage data. The incorporation of semantic information into the process of observation specification and data collection fulfills the second condition: the provision of meaningful information. The first part of this section (Subsection 4.1) shows the architecture of our observation system and how it works. The second part (Subsection 4.2) concentrates on the visual observation spec-

ification language. This language enables the aforementioned stakeholders of product information, such as the knowledge engineer, to "program" the observation system.

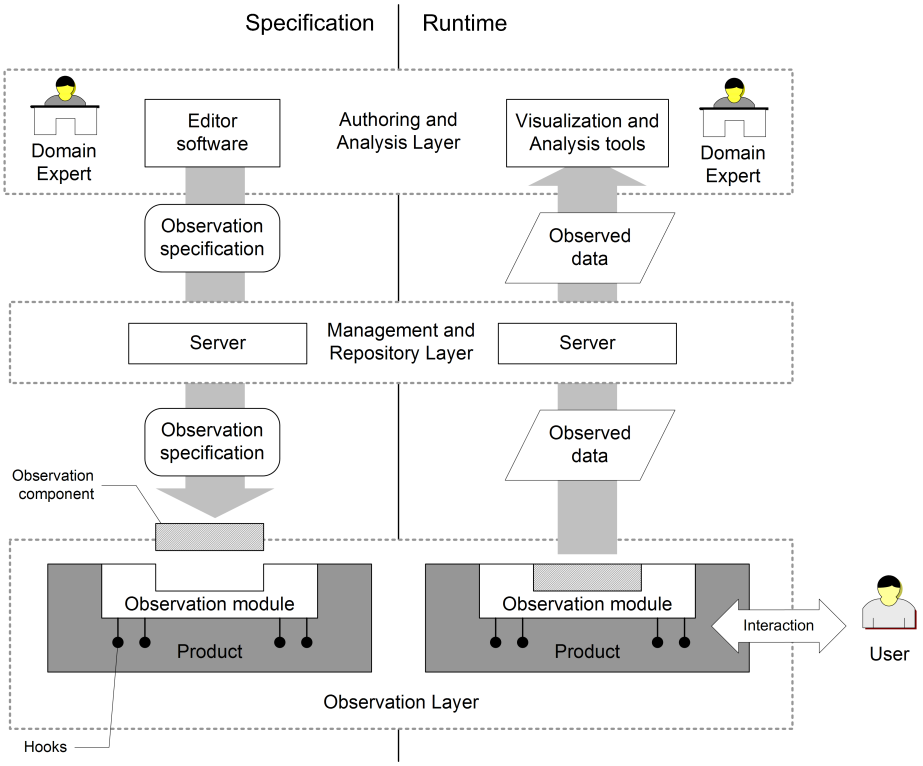

Fig. 4. Observation system architecture

4.1 Observation architecture

An observation system can be described as a layered architecture with three main parts: (i) the *authoring and analysis layer* where information collection is specified and outcomes are analyzed, (ii) the *management and repository layer* (MRL) acting as a middleware for specification and data storing, and (iii) the *observation layer* where the actual data is collected. Figure 4 shows an overview on the system and also what happens during the specification and runtime phases. The domain expert specifies what should be observed and how the acquired data should be represented in a visual language using the graphical editor in Figure 5. This observation specification (OS) is then transferred to a server on the MRL which in turn will distribute the OS to product instances. Those product instances reside in the user's home environment and connect via a communication channel, like the Internet, to the repository layer's server. As soon as the product is

configured based on an OS, the process of product usage observation starts, and the collected and processed data is delivered back to the server on the MRL. This data stored in the server can then be retrieved for analysis purposes.

We have developed the D'PUIS (Dynamic Product Usage Information System) as a platform-specific realization of the approach [25]. This system consists of the following parts: (i) a visual editor to specify OSs, (ii) a web application that distributes OSs and provides storage for collected product data, and (iii) an observation module which is integrated into product instances. An infrastructure connects these parts and enables an automatic flow from observation specification to actual product usage data. Since the focus of this paper is on semantic annotation and analysis of data, a detailed explanation of system integration details is not provided but can be found in [26, 27]. The next subsection describes the graphical editor and its visual language, which are part of the system part in (i).

4.2 Visual language

In the context of semantically supported data collection, an interesting part of the observation system is the visual language being the place where semantic links between data items are initially constructed. The visual language is a domain-specific language for observation specification and is intended to be used by domain experts (cf. Figure 5). One of the crucial aspects is the abstraction from low-level programming and system engineering matter. The language abstracts also from data base persistence, data conversion, concurrency issues and the potential synchronization problems that arise in large distributed systems. The language elements allow to connect data sources to processing blocks, to cache data for aggregation and to export the acquired information to the repository layer where it is stored and can be accessed for analysis (cf. Figure 4).

The first experiments with this kind of expressivity have been promising and a huge amount of data was collected. However, as indicated earlier, the tremendous effort needed to filter out the relevant information led to the conclusion that not only reduction of data, but also a more elaborate linking between data items was needed. Therefore, the visual language was extended in order to represent concepts from a linked ontology. In practice, this link is maintained during data collection and processing until a data item is finally stored as a log entry on the server. The simple addition of meta data, let it be product-, observation- or experiment-related, helps in the analysis phase to classify and categorize data according to the intended meaning (as expressed by the attached concepts).

Figure 5 shows an illustrative example of the graphical editor that provides authoring support for the visual language. The semantic connections are visualized as rectangles with a dashed border connected to other blocks via equally dashed arrows. The direction of the arrows expresses that semantic concept are *attached to* the flow of information in the specified observation behavior. In the foreground of Figure 5, an excerpt of the ontology is depicted which is used for the semantic annotations.

Data that is acquired in the described way is not only more meaningful, but also it is *self-contained*. This is an important step forward as all the (usually implicit) information about the observation process, such as the characteristics of the observation environment, and the nature of data sources, is *explicitly* stated in a *machine-readable*

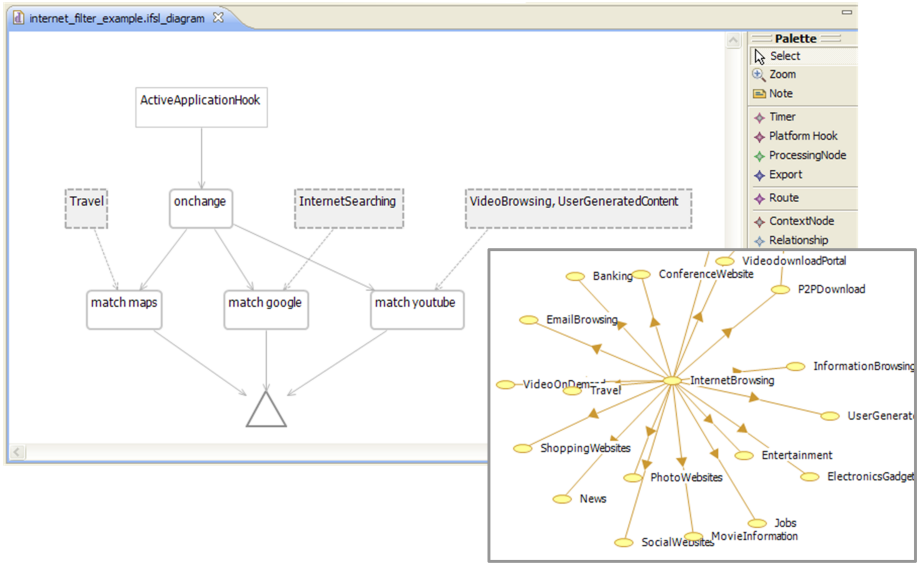

Fig. 5. Visual editor for observation specification with connected ontology, note the "travel" concept being linked from ontology to observation specification in the background

form. In the analysis phase, specialized semantic process mining techniques can exploit such information efficiently.

5 Semantic process mining

Semantic process mining uses semantic information to lift the analysis provided by current process mining techniques to the conceptual level [17, 19]. Process mining techniques support various types of analysis based on the behavior registered during the execution of some process [18]. This behavior is registered in the so-called event logs. These logs contain data about the steps (or tasks or activities) that have been performed, by which performer, at which times, using which data fields, etc. In the setting of our work, a step could be a functionality of a product that has been started by an end user, or an answer to a feedback questionnaire. Based on the data in event logs, different types of models can be automatically discovered by process mining techniques. One example are models that show the control-flow structure of a given process. As an illustration, Figure 6 depicts two automatically mined process models for an event log containing test executions. These models actually both portray the typical order that users follow while using certain products, but they reside on different levels of abstraction. As we will explain later, the possibility to mine models on different levels of abstraction is one of the benefits of using ontologies.

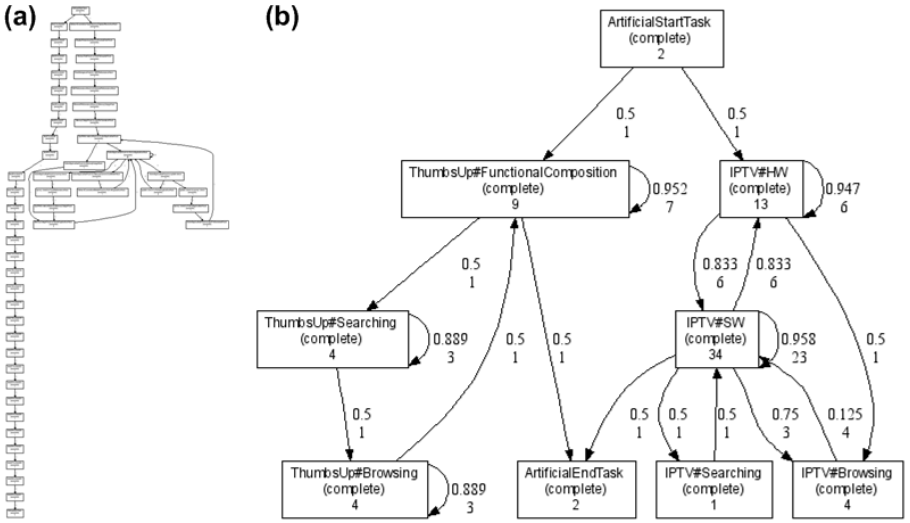

Fig. 6. Two mined models for an event log generated by our system. Model (a) is made using all events in the log and no semantic information. Model (b) provides a more abstract view of this same process. This view is based on the semantic information in the log.

As can be seen in Figure 1, after the observation phase we need to convert and analyze the data. In the remainder of this section, we first describe the data conversion phase (Section 5.1) and then the data analysis phase (Section 5.2) in more detail.

5.1 Data Conversion

The process mining techniques are freely available in the ProM framework [21]. This tool has a common input format, called *M*ining *XML* (MXML), which supports the representation of the different elements in event logs. However, the MXML format is purely syntactic. In other words, the process mining techniques cannot reason over the concepts behind the labels in the MXML files because this notion is simply not there. That is why semantic process mining techniques have been developed. They have extended the MXML format in such a way that elements in these files can be annotated with concepts in ontologies [17, 19]. This new format is called *S*emantically *A*nnotated *MXML* (SA-MXML). Furthermore, new semantic process mining plug-ins have been developed to benefit from the extra semantic layer provided in the SA-MXML log. In a nutshell, the SA-MXML format supports the linkage of syntactical elements in the previously existing MXML format with one or more concepts in ontologies. As explained in [19], this linkage is used by the semantic process mining tools to perform *subsumption* reasoning and, therefore, provide for the analysis of event logs at different conceptual levels.

ProM*import* [20] supports the development of plug-ins to convert logs from different (commercial) systems to the (SA-)MXML log format used by ProM. Just like ProM,

ProM*import* is also open-source and it is freely available at www.processmining.org. We have developed a ProM*import* plug-in that automatically extracts the recorded data from the D'PUIS database and converts them to the SA-MXML format. Note that this data conversion preserves the semantic annotations collected during the observation phase for analysis.

5.2 Data Analysis

As indicated in [28], seven semantic process mining plug-ins have been added to the ProM tool so far. From these, three are especially useful for our approach: *Ontology Abstraction Filter*, *Performance Metrics in Ontologies* and *Semantic LTL Checker*.

The *Ontology Abstraction Filter* plug-in supports the mining of process models at different levels of abstraction. The desired level of abstraction is determined by selecting or deselecting concepts linked to events (the actual instances of these concepts) in logs. The selection is based on a view of the ontology that shows only the concepts (and all of their super concepts) that are referenced in a log. Figure 6 illustrates this. Note that the model in Figure 6 (b) has a higher level of abstraction than the model depicted in Figure 6 (a). This is the case because, by using the *Ontology Abstraction Filter* plug-in, we have mapped the events in the log to some of their super concepts. For instance, we have aggregated all instances that link to some form of *internet browsing* (cf. ontology in Figure 5) to the super concept *IPTV#Browsing*. This is possible because the use of ontologies allow us to associate the concept *browsing* to some of the labels in the log.

The *Performance Metrics in Ontologies* provides feedback about (i) the processing times of tasks (or events) and (ii) throughput times of process executions. In our approach, the feedback in (i) is particularly important because it indicates how much time users typically spend in using certain functionalities of products. Moreover, this plug-in also shows how frequently instances of a given concept have been performed. Figure 7 contains a screenshot of this plug-in in action. Note that the coloring of the concepts in the ontology is based on the frequency of instances. From this graph, it is very intuitive to spot that software functions (see entry for concept *SW* at the right-side table) are more frequently used than hardware (see concept *HW*) ones.

The *Semantic LTL Checker* plug-in can be used to audit properties in the log. For instance, we can use this plug-in to check if the end users are always answering some feedback question after using certain functionalities. The novelty here is that these properties are defined based on the semantic annotations. So, the analysis and re-use of these properties is more robust. All these semantic plug-ins use reasoners [28] to infer the subsumption relations needed for the semantic analysis.

Finally, note that—due to its plug-able architecture—ProM can be easily extended by custom analysis tools, tailored towards the needs of a specific experiment or domain. This way, we can quickly add to the analysis capabilities while leveraging the existing infrastructure (e.g., to efficiently load and process large logs including semantic annotations).

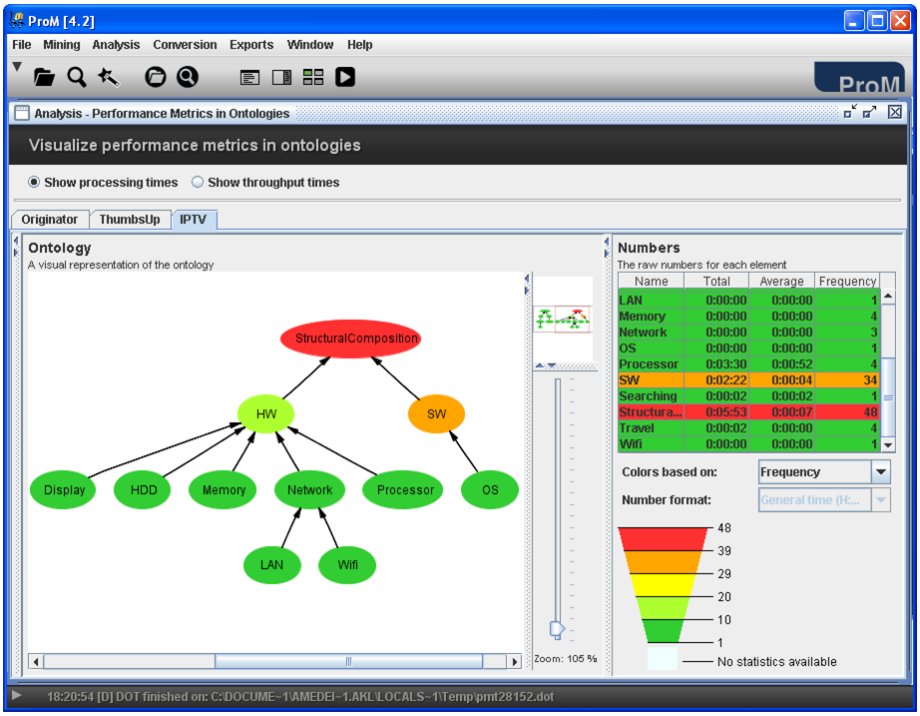

Fig. 7. Screenshot of the *Performance Metrics in Ontologies* semantic ProM plug-in. The current view shows the frequencies of tasks linking to concepts.

6 Case Study

The application of semantic observation and analysis is currently being used in industrial case-studies carried out with a large Dutch consumer electronics manufacturer. The setting is a new product with a couple of innovative features being under development. However, there is no reliable prior experience to assess whether the product is suitable for the target market. Furthermore, information about customer preferences and expectations is not available. Therefore, an unusually long and extensive conceptualization phase within the product creation process was decided upon and this will be the general study setting. To approach the problems outlined above, working prototypes of the product, called demonstrators, are equipped with observation facilities and are given to key testers. These demonstrator machines, once in use, deliver information about user interaction to a central repository (cf. Section 4) that is accessible for analysis purposes.

In the following we first describe the general setup and current status of the case study (Section 6.1) and then give an outlook on one particular aspect that we plan to investigate in the context of this study, namely the combination of objective and subjective data (Section 6.2).

6.1 General Setup

In a first study, over 800.000 data items were collected from demonstrator machines world-wide. This showed the general applicability of our approach to distribute testing machines in several countries and to remotely observe interaction with the systems. The results helped to answer questions of domain experts about the usage of the demonstrators. However, this study revealed huge difficulties to analyze this mass of atomic and non-connected data. Learning from this prior study, a second study has been initiated to explore the application of semantic meta-data within all phases of the flow. The preparations and setup of this study, which is currently in progress, can be briefly described as follows.

A new high-level link (referring to semantic meta-information) between data collection and analysis has been introduced. To establish this link, the first step was to develop the necessary ontologies. Together with domain experts inside the company we defined a large *product-specific* ontology which is used in the observation specification (partly displayed in Figure 5). An additional *experiment-specific* ontology is refined iteratively: in the beginning, only few basic categories were defined by estimating important parts in user-system interaction. After the first days of observation testing, the collected data items were analyzed and the outcome of the analysis indicated a lot of activity in one of the basic categories. Subsequently, this category could be refined to a set of about 30 subcategories. So, according to these first results, a more fine-grained view on the product usage was enabled from then on. The generic *observation ontology* (cf. Figure 3) is used as a support to distinguish events and status information coming from different origins on a basic level. Demonstrator machines provided already first semantically structured information which could be automatically processed by the system presented in this paper. Extended analysis of the collected data will be performed as soon as this second case study is completed (i.e., as soon as the experiment has finished). We expect that, with the help of semantic meta-information, the resulting data become much more comprehensible and lead to a better understanding of the product usage.

6.2 Opportunities in Combining Objective and Subjective Data

A very interesting aspect that we plan to investigate in the future is the combination of objective and subjective field data. In this paper, we focused on objective data, i.e., *usage data* directly recorded by the product (key presses, browsing behavior etc.). However, first attempts have been made to display *surveys* on the screens of the observed product according to the reported usage. Likewise, the observation system offers a possibility for *user-initiated* feedback. Data collected by these subjective measures are stored in the same data base as the objective usage data, and they are also semantically annotated (cf. the `SubjectiveUserEvent` concept in Figure 3). For instance, a question about the usage of the wireless keyboard would be connected to the corresponding ontology concept “wireless keyboard”.

Given the availability of both objective and subjective data, a number of new opportunities emerge both in terms of data analysis and data collection. For example, the context of the provided feedback can be extracted from the objective usage data (“What

was the user doing before? "). Furthermore, subjective measures can be triggered automatically based on the occurrence of certain usage patterns to gain more insight into motives ("Why was the user behaving that way?"). Here, the necessity of structuring the logged data by semantic concepts becomes even more apparent as data from very different perspectives is collected and related to the same usage process. Thus, the semantic annotations provide a means to abstract and focus on the data of interest.

7 Conclusion

In this paper, we presented a novel approach to semantically link the observation and analysis of product usage data by conceptual information captured in ontologies. This link yields two main benefits: (i) it adds focus to the potential mass of captured data items; and (ii) it reduces the need for extensive post-processing of the collected data. Together, these two benefits speed up the information feedback quality and cycle towards development. Furthermore, we presented a tool chain that supports our approach throughout the phases of observation specification, data collection, processing and analysis, and outlined a possible application scenario based on an industrial case study that is currently performed. As a next step, we will evaluate the data collected in this case study. Furthermore, we would like to apply our approach in different contexts to see whether it is sufficiently generic.

We presented our vision of a fully automated data collection, processing, analysis and presentation chain which is specified by only a few (potentially re-usable) documents. Ontologies and visual languages seem to be good candidates for such specification documents as they are accessible to the actual stakeholders of the observed usage data (e.g., the various domain experts). By putting these people in the position of being able to *specify what they want to observe*, one of the main problems in log analysis, namely data quality, can be addressed. In many real-life scenarios, the data are often still of a poor quality; because of a low priority in implementing logging facilities, and a lack of anticipation of the kind of analysis that should be eventually performed, collected data are not good enough to answer all the questions of interest. However, because of the immense opportunities and increasing feasibility (due to automated approaches like presented in this paper) it can be expected, that the integration of observation functionality will have a more prominent role in future product developments, which we have termed *design for observation*. As a consequence, better analysis results can be expected.

Acknowledgments

This work is being carried out as part of the *Managing Soft-Reliability in Strongly Innovative Product Creation Processes* project, sponsored by the Dutch Ministry of Economic Affairs under the IOP-IPCR program. Some of the authors are also supported by the European project SUPER [29]. Furthermore, the authors would like to thank Christian Günther for his help in the pre-experiment analysis, and the development team for the kind support and the possibility of applying our approach in a real product development context.

References

1. Brombacher, A., Sander, P., Sonnemans, P., Rouvroye, J.: Managing product reliability in business processes 'under pressure'. Reliability Engineering & System Safety **88** (2005) 137–146
2. den Bouwmeester, K., Bosma, E.: Phases of use: a means to identify factors that influence product utilization. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM Press (2006) 117–122
3. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**(2) (1993) 199–220
4. Hartson, H., Castillo, J.: Remote evaluation for post-deployment usability improvement. Proceedings of the working conference on Advanced visual interfaces (1998) 22–29
5. Hilbert, D.M., Redmiles, D.F.: An approach to large-scale collection of application usage data over the internet. icse **00** (1998) 136
6. Kabitzsch, K., Vasyutynskyy, V.: Architecture and data model for monitoring of distributed automation systems. In: 1st IFAC Symposium on Telematics Applications In Automation and Robotics, Helsinki (2004)
7. Kort, J., de Poot, H.: Usage analysis: combining logging and qualitative methods. In: CHI '05: CHI '05 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM Press (2005) 2121–2122
8. Shifroni, E., Shanon, B.: Interactive user modeling: An integrative explicit-implicit approach. User Modeling and User-Adapted Interaction **2**(4) (December 1992) 331–365
9. Estublier, J., Vega, G.: Reuse and variability in large software applications. In: ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, New York, NY, USA, ACM (2005) 316–325
10. Myers, B.A.: Visual programming, programming by example, and program visualization: A taxonomy. In Glinert, E.P., ed.: Visual Programming Environments: Paradigms and Systems. IEEE Computer Society Press, Los Alamitos (1990) 33–40
11. Evans, E.: Domain Driven Design. Addison-Wesley (2004)
12. Casati, F., Shan, M.: Semantic Analysis of Business Process Executions. In: 8th International Conference on Extending Database Technology (EDBT '02), London, UK, Springer-Verlag (2002) 287–296
13. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business Process Intelligence. Computers in Industry **53**(3) (2004) 321–343
14. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: a Vision Towards Using Semantic Web services for Business Process Management. In: IEEE International Conference on e-Business Engineering (ICEBE 2005). (2005) 535 – 540
15. O'Riain, S., Spyns, P.: Enhancing the Business Analysis Function with Semantics. In Meersman, R., Tari, Z., eds.: OTM Conferences (1). Volume 4275 of Lecture Notes in Computer Science., Springer (2006) 818–835
16. Sell, D., Cabral, L., Motta, E., Domingue, J., Pacheco, R.: Adding Semantics to Business Intelligence. In: DEXA Workshops, IEEE Computer Society (2005) 543–547
17. Alves de Medeiros, A., Pedrinaci, C., van der Aalst, W., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L.: An Outlook on Semantic Business Process Mining and Monitoring. In Meersman, R., Tari, Z., Herrero, P., eds.: OTM Workshops (2). Volume 4806 of Lecture Notes in Computer Science., Springer (2007) 1244–1255
18. van der Aalst, W., Reijers, H., Weijters, A., van Dongen, B., Alves de Medeiros, A., Song, M., Verbeek, H.: Business Process Mining: An Industrial Application. Information Systems **32**(5) (2007) 713–732

19. de Medeiros, A.A., van der Aalst, W., Pedrinaci, C.: Semantic Process Mining Tools: Core Building Blocks. In: Proceedings of the 16th European Conference on Information Systems (ECIS). (2008)
20. Guenther, C., van der Aalst, W.: A Generic Import Framework for Process Event Logs. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103. (2006) 81–92
21. van der Aalst, W., van Dongen, B., Günther, C., Mans, R., Alves de Medeiros, A., Rozinat, A., Rubin, V., Song, M., Verbeek, H., Weijters, A.: ProM 4.0: Comprehensive Support for Real Process Analysis. In Kleijn, J., Yakovlev, A., eds.: Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007). Volume 4546 of LNCS., Springer-Verlag, Berlin (2007) 484–494
22. : W3C: Web Ontology Language (OWL). http://www.w3.org/2004/OWL/
23. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language wsml: An overview. In Sure, Y., Domingue, J., eds.: ESWC. Volume 4011 of Lecture Notes in Computer Science., Springer (2006) 590–604
24. : WSMT: Web Service Modelling Toolkit. http://sourceforge.net/projects/wsmt
25. Funk, M., van der Putten, P.H.A., Corporaal, H.: Specification for user modeling with self-observing systems. In: Proceedings of the First International Conference on Advances in Computer-Human Interaction. (2008)
26. Funk, M., van der Putten, P.H.A., Corporaal, H.: Model interpretation for executable observation specifications. In: Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute (2008) to be published
27. Funk, M., van der Putten, P.H.A., Corporaal, H.: UML profile for modeling product observation. In: Forum on Specification and Design Languages (FDL'08), IEEE Computer Society (2008) to be published
28. Alves de Medeiros, A., van der Aalst, W., van den Brand, P., Weijters, A.: D6.5 - Semantic Process Mining Prototype. SUPER European Project Deliverable, http://www.ip-super.org/res/Deliverables/M18/D6.5.pdf (2007)
29. European Project SUPER: Semantics Utilised for Process Management withing and between Enterprises. http://www.ip-super.org/