Shaken not Stirred: Mixing Semantics into XPDL

Philip Webster & Victoria Uren
University of Sheffield
Department of Computer Science
Regent court
211 Portobello
Sheffield S1 4DP
{P.Webster,V.Uren}@dcs.shef.ac.uk

Marcus Ständer
Technische Universität Darmstadt
Fachgebiet Telekooperation
Hochschulstr. 10
64289 Darmstadt, Germany
staender@tk.informatik.tu-darmstadt.de

ABSTRACT
Ubiquitous computing requires lightweight approaches to coordinating tasks distributed across smart devices. We are currently developing a semantic workflow modelling approach that blends the proven robustness of XPDL with semantics to support proactive behaviour. We illustrate the potential of the model through an example based on mixing a dry martini.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Constructs and Features – *datatypes and structures*, I.0 [**Computing Methodologies**]: General, J.7 [**Computers in Other Systems**]: Consumer Products,

General Terms
Languages

Keywords
XPDL, Ubiquitous computing, Semantic workflows.

SBPM2010, May 30–31, 2010, Heraklion, Crete, Greece.

1. INTRODUCTION
Imagine a stylish apartment in the not so distant future. Bob, a young IT consultant, has invited his boss to dinner. To impress her, he plans to serve dry martinis: four parts gin, one part dry vermouth and a green olive, chilled to the dew point and shaken or stirred according to her preference. Luckily for Bob's career prospects he has invested in several "smart" consumer products that will assist him in creating the perfect cocktail, rather than a shot of lukewarm gin with a medicinal aftertaste.

The SmartProducts project is investigating the technologies required to make scenarios such as this one a reality. The project envisages systems in which some smart products would incorporate sensors that can gather environmental data; in this paper, a cocktail shaker incorporating a temperature sensor is taken as an example. Some smart products would also have enough capacity to reason over ontologies or execute workflows; in this paper, this kind of product is illustrated by a device called the Cocktail Guide. Wireless communication would be used to exchange information between different products in ad hoc ubiquitous environments. Workflows would provide a means to model tasks that involve a sequence of activities, and to coordinate activities being carried out by several products in cooperation with human users.

It has been said that modelling behaviour as workflows causes "users to lose control over their work and work to lose the benefit of the insights which users bring" [Dourish 1996]. This is a risk for commercialisation of smart products because, unlike a business environment where employees can be compelled to comply with workflow related practises, whether they like them or not, buying a smart cocktail shaker is a voluntary act. Therefore, to enhance the experience of using these products, we propose to add semantic descriptions to workflows to allow ubiquitous systems to copy the users' capability of creating links to objects, combining and reasoning. Ultimately the aim would be to deliver proactive behaviour based on context information from the environment, for example, recommending workflows to the user, or identifying smart products in the vicinity, which could carry out a given workflow activity.

Annotated workflows have already been investigated in the fields of business process management, semantic web services and grids; a review is provided by [Lautenbacher 2007]. All these fields are characterised by something that the ubiquitous

computing environment we have described notably lacks, which is access to industrial strength computer processing power. By contrast SmartProducts' technology needs to be deployed on consumer products. This means that cost is a major factor in the selection of electronic. Even a component costing one euro would significantly increase the final cost of some smaller smart products. These commercial considerations mean that, for the purposes of current research, we are aiming at working with gumstix (http://www.gumstix.com). Gumstix is an open source specification for a computer on a circuit board about the size of a stick of chewing gum. The targeted gumstix have a 600 MHz processor and 256 MB of SDRAM [SmartProducts D6.2.1], Although the actual electronics used when products go into production would most probably not be gumstix, they provide a readily available research platform of about the right size and complexity.

Running a workflow execution engine on a gumstix platform is a big leap from current standards, and in this example we assume that the workflow execution will be controlled by the PDA device which hosts the Cocktail Guide. Our first step towards achieving semantic workflows executing on small devices has been to develop a lighter weight modelling approach than those that currently exist. This paper presents the work we have done so far.

Section 2 discusses related work, especially work on semantic annotation of workflows. Section 3 presents the details of our proposed approach, with discussion of some of the design choices that were made. Section 4 illustrates functionality the annotated workflows should support using the running example. In Section 5 we present conclusions.

2. Related Work

Modeling task related behavior has been studied extensively because of its obvious commercial importance. Methods that have been considered include task models such as GOMS [Card 1983] and CTT [Paterno 1997], graph models including Petri Nets [Salimifard 2001], and process definition languages like BPEL [IBM 2007] and XPDL [WfMC 2008]. Since especially the latter ones have been standards for some time now, they are often used as bases for manual, semi-automated or fully automated systems. In the literature there are basically two different concepts for such process models: *business processes* and *workflows*. The term "business process" thereby describes processes that are focused on high level descriptions, where objectives play an important role, while the term "workflow" slides more into the direction of grid computing, which is much more close to technical details of the environment. Due to the kind of processes in smart environments, we stick to the term workflow.

There are many related projects, which are using workflow technology combined with semantic information. DEMAC [Kunze 2006] for example uses DPDL, an extension of XPDL. DPDL allows annotations about required devices to be attached to the different participants of a workflow and thus allows the system to choose appropriate devices during runtime. A possible problem with such an approach can appear if something changes, e.g. a new device with previously unknown capabilities is deployed in the environment. Suddenly the annotation could be unsuitable, since it no longer describes the best suitable product.

The SmartProducts approach is similar in some respects to the approach taken by Kunze, Zaplata and Lamersdorf [Kunze 2006] in that cooperation between devices is emphasised over orchestration of participants or services. It is expected that control of process execution will be transferred between the participating devices, rather than being managed by a centralised workflow execution engine. The SmartProducts consortium agreed with the statement by [Kunze 2006] that a centralised engine may become a "single point of failure" and potentially become a "bottleneck during execution time". All Smart Products developed for the project communicate over local wireless networks, and may have low communications bandwidth. SmartProducts differs from DEMAC in that the process definitions are not transferred from device to device as execution progresses. Consequently, the transactional and error handling additions made to XPDL in DEMAC are not required in our case.

SUPER (http://www.ip.super.org) is an integrated project providing tools to support the creation and execution of semantically-enhanced workflows. The SUPER project also uses annotations to provide additional information in workflows. They use links to ontologies, goals, web services and more [SUPER 2009] to allow semantic workflow composition, relate input/output to their ontology and allow inclusion of web services.

The project provides an Eclipse-derived editor based on the WSMO Studio editor (http://www.wsmostudio.org/) called BPMO Editor [Dimitrov 2007] which allows a user to create workflow process definitions using BPMN (Business Process Modelling Notation) or EPC (Event-driven process chain) or to load existing process definitions in these formats and to then add semantic annotation to components of the model, based on individuals from an OWL (Web Ontology Language) ontology definition. Process definitions created using the BPMO editor are then converted to BPEL by a plugin for execution on a workflow execution engine. However, this approach is closely connected to the software composition of different services and not to the distribution of workflows on different products, having limited resources, e.g. being able to execute one workflow at a time only.

The approach to process definition, annotation and execution in SmartProducts differs from the work presented for SUPER in a number of areas. The foremost difference is the human-centric approach that is central to the SmartProducts platform - products are intended to assist a human user to complete a task rather than being a set of services to be orchestrated. BPEL has weaker support for human participants - this was added initially as a vendor extension (BPEL4People), whereas BPMN and the XPDL serialisation format have human participant capabilities included as standard. This difference in scope between BPEL and BPMN affected the choice of execution engine, and had knock-on effects on the choice of process definition languages and tools that could be used without incurring complexity and performance issues.

Beyond the human participant emphasis, technical reasons also influenced the selection of a non-BPEL engine. SmartProducts uses a workflow execution engine that can execute BPMN (serialised as XPDL) directly, thus eliminating the need for a BPMN-to-BPEL translation layer. Much research into BPMN to BPEL translation has been done, with emphasis on various techniques for preserving the characteristics of a BPMN process diagram when converted to a BPEL executable model. Depending on the modeling style used when creating a BPMN diagram, the resulting BPEL produced by a conversion algorithm may have increased complexity (Recker, J. C. M., 2006) and associated loss of human comprehensibility, due to the conceptual mismatch between the two languages. The solution preferred by

SmartProducts was to use XPDL as the serialisation format for BPMN process diagrams, and also as the execution format processed by the workflow engine. This eliminated the complexity issues that would be faced if a conversion were required between the definition and execution phases.

In addition, the BPEL approach places heavy emphasis on the use of Web Services to perform execution of the individual blocks of activity in a diagram, and relies on XML-based formats for service invocation and data transfer. By comparison, the BPMN/XPDL approach allows code to be associated directly with the activities represented in a process definition. In the XPDL model, implementation of remote calls is left to the developers. This was an advantage for SmartProducts, as the platform is intended to run on a distributed set of resource contrsained devices, with no central 'master' process co-ordinator. Typical SmartProducts devices may be smart kettles or smart ovens, and as such will have relatively slow CPUs and small amounts of memory, making efficient methods for data transfer and processing very important. Use of a full Web Services framework would massively restrict the functionality of the SmartProducts devices, as the WSDL and SOAP processing overhead would be much greater than the overhead imposed by the alternative lightweight embedded middleware used in SmartProducts. Each device can execute relevant portions of a process directly in a small and functional embedded workflow execution engine, with inter-participant communication achieved via the use of a communication middleware layer (MundoCore) which is also embedded on the devices.

Thus, while the SUPER project's implementation was guided by the requirements imposed by the aim to satisfy the needs of large enterprises seeking to control and monitor commercial business processes on centralised workflow execution servers, the SmartProducts implementation is aimed at a radically different environment: clusters of small devices working together with a human user to achieve goals specific to the human user participating in the process - with the added flexibility that the human can influence the execution path and non-human participant binding dynamically during process execution, rather than at process definition time.

Further approaches like NEXUS (http://www.nexus.uni-stuttgart.de) or ASTRO (http://astroproject.org) also provide the ability to use semantic information. Unfortunately central workflow management or missing possibilities to describe the elements of a workflow flexibly enough make them not completely suitable for our SmartProducts setting.

3. SEMANTIC WORKFLOW MODEL

In this section we detail the semantic workflow model and discuss our design decisions.

3.1 The Process Definition Language

One of the central issues in creating smart environments is the modeling and handling of processes. These processes require a semantically well defined language allowing developers to define for example the organizational structures of the different steps, the participants or how to use automation capabilities of some smart product. Thus, process descriptions range from a very high level view down to very system specific details. Regarding the established standards for process modeling, like BPEL [SRC], JPDL[SRC] or XPDL [WfMC 2008] it has shown that XPDL is a suitable base for such descriptions [Kunze 2007]. The development is eased since XPDL supports many workflow patterns, which are often used while modeling, directly [van der Aalst 2003]. Further, since XPDL is based on the business process modeling notation BPMN [OMG 2009], the workflows can be visualized using standardized human readable graph visualizations. There are several open source editors available (e.g. JaWE), which help developing the workflows. During runtime, this visualization eases tracking of the current state of a workflow. Using the XPDL standards also allows developers to embed their own code into the workflows and thus automate processes by directly steering certain hardware or call own software from within the workflow. To further extend the power of the language, the WfMC has designed XPDL to be extensible. There are tags like the ExtendedAttributes, which can be used to append data to different parts of the workflows.

Consequently, our proposed model is an extension of XPDL using semantic annotation to link in the ontologies. This approach aims to blend the power of semantics with the proven capabilities of XPDL.

3.2 Role of Rules

In a first step, the workflows get annotated with information concerning when to start that workflow. Usually there are two possible ways to automatically start a workflow: (1) having a big workflow that permanently runs and that covers every possible situation and then starts sub workflows or (2) trigger workflows from outside. Basically both opportunities are based on the definition of a set of rules. In our approach we directly attach these rules to the headers of the workflows. The purpose of attaching annotations and trigger rules to the workflow itself is data transfer. It packages the semantic, non-semantic and workflow information together so that they can be conveniently stored, e.g., on a new smart product when it is shipped or on a website that provides new workflows to smart product owners. The workflows can be treated as simple XPDL and, if more powerful computing facilities are available, semantic annotations and rules can be transferred to them for reasoning.

At the time of writing, the rule language that will be used for the SmartProducts infrastructure remains undecided: the web standard SWRL (http://www.w3.org/Submission/SWRL/) and Jess (http://www.jessrules.com/) are candidates. To minimize computational requirements, it is likely that a forward chaining rule engine will be used.

The format of rules will be determined by the reasoner selection but we can assume for the purpose of discussion that they may take a form such as:

IF (conditions) THEN (parameters)

Where the conditions are Boolean expressions containing context values from the SmartProducts environment and the parameters are optional context values that identify additional data that the workflow may require during its execution.

Workflow identifiers are not explicitly included in the definition of rules given above. This is deliberate – the SmartProducts platform will be used in environments where devices from multiple product vendors co-operate to complete a process. Each vendor may choose to include a number of process definitions with their product, and vendors may not conform to a restrictive set of rules for defining workflow identifiers. For this reason, the

SmartProducts runtime will generate the identifiers used by the process execution engine, eliminating the possibility of conflicts by ensuring uniqueness. Vendors may still include their own workflow identifiers, but these will not be the ones used during execution.

All trigger rules will be removed from the process definitions and stored elsewhere, in the trigger component, while the workflows themselves will be located in a process repository. Workflow identifiers will thus be made unique within a SmartProducts environment without the need to impose restrictions on vendors.

For the initial design, we reuse the concept of formal parameters used in the XPDL standard. When the workflow is started, so-called actual parameters are mapped to the formal parameters, allowing the execution engine to pass required information, like the user that issued a workflow. Since these parameter signatures form the "interfaces" of the workflows, they can later be described by standards like the Web Service Description Language (WSDL) to allow workflows to be used more dynamically.

We have identified two situations in which rules are required. The first is for recommending workflows. This will happen outside of the workflow engine. Therefore rules of this type will be added to workflows as annotations so that they can be extracted and reasoned with to determine the proactive behaviour in the ubiquitous environment. The second situation concerns rules that are required in order to control the flow of an executing workflow. XPDL already has facilities to add rules of this kind to transitions. We propose to reuse this feature rather than add semantic rules to transitions.

The workflow triggering rules should support the proactive recommendation of workflows to the user. Default rules for the triggering of workflows may be attached to workflows when they are originally created and shipped to the user. A default rule for the dry martini could be that it should not be recommended before 6pm or to anyone under the age of 18. Individual users may wish to supplement these default rules with personal preferences, for example permitting the suggestion of cocktails from midday onwards at weekends and on holidays or, for users who abstain from alcohol, rules that ensure they are only offered nonalcoholic cocktails.

3.3 Annotation

Annotation provides the opportunity to add both semantic and nonsemantic metadata. This could be of the following kinds:

- URIs (linking to instances of an ontology) (semantic)
- Locally defined tags, required because an ontology will never be complete (could become semantic if fused with the ontology)
- Snippets of text that could be presented to the end user during workflow execution (not semantic)
- Links to other resources such as images (not semantic)
- Rules (semantic)

The annotation approach that we propose uses existing XPDL conventions. In principle we make use of two different kinds of information when processing a workflow. Informational metadata such as required capabilities of a product, semantic information about the activity, etc. and information that directly belongs to the information flow of the workflow, like text, names, or links to images. While the metadata is put into extended attributes, internal data is stored in workflow variables.

Adding annotations

To allow such annotations, WfMC's XPDL standard defines the *ExtendedAttribute* element as follows:

"6.4.14.1. Extended Elements and Attributes

The primary method to support such extensions is by the use of extended attributes. Extended attributes are those defined by the user or vendor, where necessary, to express any additional entity characteristics. XPDL schema supports namespace-qualified extensions to all the XPDL elements. The XPDL elements can be extended by adding new child elements, and new attributes." [WfMC 2008]

Other options within XPDL, such as *ExternalReference*, have too restricted scopes of use. Extended attributes are a flexible approach that we have used to add semantic annotations by developing a "vocabulary" of different types of extended attributes. They can be added to any kind of XPDL element. Therefore annotations could be added in the workflow header if they apply to the whole workflow or be attached to specific XPDL elements such as activities, applications, participants, performers or transitions if a more precisely scoped annotation is required. However, so far, we are only annotating activities, applications and participants. Performers and transitions should not have to be modeled explicitly for reasons explained below.

The WfMC guidelines imply that it is also possible to define *ExtendedElements*, for example an Application element could be defined for smart product software. However in practice we cannot find evidence of this being done. Therefore we propose to define a vocabulary of extended attributes and guidelines for their use specifying which kinds of elements they can be attached to.

In SmartProducts, lists are used in ExtendedAttributes. This permits multiple values to be added to one workflow element with a given semantic annotation. In practice, this can be used to provide a list of expected tools or devices that could be used by a participant to complete an activity, or it could refer to a list of ingredients to be used for a given step in a recipe.

This approach realizes a SmartProducts metadata element set, which is more controlled than totally free annotations in which any extended attributes could be attached to any element. As a starting point we propose three named extended attributes, PRODUCT_CLASS, METADATA and ACTIVTY, which are defined in table 1. The workflow elements these can be attached to are outlined below.

Participant – participants that are smart products can be annotated with:

- PRODUCT_CLASS
- METADATA

The Participant PRODUCT_CLASS annotation should not be used to provide an exhaustive list of acceptable devices that may

be used to perform an activity – rather, it enumerates the preferred devices as envisaged by the process designer. The flexibility of the SmartProducts platform permits users to utilize alternative devices not explicitly defined within the process definition. It may also be possible for a user to use a non-smart device to perform an activity, and if the workflow engine was designed to strictly reject any non-specified devices, the process would be stalled indefinitely in these cases.

Activity – activities that are performed by smart products can be annotated with:

- PRODUCT_CLASS
- ACTIVITY
- METADATA

Application – applications describing smart products software can be annotated with:

- PRODUCT_CLASS
- ACTIVITY
- METADATA

The XPDL Performer element is not annotated, as this serves only as a link between a Participant and an Actvity (both of which are already capable of accepting SmartProducts annotations). The XPDL Transition element is also not annotated, as the aim of SmartProducts is centred on the performance of activities rather than the flow of the process. The use of expressions as conditions in transitions is considered sufficiently flexible for the project.

It makes sense to have a place in a workflow where people can store trigger rules that they write to define context conditions that would trigger that flow. These could then be extracted from the workflow and added to the Ubiquitous Data Store. We propose the use of ExtendedAttribute to embed a trigger rule into the workflow. In this case the format of the rule is irrelevant from the viewpoint of the workflow and can be handled as a text string.

Table 1. Definitions of SmartProducts Extended Attributes

Annotation name	Description
PRODUCT_CLASS	A product type or types as defined in an ontology. Based on the product class it should be possible to identify substitute products with similar functionality.
METADATA	An instance or instances from a related domain ontology, excluding products and activities. It supports domain specific annotation.
ACTIVITY	A type of action that is required to complete an activity. This supports the identification of products based on their capabilities rather than their type.

4. WORKED EXAMPLE

An example of a typical SmartProducts scenario is presented to illustrate the use of the semantic annotations added to XPDL, and how the SmartProducts platform makes use of this additional information to allow new functionality to be implemented. These examples refer to the workflow which is illustrated in figure 1.

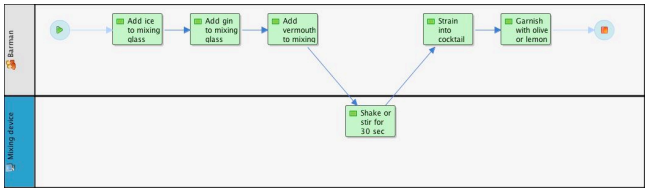

Figure 1: A workflow for creating a Dry Martini

4.1 Selection of Workflow

Imagine Bob can remember that the very stylish cocktail favoured by James Bond, includes gin, but he can't remember what it is called. He turns to his Cocktail Guide, a piece of software that is installed on his PDA, and which can handle the execution of cocktail-making workflows. A search for gin quickly pulls up a list of cocktails, which include the ingredient gin. This is possible because the XPDL Activity elements have been annotated with METADATA ExtendedAttributes that draws on a domain ontology that describes the typical ingredients of cocktails.

The following example shows the SmartProducts annotations added to the 'add gin to mixing glass' step in the creation of a dry Martini.

<Activity Id="add_gin" Name="Add gin to mixing glass">

<Performer>barman</Performer>

<ExtendedAttributes>

<ExtendedAttribute Name=" http://www.smartproducts-project.eu/ontologies/cocktails.owl #METADATA" Value="Gin"/>

<ExtendedAttribute Name=" http://www.smartproducts-project.eu/ontologies/cocktails.owl #ACTIVITY" Value="Pouring"/>

<ExtendedAttribute Name=" http://www.smartproducts-project.eu/ontologies/cocktails.owl #PRODUCT_CLASS" Value="MixingGlass"/>

</ExtendedAttributes>

</Activity>

4.2 Selection of Devices

The smart devices in Bob's house are wireless enabled. Therefore the Cocktail Guide can recognize which devices are available in the apartment. Each device can broadcast a semantic description, which specifies the kinds of action it can perform and the kinds of contextual information it can provide. The workflows are similarly annotated with semantic metadata which describe the actions required to complete activities and context information required to coordinate the process.

The constrained hardware of ubiquitous computing environments will compel us to keep the reasoning we do lightweight and to prove that "a little semantics goes a long way" [Hendler 2003].

The kinds of lightweight reasoning tasks that will be needed are detailed below.

Semantic Querying: Semantic reasoning is required to match the needs expressed in the workflow annotations against the capabilities of the devices in the environment. In our running example, Bob's apartment contains two alternative devices for mixing the cocktail: the SmartSpoon and the SmartShaker both of which can perform mixing and detect temperature.

Mapping: Just as nobody can be compelled to buy a smart cocktail shaker, nobody can be compelled to buy all their appliances from the same manufacturer (see the discussion on vendor specific rules in section 3.2). Consequently, different appliances with similar capabilities will be described differently. Taking a semantic approach therefore has clear advantages. Ad hoc mapping techniques can be envisaged which could recognize that "Blending" in one ontology is similar to "Mixing" in another.

The example below attaches a metadata URI to a Participant element. The scope of this annotation is restricted to the participant. This example comes from the header of a workflow about making a dry Martini: the annotation identifies that a mixer can be used and that there are two smart activities that may be carried out by a compatible mixer. Here there is a choice between a shaken or stirred Martini.

<xpdl:ExtendedAttributes>

<xpdl:ExtendedAttribute
Name="JaWE_GRAPH_PARTICIPANT_ID"
Value="Mixer"/>

<xpdl:ExtendedAttribute

Name="PRODUCT_CLASS"

Value="http://www.smartproducts-project.eu/ontologies/cocktails.owl#Shaker, http://www.smartproducts-project.eu/ontologies/cocktails.owl#Stirrer"/>

<xpdl:ExtendedAttribute

Name="ACTIVITY"

Value="http://www.smartproducts-project.eu/ontologies/cocktails.owl#Stirring, http://www.smartproducts-project.eu/ontologies/cocktails.owl cocktails.owl#shaking"/>

</xpdl:ExtendedAttributes>

4.3 Guiding the user

Bob's Cocktail Guide, can guide him through the dry martini recipe step by step. This will be achieved by using sections of text and images embedded in the workflow, which can be relayed to Bob through his preferred communication screen; in this case the T.V. set in his living room.

The workflow model requires a way in which to store sections of the original text or diagrams. It is common practice in XPDL to store such text in variables using the XPDL element DataFields. This therefore is the element we propose to use. The example below illustrates DataField syntax.

<xpdl:DataFields>

<xpdl:DataField Id="recipe_1" IsArray="FALSE" Name="dry Martini recipe">

<xpdl:DataType>

<xpdl:BasicType Type="STRING"/>

</xpdl:DataType>

<xpdl:Description>Take four parts gin, one part dry vermouth and place in a cocktail shaker with ice </xpdl:Description>

</xpdl:DataField>

</xpdl:DataFields>

4.4 Incorporating Context

To be drinkable the dry martini must be sufficiently chilled. The workflow is written in such a way that it will not proceed to the serving step until it has confirmation that the drink has reached the right temperature. Bob's smart cocktail shaker contains a temperature sensor, and the workflow execution engine can make use of this information source.

In general, sensor information can be seen as part of the context and designers of workflows should only need to specify when they react on context changes. In the example above they should only need to define that "*the drink should have the right temperature*". The detailed information regarding what the 'right temperature' means and where this information can come from, should be described in the ontology. The temperature could be measured by different temperature sensors included in or attached to a product, or the user could press a button that acknowledges the action.

However, concerning sensor data there is still an open issue. Thinking of pure OWL annotations it is not clear how best to define ranges of sensor data. Describing that the temperature is equal to 10° Celsius seems reasonable, since this is a single semantic element. Defining variable ranges like 10-20° Celsius or 10-21° Celsius does not fit well into the current schema of annotating with simple semantic elements. Thus, more complex management information may need to be added in the future.

5. Conclusions

A review of the approaches to semantic annotation of process definitions taken so far in existing research led us to conclude that the SmartProducts platform would need to develop a new approach. This approach makes use of the capabilities of a standard process definition serialisation format (XPDL) that also has good support as a language that can be executed by process engines. Semantic annotation functionality was added to improve retrieval of appropriate workflows and to support functionality such as identifying products that can compete a given activity.

This paper presents the initial effort that has been made toward this goal. A working version of the system, which includes a newly-defined set of annotations that can be applied to XPDL has been presented. In additional work not presented here, an editor prototype that provides support linking of individuals from ontologies to process definition components, and a workflow engine that can execute semantically-enhanced process definitions are being developed. Future work will enhance the expressiveness and flexibility of the annotation system while also retaining the ability to use the annotated workflows on lightweight devices co-operating with a human user.

6. REFERENCES

[Bowman 1993] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. ACM Trans. Program. Lang. Syst. 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.161471.

[Brown 2003] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE. In Proceedings of the 16th Annual ACM Symposium on User interface Software and Technology (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM Press, New York, NY, 1-10. DOI= http://doi.acm.org/10.1145/964696.964697

[Card 1983] Card S, Moran T, Newell A, 1983. The psychology of human-computer interaction. Lawrence Erlbaum Associates Inc., 1983

[Dimitrov 2007] Dimitrov, M., Simov, A. Stein, S., Konstantinov, M. 2007 A BPMO Based Semantic Business Process Modelling Environment, Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM-2007), Vol-251, CEUR-WS, June 2007, ISSN 1613-0073

[Ding 1997] Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[Dourish 1996] Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., & Zbyslaw, A. (1996). Freeflow: mediating between representation and action in workflow systems. In Proceedings of the 1996 ACM conference on Computer supported cooperative work (p. 198). ACM. Retrieved from http://portal.acm.org/citation.cfm?id=240252.

[Forman 2003] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.

[Fröhlich 2000] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM Press, New York, NY, 526-531. DOI= http://doi.acm.org/10.1145/332040.332491

[Hendler 2003] On Beyond Ontology, Keynote talk, International Semantic Web Conference 2003.

[IBM 2007] IBM, B. Systems, Microsoft, SAP, and Siebel. Web services business process execution language (WS-BPEL, BPEL), version 2.0 specification, 2007

[Kunze 2007] Kunze, C., Zaplata, S., & Lamersdorf, W. (2007). Mobile processes: Enhancing cooperation in distributed mobile environments. *Journal of Computers*, *2*(1), 1–11.

[Lautenbacher 2007] Florian Lautenbacher, Bernhard Bauer, A Survey on Workflow Annotation & Composition Approaches, Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SemBPM) in the context of the European Semantic Web Conference (ESWC), pp. 12-23, 7th June 2007, Innsbruck, Austria

[Ouyang 2008] Ouyang C., Dumas M., van der Aalst W.M.P., et al. (2008). Pattern-based translation of BPMN process models to BPEL Web services. International Journal of Web Services Research. 5 (1), 42-62.

[Paterno 1997] Paternò F, Mancini C, Meniconi S. ConcurTaskTrees: A diagrammatic notation for specifying task models. In: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction table of contents, pp. 362 - 369 1997

[Recker 2006] Recker, J. C. M., Jan. (2006). On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages.

[Salimfard 2001] Salimifard K, Wright M. Petri net-based modelling of workflow systems: An overview. European journal of operational research. 2001;134(3):664–676.

[Sannella 1994] Sannella, M. J. 1994 Constraint Satisfaction and Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.

[SmartProducts D6.2.1] D6.2.1: Initial Architecture and Specification of Platform Core Services, SmartProducts, 2010

[Spector 1989] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. Acm Press Frontier Series. ACM Press, New York, NY, 19-33. DOI= http://doi.acm.org/10.1145/90417.90738

[Tavel 2007] Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[WfMC 2008] The Workflow Management Coalition Specification, Workflow Management Coalition, Workflow Standard Process Definition Interface-- XML Process Definition Language, Document Number WFMC-TC-1025, Document Status – Final Approved, October 10, 2008, Version 2.1a

[Yu 1989] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.