Enterprise Attention Management System

Darko Anicic[1], Nenad Stojanovic[1], and Dimitris Apostolou[2]

[1] FZI Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
~name.surname@fzi.de
[2] FDepartment of Informatics Decision Support Systems Lab, University of Piraeus, Greece
~dapost@unipi.gr

Abstract. We present a novel approach for proactive support of user in knowledge intensive organisations. Whilst once information was a scarce resource, nowadays all kinds and qualities of information are available. However human attention has become a scarce resource which is difficult to manage and support. Our attention management system proactively supports the user in dealing with processes, activities and tasks defined by a semantically-enhanced business workflow. Moreover the user is supported in reacting on changes respecting the users' context and preferences. The approach is based on an expressive attention model, which is realized by combining context-aware ECA rules with ontologies and an appropriate preference model. We present the system's architecture, describe its main components and present early evaluation results. Our system is particularly deployed in a use case related to eGoverment. Nevertheless the architecture we present is general, and may be used in all kind of information and knowledge systems where handling the user attention is of an important interest.

1 INTRODUCTION

Success factors in knowledge-intensive and highly dynamic business environments include the ability to rapidly adapt to complex and changing situations and the capacity to deal with large quantities of information of all sorts. For knowledge workers, these new conditions have translated in acceleration of working performance, multiplication of projects in which they are involved and increased collaboration with colleagues, clients and partners. Knowledge workers are overloaded with potentially useful and continuously changing information originating from a multitude of sources and tools. A significant part of a knowledge worker's day can be occupied with searching and looking for information.

In order to cope with changes of the business environment, the attention of knowledge workers must be always paid on the most relevant information sources. Indeed, a basic requirement of knowledge workers is to be up to date with information while facing an information overload situation. In other words, the issue is how to select those information resources whose reading will give most benefits to the reader. Moreover, agility and proactive computer can be

useful in an environment of knowledge workers with overburdened memories: The computer should know what a knowledge worker works with and show him/her relevant information before they need them, in a kind of pre-search function.

According to [1], attention is defined as a "focused mental engagement on a particular item of information". According to [2], attention is a process of selection and selective processing, required because the brain has a limited information processing capacity. In an enterprise context, attention management refers to supporting knowledge workers focus their cognitive ability on a particular organizational task and on the information resources required to accomplish it. In particular support is required for searching, finding, selecting and presenting the most relevant and up-to-date information without distracting workers from their activities. Information retrieval systems have provided means for delivering the right information at the right place and in right time. The main issue with existing systems is that they do not cope explicitly with the information overload, i.e. it might happen that a knowledge worker "overlooks" important information.

The goal of attention management systems (AMS) is to avoid information overload and to provide notifications about new and changed, relevant information [1]. Moreover, the frequently changing environment requires not only very effective systems for alerting knowledge workers that some relevant piece of information has appeared or has been changed, but also effective recommendation how to deal with these changes.

Our approach for managing attention in a business environment is not just to support receiving new relevant information proactively, but also to enable relevant reaction on this information (i.e. on a change in general). Such an action can be an already predefined workflow, but also ad-hock generated procedures according to the currently available knowledge/experience. In that sense, our approach of an enterprise Attention Management System (AMS) goes beyond informing a user proactively that something relevant has been changed, toward proactive preparing and supporting the user to react on that change. Our approach puts forward a comprehensive reasoning framework that can trigger a knowledge base in order to find out the best way to react on a change. We base such a framework on a combination of reactive, deductive, and preference rules and ontologies.

The paper is organized as follows: in the second section we analyze requirements and outline a framework for an enterprise attention management system, in the third section we present the SAKE attention management model, in the fourth section we present the architecture of the SAKE attention management system encompassing various new functionalities to address relevant attention-related issues. We conclude by suggesting outlets for future research and development in information technology for the purpose of managing users' attention in knowledge-intensive environments.

2 Requirements for AMS

This section summarises the basic requirements for an Enterprise Attention Management system:

1. Flexible modeling of information in order to enable focusing of attention on different abstraction levels. For example, a user interested in information about pets should be alerted for new information about domestic animals. Another issue is modeling the usage of information by a community of users in order to stimulate sharing of implicit knowledge. For example, users looking for front tyre pressure must be proactively fed with the rear wheels' pressure as most technicians are interested in both in most maintenance situations.
2. Context-awareness in order to support a particular decision making process. For example, new law about animals triggers different alerts in different regulation and business process areas.
3. Management of preferences for enabling efficient extraction of interesting information. In particular, there is a need for an expressive formalism for the description of preferences, including when to alert a user, but also how to react on an alert.

Figure 1 presents the general framework of Enterprise Attention Management (EAMS) that builds on the aforementioned requirements.

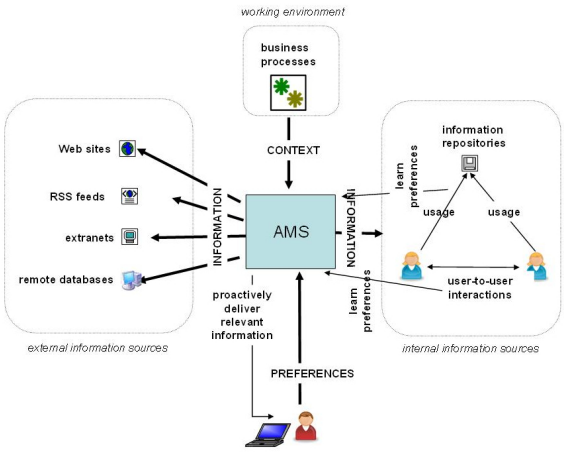


Fig. 1. Enterprise Attention Management Framework

Our framework is developed along 3 axes:

1. Information represents all relevant chunks of knowledge that can be found in the available information repositories and sources. In the business environment of an organization, sources of information can be both internal and

external to the organization. Moreover, information can be represented either formally (e.g. using information structuring languages such as XML) or informally. Finally, information may be stored in structured repositories such as databases that can be queried using formal languages or in unstructured repositories such as discussion forums.

2. Context defines the relevance of information for a knowledge worker. Detection of context is related to the detection of the user's attentional state which involves collecting information about users' current focus of attention, their current goals, and some relevant aspects of users' current environment. The mechanisms for detection of user attention that have been most often employed are based on the observation of sensory cues of users' current activity and of the environment; however others, non-sensory based, mechanisms also need to be employed to form a complete picture of the user's attentional state [3].
3. Preferences enable filtering of relevant information according to its importance/relevance to the given user's context. In other words, the changeability of resources is proactively broadcasted to the users who can be interested in them, in order to keep them up to date with new information. Users may have different preferences about both the means they want to be notified and also about the relevance about certain types of information in certain contexts. User preferences can be defined with formal rules or more informally by means e.g. of adding keywords to user profiles. Moreover, even when employing mechanisms capable of formalizing the users' preferences, a certain level of uncertainty about users' preferences will always remain. For this reason, dealing with uncertainty is an important aspect of attention management systems. Equally important is the way preferences can be derived: by explicitly specifying them or be learning techniques.

3 The SAKE Attention Model

This section presents an attention model which we have developed in SAKE[3] project. Unlike other similar models, our attention model proactively supports the user in reacting on changes respecting the user's *context* and *preferences*. The approach is realized by combining context-aware reactive rules, with ontologies, and an appropriate preference model. Such an expressive attention model is the basis of the SAKE Event-and-Context driven Attention Management System (ECAMS).

Figure 2 presents the conceptual attention model behind SAKE. The model assumes that the interactions between users and internal information sources are logged including the *business context* (e.g., business process) in which the interactions happened. Some log entries can be defined as *events* that cause *alerts*, which are related to a user, a problem domain and associated to a priority level.

[3] SAKE - Semantic-enabled Agile Knowledge-based eGovernment is an EU funded project (IST 027128): http://www.sake-project.org

Every *alert* invokes *actions*, that can be purely informative (i.e. an information push) or executable (i.e. to execute a business process).

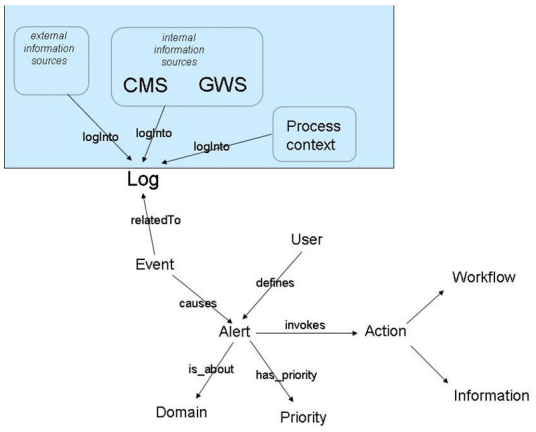


Fig. 2. Conceptual Attention Model

In the core of the SAKE approach we use a new form of reactive rules, i.e. *ECCA* (Event - Context - Condition - Action) rules; their general form is:

ON *event* WITHIN *context*, IF *condition* DO *action*.

Justification for using this new form of reactive rules in our attention management system is elaborated in Section 4.2.

Relevant events and actions are usually triggered by interactions taking place in organisational systems, such as the SAKE workflow, Content Management System (CMS) and the GroupWare System (GWS) or by external change detectors. The later are implemented with the Change Notification System (CNS), a component that can be configured to monitor web pages, RSS feeds and files stored in file servers for any change or specific changes specified by regular expressions (e.g. new web content containing the keyword "sanitary" but not "pets").

4 The SAKE Event-and-Context driven Attention Management System

Figure 3 depicts overall architecture of the SAKE ECAMS system that suits to the general EAMS framework given in Figure 1. The figure also shows other core components of SAKE, i.e. the SAKE Workflow Management System (WfM), Content Management System (CMS), the GroupWare System (GWS) and the Change Notification System (CNS). In the following, we first outline roles of

SAKE components which do not belong to SAKE ECAMS, while a more detailed description of ECAMS is given through next subsections.

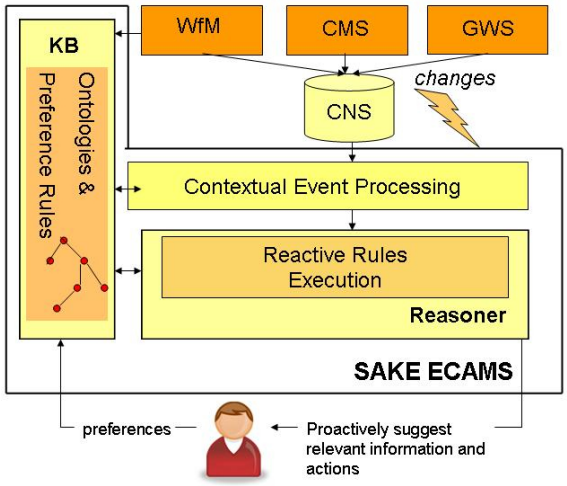


Fig. 3. SAKE Overall Architecture

Relevant events and actions are usually triggered by interactions taking place in organisational systems, such as the SAKE WfM, CMS and GWS or by external change detectors. The later are implemented with the CNS, a component that can be configured to monitor web pages, RSS feeds and files stored in file servers. The SAKE content management system (CMS) enables storage and provision of content by:

- supporting the annotation of content with metadata as well as relations between different content items;
- semi-automatic population of metadata using text mining methods; and
- realizing semantics-based search that retrieves content based on both full-text and metadata.

The SAKE GroupWare system (GWS) supports information sharing and creation by:

- supporting the annotation of the interactions between users;
- enabling identification of communities of practice from mining their interactions and their specific vocabularies by social tagging; and
- searching for experts based on their profiles as these are created explicitly and implicitly during their interaction with the system.

The SAKE workflow management system (WfM) coordinates execution of users tasks in SAKE system by:

– integrating GWS, CMS and ECAMS itself;
– strongly supporting business context management and sharing of the actual user's context.

The SAKE Change Notification System (CNS) is a server based change detection and notification system that monitors changes in the environment which is external to SAKE. It can be configured to monitor web pages, RSS feeds and contents of file servers. Users and the administrator can create new notification queries for finding and displaying interesting changes. When creating a query, users can define if they want to monitor a web page for any change or specific changes in links, words or a selected section specified by a regular expression. Moreover, users can select a topic of interest from a list. If new web page content is added that is related to the topic or an RSS feed update contains information related to the topic, then the user is notified. CNS relies on the services of Nutch (http://lucene.apache.org/nutch/), an open source crawler, which is used for fetching and parsing HTML documents, RSS feeds as well as resources in other supported formats.

In the remaining part of this section we describe the core components of the SAKE Event-and-Context driven Attention Management System.

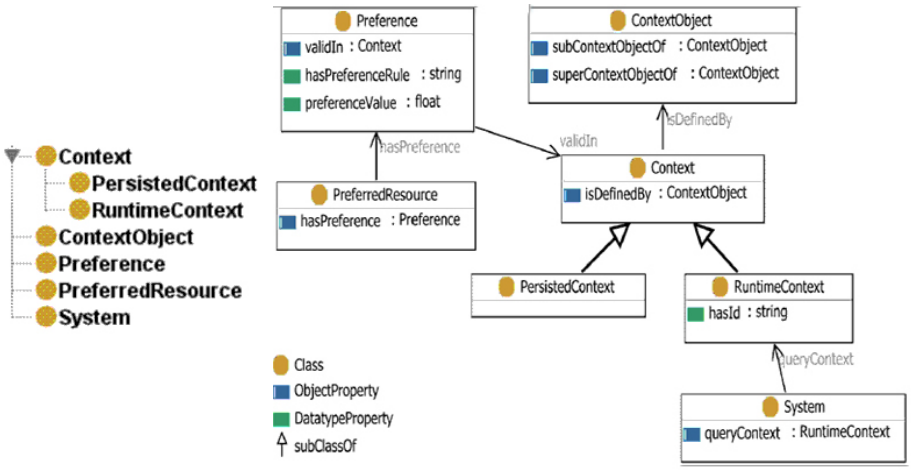
4.1 SAKE Ontologies

Conceptual information model in SAKE is realised via ontologies. For example, ontologies are used to model change events, describe various information resources, express user's contexts and preference rules etc. In the following, we further discuss roles of SAKE ontologies and outline their content.

Preference Ontology SAKE proactively delivers information resources (e.g., different documents and files) to a user. The resource delivery is realised in a process of matching the business context on one side, and user's preference rules on the other side (preference rules are described in Section 4.3). Relationship between the business context and user defined preferences is handled via the validIn relation in the preference ontology, Figure 4(b) (e.g., particular preference rule is validIn a certain context). The preference ontology 4(a) is typically imported by another ontology which maps its own concepts to this ontology. More specifically, by subclassing PreferredResource the importing ontology defines for which type of resources (i.e., individuals) the user can define preferences. Similarly, subclasses of ContextObject should be defined in order to indicate which type of individuals the Context consists of (i.e., the business process, activity, task, and the user).

Furthermore, we differentiate between a RuntimeContext and a PersistedContext. The RuntimeContext reflects the user's current context and changes dynamically with the user's interactions within the system. This context may be used to track user's behaviour in the system. However, if a user's interaction is logged in the system as a persistent activity (e.g., the creation of a new document) the user's current context will be persisted (using the PersistedContext).

The RuntimeContext and PersistedContex are utilised by the Context Observer to extract the current business context, and hence, enable resource delivery based on that business context.



(a) Class hierarchy (b) Class diagram

Fig. 4. Preference Ontology.

Information Ontology The Information ontology (Figures 5 and 9(a)) contains the concepts and relations about information resources for which we want to express preferences, such as documents and forum messages. On the top level we have separated the domain concepts from value types. The FiletypeValue class defines the different file types a file in the SAKE system can have.

In the InformationSource sub-tree we differentiate between information sources which are of an abstract nature (such as persons), external information sources such as Web pages and RSS feeds, and information sources which physically exists in the SAKE system, such as documents, forums or e-mails. We further divided the physical information sources into CMS-specific and GWS-specific entities. This FiletypeValue class represents the file type (indicated by the file extension) of a document, for example PPT, PDF, DOC, etc. Note that one subclass of filetype can describe multiple file extensions, such as JPG can be a .jpg or .jpeg file.

Log Ontology There are many sources of changes that can affect an information resource, like adding, removing, deleting a new document or starting a new discussion. The Log ontology (Figures 6(a) and 6(b)) is used for representing these changes in a suitable format. There are four subclasses of Event: AddEvent,

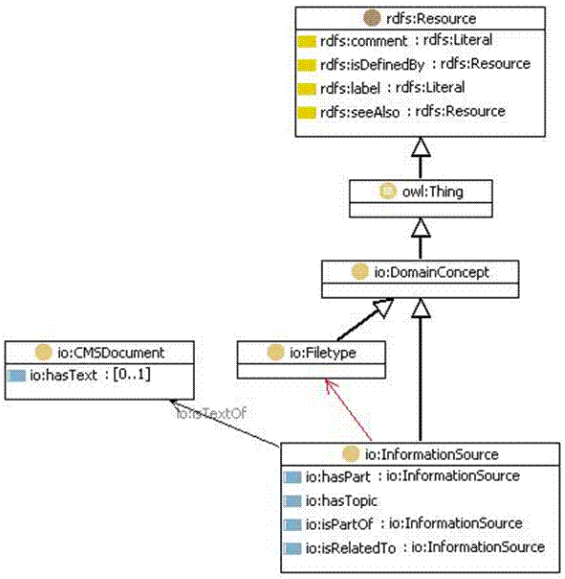


Fig. 5. Information Ontology, Class diagram

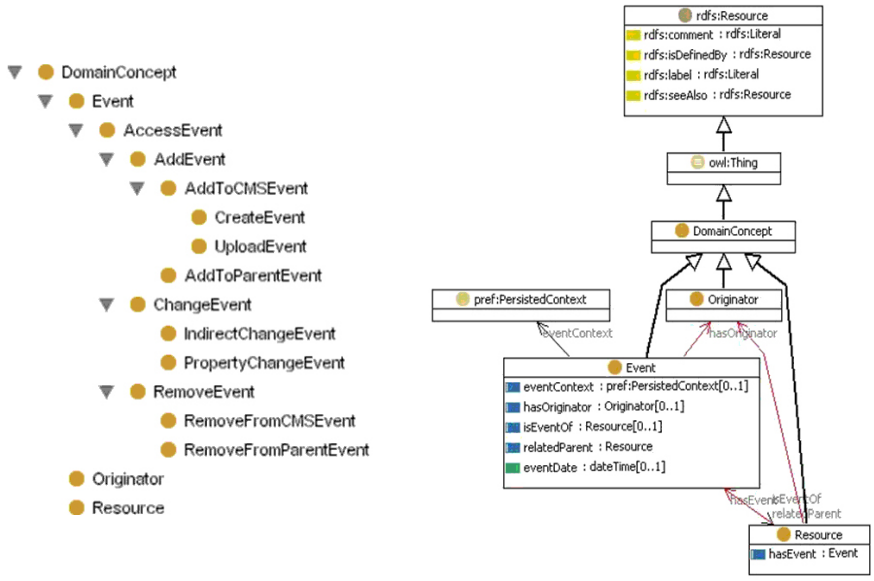
RemoveEvent, ChangeEvent and AccessEvent. AddEvent is responsible for the creation of new events, e.g. a new document has been added to the SAKE CMS. It contains two subclasses: AddToCMSEvent, meaning the addition of a resource to the CMS and AddToParentEvent, meaning the addition of an existing child to a parent element, e.g. posting a new message to a discussion thread (Figure 7).

RemoveEvent is dedicated to the deletion of the existing elements from the system, like the deletion of a document from CMS. It consists of RemoveFromCMSEvent, meaning the removal of a resource from the CMS and RemoveFromParentEvent, meaning the removal of a child from a parent element, but the child is still existent.

ChangeEvent is responsible for the modification of an existing individual, e.g., the change in the name of the author of a document. It consists of: PropertyChangeEvent, meaning that some properties of an individual have changed and IndirectChangeEvent, meaning a change caused by some other event.

AccessEvent is dedicated to the access of an existing individual. It represents a very broad class of events like reading a document, for which is very complicated to define the semantics clearly. For example, did someone who opened the document and closed it after five minutes, read the document or just opened, considered it as not interesting, but forgot to close it immediately?

We differentiate subclasses AddEvent and RemoveEvent by addition/removal of resources to/from the CMS and by addition/removal of a resource to/from a parent/child relationship using the isPartOf property. AddToCMSEvent is fur-



(a) Class hierarchy (b) Class diagram

Fig. 6. Log Ontology.

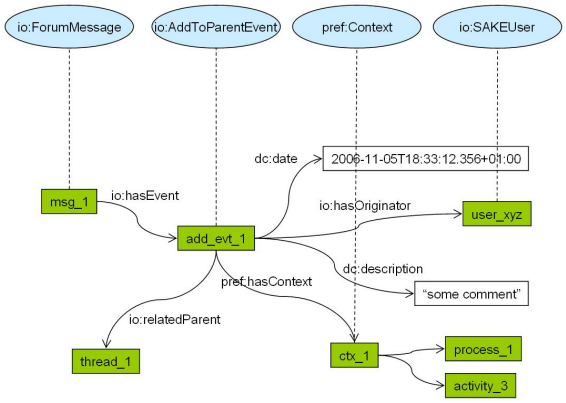


Fig. 7. AddToParentEvent for Adding a New Event

ther differentiated by either creating a resource within the SAKE system or uploading an existing resource. For ChangeEvents, we distinguish between changes of the resource's properties (e.g. metadata) and changes which are caused by some other event.

Properties of an event are the resources the event relates to, the user who originated the event, a timestamp when the event occurred, an optional de-

scription of the event and a copy of the current runtime context. In the case of ChangeEvents we add the names of the resource's changed properties, and optionally link to another event which caused this ChangeEvent.

We do not hard-code the propagation of events from child to parent, instead we define them in SWRL (Semantic Web Rule Language) rules[4], such as:

$$addToParentEvent(?E) \wedge resource(?RES) \wedge hasEvent(?RES,?E) \wedge relatedParent(?RES,?RES2) \wedge swrlx:createIndividual(?E2) \Rightarrow indirectChangeEvent(?E2) \wedge hasEvent(?RES2,?E2)$$

Fig. 8. Automated Event Creation with SWRL

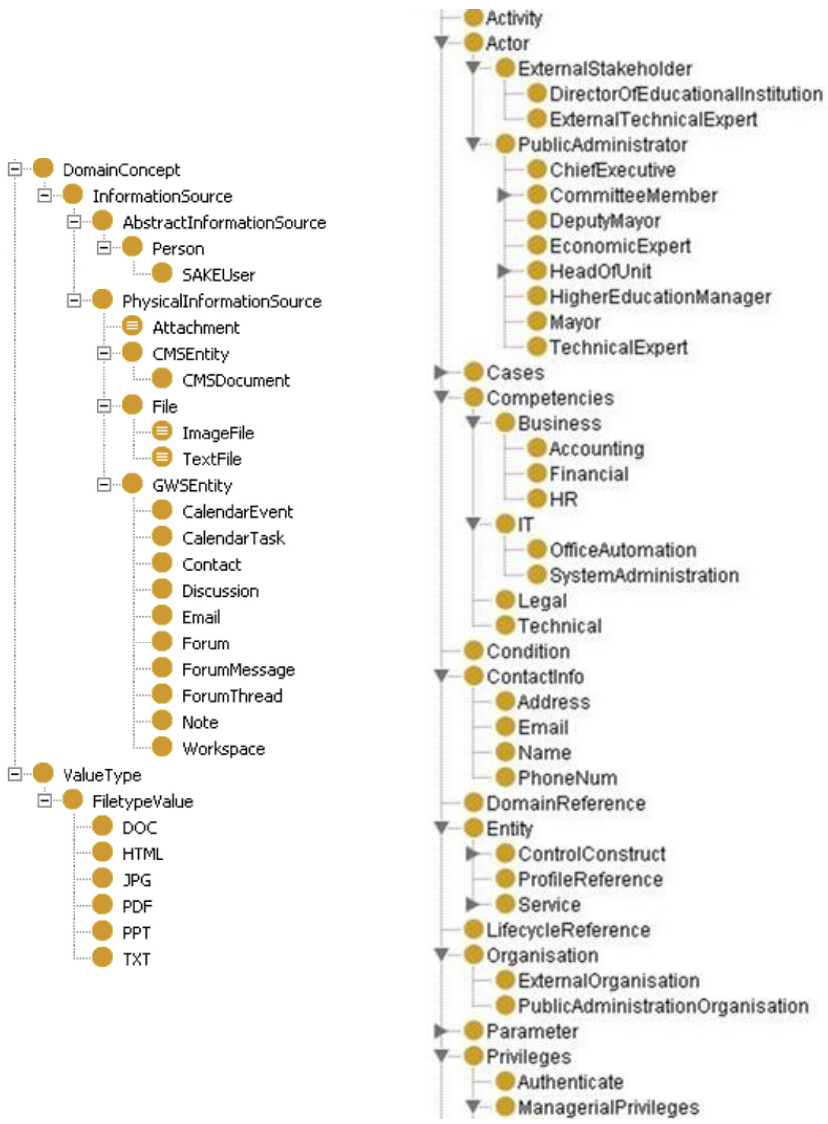
Default rules state that the addition/removal of a child object triggers a ChangeEvent for the parent object. However, in order to be more flexible, we could also state that the modification of a specific child object also causes the modification of its parent. Note that in this way, we may use events to specify more complex events (e.g., indirectChangeEvent). Those complex events are created using SWRL rules and a number of built-in predicates supported by KAON2 . Although realised in a declarative way, Complex Event Processing (CEP) in SAKE is still limited, and it is subject of our future work. Particularly, we will continue developing declarative CEP. The advantage of such an approach is that definition of a complex event may easily be altered by changing only a logical rule.Further on, inconsistencies in CEP are handled by means of logic.

Process Ontology While the other SAKE ontologies are rather general, the process ontology (Figure 9(b)) is specific w.r.t the SAKE use case scenario. The use case describes a knowledge intensive eGovernment organisation and necessities help for knowledge workers in their daily business. The process ontology describes main concepts related to various legal procedures in a municipality administration and their relationship.

4.2 Contextual Event Processing

Reactive rules (such ECA and production rules [4]) are usually used for programming rule-based, reactive systems, which have the ability to detect events and respond to them automatically. However in many cases there is a gap between current reactive rules that enable reaction to an event (change), and the reality, in which reaction is relevant only in a certain *context* [5]. As event-driven reactive systems act autonomously, a central issue is the ability to identify context during which active behaviour is relevant and the situation in which it is required.

[4] SWRL: http://www.w3.org/Submission/SWRL/



(a) Information Ontology (b) Process Ontology

Fig. 9. Ontologies (Class Hierarchies) for the SAKE Use Case.

In SAKE, the business context is derived using a context observer. This component links to enterprise systems (such as workflows) and extracts the current business process, activity, and task the user is working on. The context describes the situation which a user is currently present in, and is utilized for derivation of information resources based on context-sensitive preferences.

The business context in SAKE is formally described (see Section 4.1), which in turn allows contextual reasoning with reactive rules. In classical reactive systems [6, 7] based on Event-Condition-Action (ECA) rules, reaction (i.e., action) is triggered by an event, provided that the condition holds. Typically the condition part provides the contextual background information. However this way of expressing the context is rather limited. In our opinion, an automated reactive system needs to be capable to deal with more complex business contexts and situations, and hence needs to *reason* before undertaking any action [8]. This is why we introduce ECCA rules where the context is explicitly represented[5]. Moreover the context is formally realized by means of ontologies, and hence allows for automated reasoning techniques to be applied.

Depending on a user's current working context (i.e., business process, activity and task), the reasoner in SAKE ECAMS automatically searches for relevant knowledge artifacts. In a nutshell of the process, the inference engine takes the user's business context (from the workflow), and gives back relevant information resources. What is relevant to a user does not depend only on a particular business context, but also on the user defined preference rules (described in the following Section 4.3).

4.3 Preference Rules

A preference is an n-ary relation between a user, multiple resources, and a preference value. Figure 10 shows how such a preference relation is formally modeled using the preference ontology.

Each preference (i.e., n-ary relation) is expressed as a deductive rule [9], represented in SWRL. Figure 11 illustrates an example of a preference rule: if userA is in the processZ, then userA has preference of value 1.0 for documents created in 2006. Among the preferred values, preferences include the business context of the user, in order to support context-awareness of the whole system (e.g., userA and processZ are related to each other by the same runtime context: ctx.

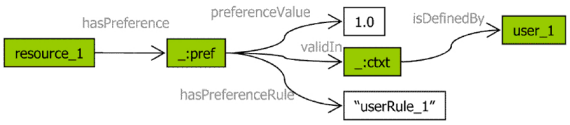


Fig. 10. Preference as n-ary relation between a resource, value, user and rule

Utilising logical rules, for expressing context-sensitive user preferences, SAKE features a very flexible preference model. One rule is used to assign different

[5] the condition part in an ECCA rule is used for representing simple condition - one that requires no reasoning (as complex computation) in order to be evaluated.

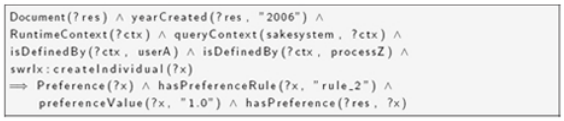


Fig. 11. A Sample Preference Rule Expressed in SWRL

preference values to different information resources based on relevant criteria of a particular user. Therefore every information resource may be assigned with different preference values by different preference rules (i.e., by different users and/or business contexts). Another flexibility of the SAKE preference model comes from an implicit representation of preferences. Since preference values are not pre-computed and persisted in the system, just adding one preference rule may significantly influence the whole preference model. Also adding a common preference to the SAKE preference model (i.e., a preference valid for all users) may be as easy as adding only one preference rule. Moreover updating existing resources, or adding new ones, does not mess up all previously created preference values. In this way, a user is given a great freedom to create particular preferences for particular processes, activities, tasks, and even to aggregate multiple preference values for one resource into a final score.

The Preference Editor supports creation of preference rules by providing a GUI for step-wise, interactive rule development, as presented in Figure 12. The user starts with selecting what kind of resources (i.e., file, forum, workspace, email etc. that is a subclass of pref:PreferredResource) s/he wants to define a preference for. This information is specified in the information ontology (Figure 9(a)), and is represented as a variable ?res in Figure 11. The preference rule, is than, further extended narrowing down the preference criteria in several subsequent steps (possible introducing new variables). For each of these steps, SAKE reasoner is used to find out the list of possible properties or property values that are available. Further on, values entered by a user are syntactically checked out (e.g., for the data type). In this way, the Preference Editor eases the process of creating valid and consistent preferences.

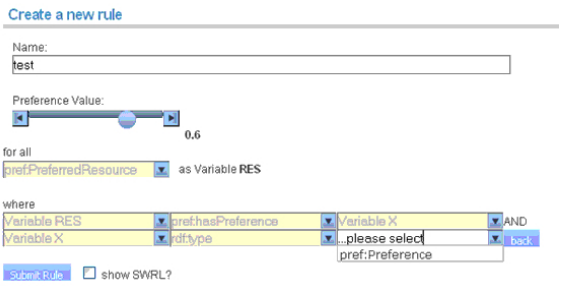


Fig. 12. Preference Editor: Step-wise, interactive rule development

Preference rules, created by the editor, are serialised to its SWRL representation and stored in the preference ontology. Finally, preference rules may also be removed (or updated) using the Preference Editor.

5 Conclusion and Future Work

In this paper we presented a novel approach for managing user's attention. The approach is realised through a reactive system that manages not only alerts to a user when something has been changed, but also supports the user to react properly on that change. In a nutshell, the corresponding system is an ontology-based platform that logs changes in internal and external information sources, observes user context and evaluates user attentional preferences represented reactive rules.

The presented system is currently under deployment in a real-environment. We have developed the first prototype, however results from the formal evaluation are still missing.

Future work of the attention management framework, presented here, will go toward a logic based event-driven reactive system (see [8]). Such a system will be fully automated and controlled by a state-changing logic. Furthermore the system will allow new reasoning services, e.g. reasoning about conflicting business contexts and situations, as well as, causal relationships between complex events, conditions, and actions.

References

[1] T.H. Davenport and J.C. Beck. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business School Press, Cambridge, MA, 2001.

[2] Jr Norton, William I and William T Moore. Entrepreneurial risk: Have we been asking the wrong question? In *Small Business Economics*, 2002.

[3] C. Roda and J. Thomas. *Attention Aware Systems: Theory, Application, and Research Agenda*. Computers in Human Behaviour 22, 557–587, 2006.

[4] B. Berstel, P. Bonnard, F. Bry, M. Eckert, and P. L. Patranjan. Reactive rules on the web. In *Reasoning Web*. Springer, 2007.

[5] David Botzer Opher Etzion Asaf Adi, Ayelet Biger and Ziva Sommer. Context awareness in amit. In *Active Middleware Services*, 2003.

[6] Norman W. Paton, F. Schneider, and D. Gries. *Active Rules in Database Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[7] Jennifer Widom and Stefano Ceri. *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, 1st edition, 1996.

[8] Stojanovic N Anicic D. Towards creation of logical framework for event-driven information systems. In *To appear in: 10th International Conference on Enterprise Information Systems*, 2008.

[9] J. W. Lloyd. *Foundations of Logic Programming*. Computer Science Press, 1989.