Supporting Adaptive Case Management Through Semantic Web Technologies

Conference Paper · January 2015

4 authors, including:

Marian Benner-Wickner
IUBH – Internationale Hochschule

Matthias Book
University of Iceland

Volker Gruhn
University of Duisburg-Essen

Supporting Adaptive Case Management Through Semantic Web Technologies

Marian Benner-Wickner[1], Wilhelm Koop[1], Matthias Book[2], Volker Gruhn[1]

[1]paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen, Essen, Germany
{marian.benner,wilhelm.koop,volker.gruhn}@uni-due.de

[2]Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
University of Iceland, Reykjavik, Iceland
book@hi.is

Abstract. In the rehabilitation management domain, we find many situations where actors have to manage complex cases. To facilitate patients' quick recovery from their individual conditions, case managers need a high degree of flexibility in organizing their tasks. Unfortunately, giving them complete flexibility is challenging for two reasons: Firstly, process owners may want to tame the flexibility to conform with compliance policies. Secondly, without complete information about the possible processes in the problem domain, software engineers are struggling to design information systems that can support these case management processes effectively. In this paper, we will therefore show how semantic web technologies can complement adaptive case management techniques in order to cope with the cases' flexibility. Following the ideas of linked data and the open-world assumption, these techniques facilitate (1) a data structure that is easily extendable, (2) data quality improvements, and (3) the definition and checking of business rules using domain concepts. As a proof of concept, we integrated the method into a case management tool and conducted a small case study using real-life examples from the rehabilitation domain.

Keywords: Adaptive Case Management, Semantic Web, Agenda-driven Case Management, Business Rules

1 Introduction

In the case management domain, business processes are less-structured and hence not automatable. Due to the high diversity between cases, there is no predefined process model that can be used by information systems to prepare the tasks for a case manager. Instead, case managers need a high degree of flexibility when organizing their tasks and gathering information from different sources. As a result, some researchers aim at making control flow process models more flexible. The case handling approach introduced by van der Aalst et al., for example, introduces the ability to skip or redo tasks in a model [1]. However, such models are usually still imperative (i.e. they only

adfa, p. 1, 2011.

allow control flows that are explicitly defined). This is an issue in case management processes, because they cannot cope with sudden events forcing the case manager to redesign the process with alternative steps. Based on the reasoning of Rychkova and Pesic [11, 15], we believe that allowing only actions defined in such models will never satisfy the degree of flexibility needed in case management. Instead, we suggest using declarative models, which allow everything that is not explicitly forbidden by defined constraints. The constraints can usually be derived from business rules of the given domain.

This paper is based on the hypothesis that this paradigm switch matches concepts of the semantic web and agenda-driven case management (adCM [2]) well. We therefore connect these concepts in order to find new opportunities for supporting case management processes. In this section, we will first describe how agenda-driven case management and semantic web concepts can each contribute to satisfying the need for flexibility resulting from the above-mentioned paradigm switch. In Sect. 2, we then discuss benefits that could be gained by connecting both fields, and which novel capabilities for case management support this would enable. In Sect. 3, we then show how such a connection can actually be implemented, and demonstrate some of the resulting features in a prototype of a “Semantic Case Management” tool. We compare the approach with the related work in Sect. 4 and finish with a conclusion in Sect. 5.

1.1 Agenda-driven Case Management

In our previous work, we introduced adCM as a new approach to support case management processes. Due to the high diversity of such processes, the approach reflects the need for high flexibility in both the control flow perspective as well as the data perspective. It also provides the capability to record process knowledge during case execution. In this section, we will briefly introduce these concepts; details can be found in [3].

Agenda. The pivotal element of adCM is a hierarchically structured list containing all entities that a case manager considers important for the case’s execution – the so-called agenda. Ideally, it is derived from common process knowledge discovered in past cases. However, the case manager is free to (re-)organize the agenda according to the particularities of each case instance, and may also start with a blank agenda. Due to this flexibility, each case manager can use agendas according to his preferences, which is in contrast to the traditional workflow management approach that tries to bind the process execution tightly to predefined control flow definitions. Consider, for example, the rehabilitation of a patient Mr. Smith. His case is managed by Mr. Fisher, who initially fills his agenda with common important process landmarks like “healing plan”, “issue orders” and “care provider ranking” (Fig. 1). They could originate from a knowledge base containing common agenda items. The subordinate items, however, are specific to the Mr. Smith’s case instance and have been added by the case manager. According to the patient’s symptoms, the healing plan contains individual indications (e.g. an ambulatory physiotherapy following an occupational rehabilitation).

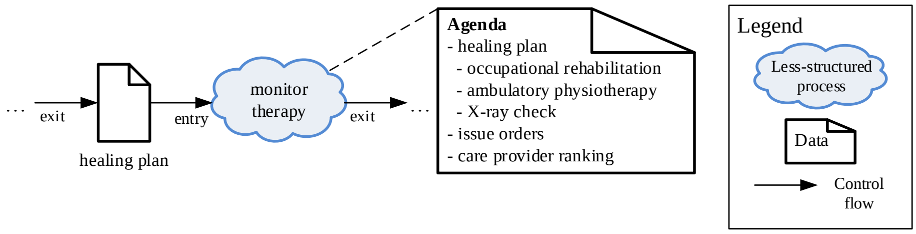

Fig. 1. Example agenda of a case from the rehabilitation management domain

Artifacts. One of the most frequently performed activities of a case manager is to search and explore information related to the case. For this purpose, the case manager will access multiple information sources, both internal and external to his organisation, some of which are structured, while others are not. We collectively call all information that the case manager gathers "artifacts". He can work with these by combining, evaluating, rating and relating the information they contain, and structure them by collecting related artifacts in individually defined workspaces. For example, Mr. Smith's healing plan is a key artifact in the rehabilitation domain. To organize the task's artifacts in the workspace, the case manager can include information and documents from heterogeneous sources. He can arbitrarily associate artifacts in his workspace with each other, and annotate these artifacts and associations with tags. In the course of the task completion, the case manager can dynamically re-arrange and relate the artifacts in his workspace. The adCM system architecture [3] is designed to store workspaces using a central knowledge base. This enables the adCM tool to perform data mining steps aiming to identify important artifacts.

1.2 Semantic Web Paradigms

Until recently, most of the information present on the Internet was meant to be used by humans, not machines. In the past decade, this was a challenge for those aiming to analyze web content automatically. Today, a set of technology standards focus on this issue, representing what is commonly known as the semantic web. We identified two main concepts of the semantic web that we believe are promising for case management support: linked data [5] and the open-world assumption [13].

Linked Data. Humans understand concepts by connecting them with terms already learned. When communicating, one can find out whether or not (most of) the concept's attributes match with those mentioned by another person, indicating that you are talking about the same term. In order to make terms comprehensible to machines, they need a capability to store and map terms with key attributes and other terms. In the semantic web, URIs are used to name things. They can be associated with each other using named relations, called properties. The corresponding graphs are stored using the resource description framework (RDF) [17]. The following listing shows an example of how to define a new class for healing plan artifacts using the turtle syntax

[18]. It is organized using triples of subject-predicate-object order. The prefix adcm denotes that the term is a core adCM concept.

```
adcm:Artifact rdf:type :Class .
adcm:Document rdf:type :Class .
adcm:Document rdfs:subClassOf adcm:Artifact .
adcm:HealingPlan rdf:type :Class .
adcm:HealingPlan rdfs:subClassOf adcm:Document .
```

This way, machines managing their own graphs of connected information can compare, exchange, or verify their knowledge. Software systems called reasoners can also infer additional properties using first-order logic. Using the example above, a reasoner can infer transitive relations such as that any healing plan is also an artifact.

Open-World Assumption. When querying information systems, users assume them to be complete and accurate reflections of the world. That is, persons not listed in a phonebook database are assumed not to be reachable by phone. This closed-world assumption works well in well-known domains and processes with little uncertainty, as these systems can actually contain all the information they need. However, in some domains like healthcare, knowledge is rarely known to be complete, as the problem domain is not entirely explored. In this case, queries for information should be understood to return an incomplete, but steadily expanding body of knowledge. Consider, for example, a case manager searching for therapies that fit a given set of diagnoses. Assuming that the search fails, this does not imply that a therapy is impossible. Instead, it is not decidable whether or not a therapy exists (it may just not be explored yet). In the field of semantic web, this concept is called the open-world assumption.

2 Benefits of Semantic Web Technologies for adCM

In this section, we will show how the connection of agenda-driven case management with semantic web technologies facilitates (1) a flexible persistency layer for process knowledge base, (2) high data quality within the knowledge base, and (3) business rule definition using terms of the given domain.

2.1 Flexible Persistency

As described in Sect. 1.1, adCM's agenda lets the case manager organize any case-relevant activities flexibly, while the workspace enables him to organize any data sources as artifacts. Both concepts put considerable flexibility demands on the data model used to store agendas and artifacts. For example, the set of artifact types is specific to the domain or business process and will change in time. Based on our experience in the field, we assume that most of the artifacts will be documents, images, spreadsheets and web links. However, it is impossible to consider this set complete. Therefore, it has to be easy to integrate new types into the data model at runtime. When using relational databases, this would impose serious efforts in schema refactoring. We therefore propose to adapt the open-world assumption of the semantic web: Using description logic, we specify only an incomplete set of common types and rules on how they can be used. This set, representable as an RDF graph, can be appended at runtime by just adding further relations. The information about each specific case, containing its agenda with all associated agenda items and artifacts, can be represented as “individuals” in this graph. The graph then acts like a semantically connected knowledge base for the case management process. Sect. 3.2 will provide more details about the design of this solution approach, including examples.

2.2 Improving Data Quality and Analysis Outcome

Since names of agenda items, artifacts and their tags can be chosen freely by case managers, the names are subject to (conceptual) noise. A software system cannot understand that different terms or spellings may be synonyms, i.e. that they have the same meaning. This data quality issue is a challenge for data mining and hampers the effectivity of recommender systems. However, semantic web technologies are capable of addressing this issue. Following the idea of linked data, we can attach additional sources of knowledge such as lexical databases or domain ontologies to the common types. This way, synonyms can be identified within the knowledge base.

Moreover, linking domain ontologies does not only help to fix case data quality issues “a posteriori”. It can also help to keep the quality level high by preventing the creation of synonyms “a priori”. To achieve this, the concepts from the ontologies can be integrated into an unobtrusive auto-complete function when naming or tagging agenda items, for example. This way, a streamlined vocabulary can be established and noise and misspellings can be reduced without burdening the user with dialogs.

2.3 Business Rule Definition and Checking

Given that our approach entailed allowing everything that is not explicitly forbidden, we do not just need mechanisms for providing flexibility, but also for limiting the case manager’s scope of action. For this purpose, business rules are useful. However, defining business rules is complicated if the modeling language does not employ domain-specific concepts. Let us assume, for example, that the process owner in Mr. Fisher’s organization wants to ensure that a case can only be closed when a healing plan has been attached. Without using domain-specific concepts, such a definition is less expressive, resulting in a cumbersome rule like “When a case contains any agenda item containing an artifact with a name similar to ‘healing plan’, then the case can have a property named ‘closed’.” Using concepts defined in a domain-specific ontology, the rule could be more expressive: “When a case contains a healing plan, it can be closed.” OWL, one of the key languages of the semantic web, enables the formal definition of such rules based on description logic. Given such rules, reasoning engines are capable of checking whether rules are violated in the case data. In Sect. 3.3, we will show how this approach can be used in more detail.

3 Semantic Case Management

In this section, we will show how the benefits discussed in Sect. 2 can be achieved by integrating the underlying semantic web technologies into the adCM approach. We will refer to this augmented approach as Semantic Case Management (SCM).

3.1 Flexible persistency

To enable tool support that can be customized to the actual working environment of the case managers, domain experts need to be able to supply the adCM toolset with domain-specific information when adopting adCM in an organization. We therefore define a meta-model that describes the core concepts of adCM but is open for custom extensions. We chose to use a logic-based language (in this case OWL) for this purpose since the definition of business rules (described in Sect. 3.3) will require reasoning capability in the data model.

Since OWL is based on the open-world assumption, we can define the adCM concepts in the meta-model, but also allow the user to supplement additional semantics about the application domain. As described later, the user can define and reason about any information, as long as he does not violate adCM's pre-defined core rules.

For the meta-model, we first defined RDF classes for each existing concept, like a case, agenda item and artifact. Each class contains properties to describe required information of the concepts. A case, for example, has a Boolean property which describes if the case is closed, whereas artifacts contain a creation date, represented by a literal of type *xsd:DateTime*. In adCM, the concepts are related to each other in distinct ways: An agenda item can contain artifacts and multiple child agenda items, an agenda contains an ordered sequence of agenda items, and so on. We can describe the semantics of these relations by RDF properties. To describe, for example, that an agenda item can contain an artifact, we first have to define a property:

```
adcm:containsArtifact rdf:type owl:ObjectProperty
```

Without restricting its allowed subject (agenda item) and object type (artifact), the property could be used in violation of its intended semantic, causing invalid conclusions. Following the idea of a declarative model allowing everything that is not explicitly forbidden, we have to prevent such (accidental) misuse of adCM concepts. Therefore, we declare the allowed subject and object type using RDFS (RDF Schema):

```
adcm:containsArtifact rdfs:domain adcm:AgendaItem .
adcm:containsArtifact rdfs:range adcm:Artifact
```

With this definition, the property `adcm:containsArtifact` can only be used to create a relation from instances of the type `adcm:AgendaItem` to instances of the type `adcm:Artifact`.

As a next step, we will show how domain experts can adapt this approach to augment the adCM meta model with additional concepts of the domain. A basic domain-specific extension to the meta model is the definition of specialized subclasses of adCM concepts. In rehabilitation management, for example, the healing plan as a central concept can be defined as a subclass of adcm:Document, as shown below. That way, it will be treated as an ordinary document by the adCM toolset, but could be specialized with properties and business rules from domain specialists (as described in Sect. 3.3). The prefix rehab denotes that the concept is neither an OWL conecpt, nor an adCM concept, but from rehabilitation management domain:

```
rehab:HealingPlan rdfs:subClassOf adcm:Document .
```

A more advanced scenario for extending the meta-model with domain-specific concepts is the definition of custom properties. Consider, for example, that an extension should allow to add due dates to agenda items and artifacts. Regardless of its realization in the UI, changes in the data structure are minimal: We can define a new literal property `rehab:dueDate` which is of type `xsd:dateTime` and can be added to `adcm:AgendaItem` or `adcm:Artifact`. Due to the extensible meta-model, the additional information is stored in the same place and in the same way as the existing data. Compared to traditional relational data models, there is no need for expensive and risky schema migration [6].

3.2 Improving Data Quality and Analysis Outcome

As motivated in Sect. 2.2, a sound vocabulary is important for the success of data mining algorithms. Identifying common agenda items is difficult and error-prone if each case manager uses a different spelling or term for the same concept. The real frequency of common agenda items or artifacts can only be measured when resolving these naming issues. Also, when annotating agenda items and artifacts with tags, it is important to use a few distinct and well-known keywords. Unlike relational databases, graph structures used in the semantic web can cope with such issues. A lexical ontology like WordNet can be used to identify semantically similar agenda items by sets of synonyms ("synsets"). For example, if a colleague of Mr. Fisher uses the word "recuperation" instead of "healing", WordNet shows that both are in the synset of the other word. In addition, domain-specific ontologies such as ICD-10 (the international statistical classification of diseases and related health problems) can be used to identify different spellings. Consider for example another colleague of Mr. Fisher, who is using only ICD codes to denote diseases instead of human-readable labels (as shown in Fig. 2). Any algorithm that is not aware of these domain-specific terms would always handle codes and labels separately. As a result, it would underestimate the relevance of a common concept, just because its usage is fragmented.

We identified two different strategies for integrating such ontologies into an adCM implementation: Firstly, the ontology can be used as a lookup service. That is, a mining algorithm can use the ontology as an API providing synonyms for a given term. This strategy is recommended for big ontologies where the benefit of directly accessing linked information is eradicated by integration effort and strongly decreased reasoning performance.

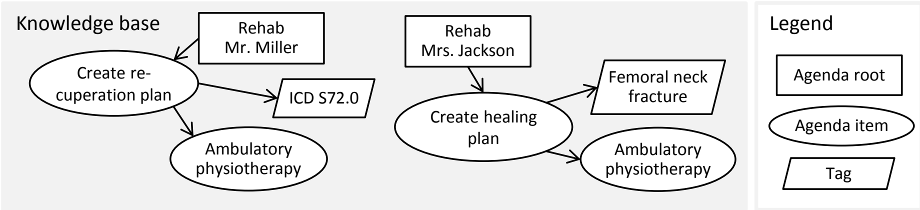

Fig. 2. Two similar cases containing different labels

This improves data quality and provides additional semantics, directly usable by all SCM tools and enhancements. Although this approach is closer to the linked-data design principle, this step should be executed before any case data is stored in the knowledge base. Otherwise, it would be time-consuming to connect all the existing terms with the vocabulary using properties like "same-as".

In our SCM prototype, we followed the first strategy and integrated an ontology as a lookup service, linking only analysis results in the knowledge base. This allows for easy change of the ontology in use when improving the analyzing algorithms or adding domain specific information. The service is employed by a synonym detection function of an algorithm identifying common agenda items. It preprocesses all agendas by replacing the agenda item names with IDs of their corresponding synsets. Frequent patterns within agendas are then calculated using the cluster synset IDs instead of natural language (which is also faster).

3.3 Business Rule Definition and Checking

As outlined in Sect. 2.3, business rule definitions are much more precise when using domain concepts. In this section, we will show how SCM supports process owners in defining and checking such business rules. We will do this using an example from rehabilitation management: At the beginning of each case, the rehabilitation manager has to classify the case using four groups (as shown by the decision tree in Fig. 3). The rule requires that each group involves a specific package of measures.

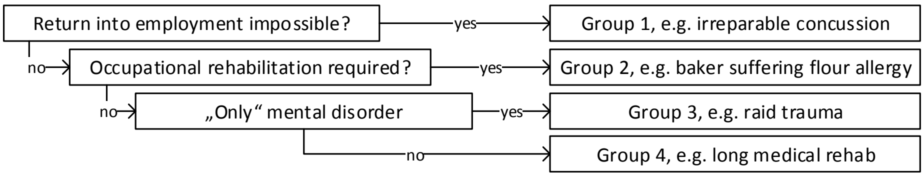

Fig. 3. Case classification decision tree (group numbers do not imply any order)

In this example, it would be beneficial if a SCM implementation could "understand" the classification and check, based on that knowledge, if the rehabilitation manager has applied the required measures. To achieve this, the classification has to be transformed into a formal language, so that a reasoning engine, as part of the SCM toolchain, can infer the classification itself. Assuming that the rehabilitation manager applies the adCM annotation function [4], he tags one of the pivotal agenda items or artifacts using keys like “employment impossible” or “reintegration impossible”. This annotation is stored in the meta model using the `adcm:hasTag` property. Using the Apache Jena rule syntax, this example can be defined formally as follows:

```
(?x adcm:hasTag ?Tag)
(?Tag adcm:name "employment impossible")
-> (?x rdfs:subclassOf rehab:Group1)
```

Based on the standard RETE algorithm [7], the Jena reasoner can then classify the corresponding group by inference. Of course, the formal definition can be extended anytime, for example due to changes in the group definition. Based on this reasoning capability, the SCM implementation can query if the required package of measures has been applied (e.g. using the query language SPARQL).

Note that the first rule discussed above is a derivation rule according to the Semantics Of Business Vocabulary And Business Rules (SBVR) standard [10]. That is, it generates new information based on existing data. However, this only covers a subset of possible business rules. Another type of rules would be a static constraint like “A healing plan must contain a due date”. To achieve this, the process owner has to define another formal rule. Given the due date example, we propose to use the violation property of Jena triggering an error message if the constraint is violated:

```
(?hp rdf:type rehab:HealingPlan)
noValue(?hp owl:onProperty rehab:dueDate)
-> (?hp rb:violation error(‘No due date!’))
```

3.4 Prototype Implementation

To conduct a small case study, we implemented the SCM approach in our adCM system prototype and defined the real life rules given above. Table 1 shows how we mapped the requirements with available (semantic) web technology:

Table 1. Requirements/Technology-Mapping

Requirements	**Technology**
Persistence Layer: Graph-based triple store	Apache Jena TDB
Data Access Layer: Interface for RDF data queries	Apache Jena Fuseki
Business Logic Layer: Manipulating RDF data	Apache Jena Ontology API
Presentation Layer: Business rule definition UI	Bootstrap
Operations: Application Server	WildFly (formerly JBoss)

When implementing the persistence and data access layer, one issue we had to address was bad performance, caused by our excessive use of reasoners to include domain knowledge and check business rules. The high expressiveness of OWL leads to reasoning algorithms with exponential time. Since the adCM approach aims at giving feedback in seconds, this is not acceptable. To address this problem, we optimized our system by replicating only case-specific data together with all rules locally on the client and synchronizing it with the complete case base on client shutdown. This solution scales very well as the small data sets are independent of each other and can benefit from load balancing and multiple processors or servers.

Another major issue when implementing SCM is usability. Defining classes and business rules using OWL syntax requires a significant amount of language expertise. Apart from the SPARQL interface provided by Fuseki, there is no distinct semantic web technology able to abstract from these languages. Therefore, we implemented a user interface (UI) for business rule definition and checking based on the Bootstrap framework. It provides both a dashboard for a compliance overview and a rule editor using a responsive design. Figure 4 shows the rule editor containing a business rule that corresponds with the case classification example. It uses a tree view to visualize the dependencies between predefined adCM meta model elements as well as domain-specific elements, if specified. If the pattern is found, the user-defined action is triggered. As long as the rules are not active, they are stored as Jena rule strings in a relational database. However, when becoming active, they are imported into the graph model and checked using the Jena reasoning engine.

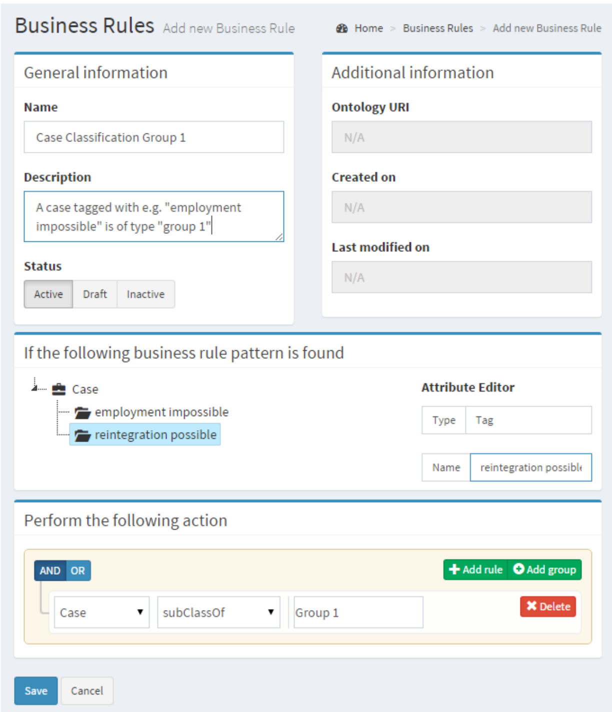

Fig. 4. SCM UI for developing business rules

As a case study, we defined the real life classification rules and the due date constraint rule and checked them against a model containing 30 artificial cases. Running on an Ubuntu server with an Intel Xeon L5640 CPU (2.27GHz), the validation took only a few seconds. To demonstrate how the UI exposes violations, we manipulated two cases so that they do not comply with the due date constraint (i.e. they are "invalid"). Fig. 5 shows a screenshot from the rule dashboard.

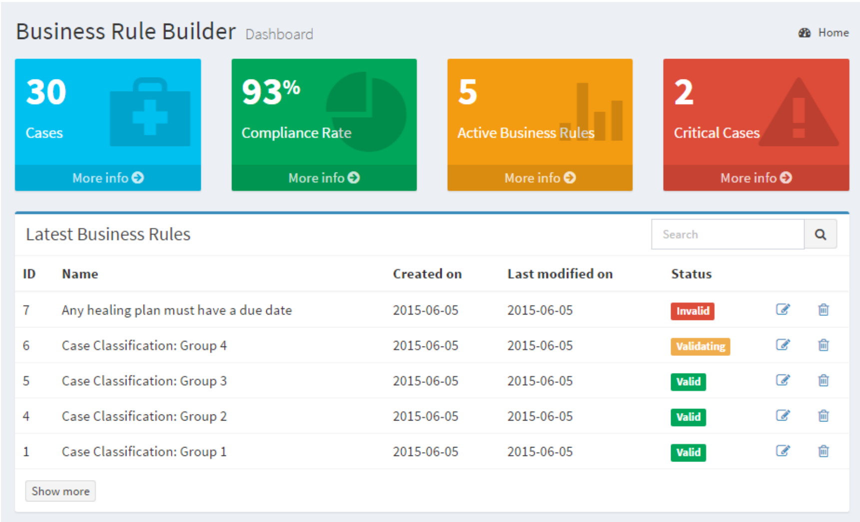

Fig. 5. SCM rule dashboard

4 Related Work

In the field of ACM, only a few researchers explicitly use semantic web technologies. For example, Heil et al. use the linked data concept to apply social routing to ACM [8]. Ruiz et al. adapt semantic web technologies to knowledge work in general, with a clear focus on privacy policies in collaborative environments [14]. McGlothlin proposes how to connect domain-specific ontologies in healthcare [9].

There are also works discussing the application of declarative business logic, which is very similar to our approach. Pesic et al. [12] and Rychikova et al. [15], for example, discuss how process models can be supplemented with declarative models. However, these approaches still focus on control flow constraints and neglect the data-centric notion of adaptive case management.

Voigt and Lehner introduce a flexible relational data model to cope with high mutability and variety in data [16]. Nevertheless, the approach lacks of the advantages that come with linked data (e.g. attaching custom ontologies using URIs).

5 Conclusion

In this paper, we discussed how semantic web paradigms and the underlying technologies can be utilized in the domain of adaptive case management (adCM). Linked data, for example, is a paradigm that can help to integrate domain-specific ontologies into the core adCM meta-model. Using languages like RDF and OWL, case data can be stored in graph structures according to formal semantics. This way, semantic reasoners can derive new facts or check the model against business rules defined by the process owner. In contrast to control-flow approaches, this approach follows the idea of a case management solution allowing sufficient flexibility for previously unknown process variations, but also providing process owners with the capability to define boundaries of the flexibility. This facilitates a paradigm shift from normative models, such as BPMN, to declarative models, such as description logics.

As a proof-of-concept, we integrated the semantic web technologies into our prototype case management system. Since the formal definition of business rules is quite difficult, we also implemented a business rule UI including a compliance dashboard for process owners. We plan to extend the functionality of the prototype so that new properties and classes can be added to the model, and additional types of business rules (according to SBVR) can be defined.

References

[1] Aalst, W. M. van der, Weske, M., and Grünbauer, D. 2005. Case handling: a new paradigm for business process support. *DKE* 53, 2, 129–162.

[2] Benner, M., Book, M., Brückmann, T., Gruhn, V., Richter, T., and Seyhan, S. 2012. Managing and Tracing the Traversal of Process Clouds with Templates, Agendas and Artifacts. In *Business Process Management*. Springer, 188–193.

[3] Benner-Wickner, M., Book, M., Brückmann, T., and Gruhn, V. Execution support for agenda-driven case management. In *ACM 2014*, 1371–1377.

[4] Benner-Wickner, M., Book, M., Brückmann, T., and Gruhn, V. 2013. Cloud Storage of Artifact Annotations to Support Case Managers in Knowledge-Intensive Business Processes. In *WISE 2011 and 2012 Workshops*, 92–104.

[5] Berners-Lee, T. 2006. *Linked Data* (http://www.w3.org/DesignIssues/LinkedData.html). W3C.

[6] Curino, C., Moon, H. J., Tanca, L., and Zaniolo, C. Schema Evolution in Wikipedia - Toward a Web Information System Benchmark Enterprise Information Systems, Volume DISI, Barcelona, Spain, June 12-16, 2008, 323–332.

[7] Forgy, C. L. 1990. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Expert Systems. In *Expert Systems*, P. G. Raeth, Ed. IEEE Computer Society Press, Los Alamitos, CA, USA, 324–341.

[8] Heil, S., Wild, S., and Gaedke, M. 2014. Collaborative Adaptive Case Management with Linked Data. In *WWW'14 Companion*. International World Wide Web Conferences Steering Committee, Geneva, Switzerland, 99–102.

[9] McGlothlin, J. P. and Khan, L. Managing evolving code sets and integration of multiple data sources in health care analytics. In *DARE 2013*, 9–14.

[10] OMG. 2013. *Semantics of Business Vocabulary and Business Rules (SBVR),* formal/2013-11-04. http://www.omg.org/spec/SBVR/1.2/PDF.

[11] Pesic, M. 2008. *Constraint-Based Workflow Management Systems: Shifting Control to Users,* Eindhoven University of Technology.

[12] Pesic, M. and van der Aalst,W. M. P. 2006. A Declarative Approach for Flexible Business Processes Management. In *Business Process Management Workshops*. LNCS. Springer, Berlin, Heidelberg, 169–180.

[13] Reiter, R. 1978. On Closed World Data Bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Springer US, Boston, MA, 55–76.

[14] Ruiz, C., Álvaro, G., and Gómez-Pérez, J.-M. A framework and implementation for secure knowledge management in large communities. In *I-KNOW '11*, 1.

[15] Rychkova, I. 2013. Towards Automated Support for Case Management Processes with Declarative Configurable Specifications. In *Business Process Management Workshops*. LNBIP. Springer, Berlin, Heidelberg, 65–76.

[16] Voigt, H. and Lehner, W. 2014. Flexible Relational Data Model – A Common Ground for Schema-Flexible Database Systems. In *Advances in Databases and Information Systems*. LNCS. Springer, Cham, 25–38.

[17] W3C. 2014. *RDF 1.1 Concepts and Abstract Syntax*. http://www.w3.org/TR/rdf11-concepts/. Accessed 7 May 2015.

[18] W3C. 2014. *RDF 1.1 Turtle: Terse RDF Triple Language*. http://www.w3.org/TR/turtle/. Accessed 7 May 2015.