On maintaining semantic networks: challenges, algorithms, use cases

Klaus Ulmschneider
Institute of Artificial Intelligence, Ulm University, Ulm, Germany

Bernd Michelberger
University of Applied Sciences Ravensburg-Weingarten, Weingarten, Germany

Birte Glimm
Institute of Artificial Intelligence, Ulm University, Ulm, Germany

Bela Mutschler
University of Applied Sciences Ravensburg-Weingarten, Weingarten, Germany, and

Manfred Reichert
Institute of Databases and Information Systems, Ulm University, Ulm, Germany

Received 8 April 2015
Revised 8 April 2015
Accepted 17 April 2015

Abstract

Purpose – The purpose of this paper is to provide methods and algorithms to maintain a semantic network (SN). In previous work, the authors introduced the SN approach for bridging the gap of aligning enterprise information with business processes, i.e. for discovering explicit relations between them. What has been neglected so far, however, is SN maintenance, which is required to keep an SN consistent, complete and up-to-date.

Design/methodology/approach – The paper illustrates an approach for SN maintenance. Specifically, the authors show how an SN evolves over time, classify properties of objects and relations captured in an SN and show how these properties can be maintained. An empirical evaluation, which is based on synthetic and real-world data, investigates the performance, scalability and practicability of the proposed algorithms.

Findings – The authors prove the feasibility of the introduced algorithms in terms of runtime performance with a proof-of-concept implementation. Further, a real-world case from the automotive domain confirms the applicability of the SN maintenance approach.

Originality/value – As opposed to existing work, the presented approach allows for the automated and consistent maintenance of SNs. Furthermore, the applicability of the presented SN maintenance approach is validated in the context of a real-world scenario as well as two business cases.

Keywords Evolution, Maintenance, Knowledge representation, Algorithm, Knowledge-intensive business process, Semantic network

Paper type Research paper

International Journal of Web Information Systems
Vol. 11 No. 3, 2015
pp. 291-326

1744-0084
DOI 10.1108/IJWIS-04-2015-0014

1. Introduction

Knowledge workers and decision-makers are confronted with a continuously increasing load of data they have to cope with during daily work. Examples include e-mails, office files, checklists, guidelines, fact sheets, Web pages and best practices. In daily practice, knowledge workers and decision-makers are not only interested in getting access to these data, but they also require comprehensive and aggregated information when performing specific tasks in a business process context (Michelberger *et al.*, 2012a). Handling such information is by far more complex and time-consuming than just storing data. For example, this information might be incomplete, incorrect, unreliable, unstructured or outdated (Michelberger *et al.*, 2011a; Rowley, 2007).

A particular challenge is to align enterprise information with business processes and to provide relevant information to knowledge workers and decision-makers. In practice, enterprise information is not only stored in distributed and heterogeneous sources but also managed separately from business processes. For example, shared drives, databases, enterprise portals and enterprise information systems are used to store the information. In turn, business processes and their tasks are managed using process-aware information systems (Reichert and Weber, 2012).

In such an environment, information and business processes are often linked manually as well as statically. For example, it has been shown that enterprise portals rather contain complex and static content (e.g. long lists of documents, large process maps). In turn, this rather confuses users (Hipp *et al.*, 2014). Therefore, it is challenging for the latter to identify the relations between enterprise information and business processes, which is crucial when performing specific process tasks.

In previous work, we introduced the semantic network (SN) approach, bridging the gap between enterprise information and business processes (Michelberger *et al.*, 2013). Such an SN can be created using in a bottom-up manner, i.e. starting with the integration of enterprise information and business processes from heterogeneous sources (Michelberger *et al.*, 2012b). Following this, the integrated information and business processes are syntactically and semantically analyzed. The resulting SN then comprises unified information objects (e.g. checklists, guidelines, forms), process objects (e.g. pools, lanes, tasks, gateways, events) and semantic relations (e.g. "is similar to", "is used by"). More specifically, information objects are needed when performing the task of a business process. In turn, process objects correspond to process elements, such as tasks or gateways, that guide process-oriented work. Finally, semantic relations allow identifying inter-linked objects in different ways, e.g. information objects referring to the same topic or objects required for performing a particular process task.

An SN constitutes the basis for the process-centric delivery of relevant enterprise information to knowledge workers and decision-makers (cf. Figure 1). More specifically, an SN offers an application interface that may be queried to retrieve the required information (Hipp *et al.*, 2013). Based on a query (e.g. "information objects relevant for creating a review"), the SN automatically delivers respective information objects to users (Michelberger *et al.*, 2012b).

To provide that process information to process participants fitting best to their current demands, the information and process objects captured in an SN need to be complete, consistent and up-to-date. Consequently, an SN must be continuously maintained. In two case studies as well as an online survey (Hipp *et al.*, 2011; Michelberger *et al.*, 2011b), we have already shown that SN maintenance is a

prerequisite to be able to continuously provide the required information to knowledge workers and decision-makers. SN maintenance, however, is a non-trivial task. For example, objects may be added (e.g. new guidelines are created), updated (e.g. a process task is modified) or deleted (e.g. a checklist is no longer valid). Likewise, relations may be established (e.g. when discovering that two documents have the same author or are stored in the same file format), updated (e.g. two forms become more similar to each other) or deleted (e.g. two documents have no longer the same author). On one hand, such changes may happen outside the SN (e.g. a checklist may have been changed in a database), i.e. the change is *exogenous*. On the other, changes may occur inside the SN (e.g. a lifecycle status change of an object like a guideline becoming outdated). These changes, in turn, are denoted as *endogenous*. Both exogenous and endogenous changes must be properly handled by the SN (cf. Figure 2).

Picking up the aforementioned challenges regarding SN *maintenance*, the contribution of this paper is as follows. First, we propose an approach for SN maintenance. Specifically, we show how SNs evolve over time and identify characteristics of object and relation properties as well as their influence with respect to SN maintenance. Second, we introduce three algorithms dealing with exogenous and endogenous changes of an SN. In this context, we examine the feasibility and costs of the algorithms through a proof-of-concept implementation. Further, we demonstrate, based on an empirical evaluation in the automotive domain, that automated SN maintenance is essential, while being practical at the same time. Note that this paper provides an extended version of the work we presented in Michelberger *et al.* (2014). The additional contributions are as follows:

- The paper illustrates the use of the three algorithms along two real-world use cases. The first one illustrates the monitoring of technologies in the automotive domain, whereas the second one deals with decision-making in the context of technology management.
- The paper comprises a detailed description of the six phases to be passed when creating an SN. In particular, the sixth phase corresponds to SN maintenance.

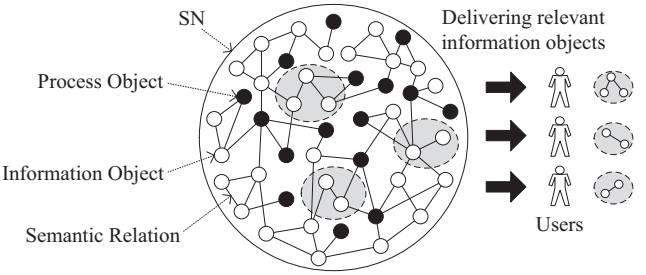


Figure 1.
Delivering relevant information objects

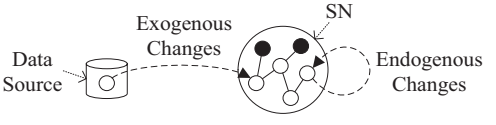


Figure 2.
Exogenous and endogenous changes

- The paper provides a detailed discussion on the advantages and disadvantages of the SN approach. Further, it provides a substantially extended discussion of related work.
- The paper provides deeper insight into the case study.

The remainder of the paper is organized as follows. Section 2 introduces the preliminaries. Section 3 then presents the SN maintenance approach. Section 4 validates the costs of the algorithms and presents a case study demonstrating their applicability. Section 5 deals with the application of the maintenance approach based on two real-world use cases. Finally, Section 6 discusses related work, and Section 7 concludes the paper.

2. Preliminaries

An SN constitutes a labeled and weighted directed graph whose vertices represent objects and the (labeled) edges represent the semantic relations between the objects with weights indicating the relevance of the relations. A weight is expressed in terms of a number ranging between 0 and 1, with 1 indicating the strongest possible semantic relation:

- *Definition 1 (SN)*: An SN is a tuple (V, E, L, W, f_l, f_w), where V is a set of vertices such that each $v \in V$ represents an information object or a process object; E is a multiset of edges such that each edge $e = (v, v') \in E$, $v, v' \in V$ and $v \neq v'$, represents a relation between such objects. The function f_l: $E \rightarrow L$ labels each edge $e \in E$ with an edge label from the set of labels L. Furthermore, the function f_w: $E \rightarrow W$ assigns a weight from the set of weights W to each edge $e \in E$. Given an edge $e = (v, v') \in E$, we call v the source and v' the destination of e.
- *Definition 2 (Neighborhood)*: Given a vertex $v \in V$, the *internal neighborhood* of v, denoted $\Gamma^-(v)$, is the set of vertices $\{v' \mid (v', v) \in E\}$. Analogously, for $v \in V$, the *external neighborhood* of v, denoted $\Gamma^+(v)$, is the set of vertices $\{v' \mid (v, v') \in E\}$. Then, the *total neighborhood* of $v \in V$ is the union of the internal and external neighborhood of v, denoted $\Gamma(v)$.
- *Definition 3 (Degree)*: The incoming degree of a vertex $v \in V$ is the number of incoming edges and the outgoing degree is the number of its outgoing edges. The total degree of v is the sum of its incoming and outgoing degree.

For example, given two edges $e = (v, v')$, $e' = (v', v'') \in E$, $v, v', v'' \in V$, we call v an internal neighbor of v' and v'' an external neighbor of v'. Thus, the total degree of v' is 2.

Note that we often refer to vertices as objects (e.g. information and process objects) and to edges as relations of an SN. We next define properties for vertices and edges:

- *Definition 4 (Properties)*: Each vertex $v \in V$ and each edge $e \in E$ has a set of properties $P(v)$ and $P(e)$, respectively, where each $p \in P(v) \cup P(e)$ is a pair (*key*,*val*). We denote *key* as the unique name and *val* as the value of p and write *key*(v) (*key*(e)) to denote *val*.

To create an SN, business processes and pieces of information, possibly from different data sources (e.g. process repositories, shared drives), are transformed into process and information objects [cf. Figures 3(a and b)], each represented by a vertex and its

according properties. The transformation ensures that proprietary formats (e.g. office formats) are converted into a uniform format, which allows analyzing the SN objects.

After that, SN objects are syntactically and semantically analyzed to detect their semantic relations [cf. Figure 3(c)] (Hipp *et al.*, 2013). First, properties (e.g. authorship) are compared (syntactic analysis), e.g. to link objects with the same author. Second, the properties of the objects are analyzed (semantic analysis). For this purpose, algorithms from the fields of data mining, text mining (e.g. text preprocessing, linguistic preprocessing, clustering, classification, information extraction), pattern-matching and machine learning (e.g. supervised learning, unsupervised learning, reinforcement learning, transduction) are applied (Hotho *et al.*, 2005; Wurzer, 2008) to further classify and group correlated objects.

Semantic relations in an SN exist between information objects (e.g. a guideline similar to another one) or process objects (e.g. an event triggering a sub-process). Additionally, semantic relations exist between information and process objects (e.g. an instruction required for executing a specific process task).

Generally, an SN is created in six consecutive phases (cf. Figure 4) following a bottom-up approach, i.e. we start with the integration of business processes and enterprise information that originating from heterogeneous sources such as databases, shared drives, enterprise portals, process repositories or enterprise information systems (Michelberger *et al.*, 2013).

In Phase 1, business processes relevant for an SN need to be identified and integrated. In this context, relevancy depends on which processes shall be supported by the SN. The business processes to be integrated must be explicitly specified, e.g. using a process modeling language such as BPMN (Freund and Rücker, 2012) or EPC (Scheer, 2002).

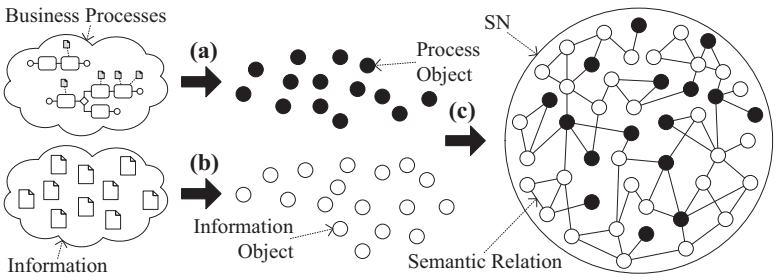


Figure 3.
Schematic creation of an SN

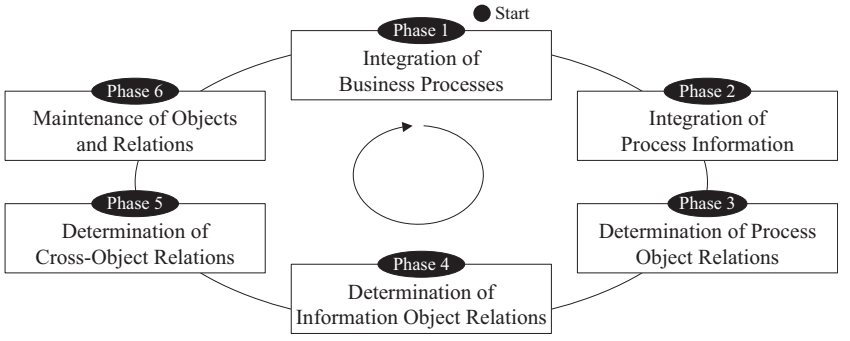


Figure 4.
Construction phases of the SN

Only such an explicit process description allows for the automated transformation of a process schema and its corresponding process instances into process *objects*. For this purpose, all relevant process objects (e.g. tasks, events, gateways, data objects, pools, lanes, sequence flows, message flows or associations) are identified. In turn, the resulting objects are then used to create the SN's first stage of expansion. In Phase 2, relevant process information (e.g. e-mails, office files, manuals, templates, forms, checklists or guidelines) is added to the SN; i.e. the already existing SN is extended by adding information objects of different granularity levels, ranging from fine-grained information (e.g. database tuple) to coarse-grained one (e.g. multi-page office document).

In Phase 3, the relations among the process objects are identified, i.e. process objects such as sequence flows, associations or message flows are transformed into process object relations. In Phase 4, the information object relations between the SN information objects are discovered. Explicit relations, like hyperlinks in documents, are discovered first. Then, algorithms from the fields of data mining, text mining, pattern-matching and machine learning are applied to discover implicit relations as well (Hotho *et al.*, 2005; Wurzer, 2008). In Phase 5, in turn, cross-object relations between information and process objects are identified. For this purpose, similar algorithms as used in Phases 3 and 4 are applied. In addition, pre-defined business rules (e.g. conditional constraints, derivations or process rules) are used to detect further relations (Michelberger *et al.*, 2012b; Wurzer, 2008). Finally, in Phase 6, the SN is *maintained*. This phase deals with the continuous integration as well as the continuous analysis of information and process objects including their corresponding relations. Note that SN maintenance constitutes a prerequisite for providing relevant and up-to-date information to knowledge workers and decision-makers. In the following, we present concepts and algorithms used in the maintenance phase.

3. Maintaining SNs

This section introduces the SN maintenance approach:

- We show how an SN evolves over time (cf. Section 3.1).
- We illustrate why properties of objects and relations are important (cf. Section 3.2).
- We present a collection of algorithms for properly maintaining SNs (cf. Section 3.3).

3.1 Evolution

SN evolution is driven by exogenous as well as endogenous changes. These, in turn, result in changes of objects and relations including their property values. Therefore, we further distinguish between evolution in depth and breadth (Oertelt and Ulmschneider, 2013). Depth is defined by the size of all property values of an SN, i.e. the amount of information (e.g. the information stored with respect to all SN objects). *Breadth*, in turn, is defined as the number of relationships in an SN, i.e. the cardinality of the set of edges.

Depth may be increased by adding objects (e.g. new documents on a shared drive), adding properties (e.g. adding keywords to a object) or updating property values (e.g. describing a property in more detail) [cf. Figure 5(a)]. In turn, deleting objects and properties decreases the depth of an SN [cf. Figure 5(b)]. Note that updates of property values might decrease depth as well. Breadth can be increased by adding relations, e.g. adding a link between two objects [cf. Figure 5(c)]. In contrast, deleting relations (e.g. two

objects no longer have the same author) decreases breadth [cf. Figure 5(d)]. Hence, depth and breadth are indicators for the cost of performing maintenance tasks.

To formalize SN maintenance operations, we need a component which is capable to adapt exogenous and endogenous changes to the SN. We achieve such functionality by the concept of an action:

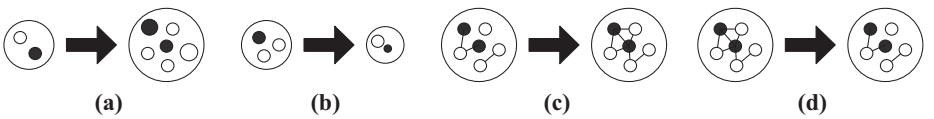
- *Definition 5 (Action)*: An action changes an SN in a structured and standardized way. Each action a has a set of parameters $PA(a)$, where each $pa \in PA(a)$ is a pair (key, val). We call *key* the unique name and *val* the value of *pa* and write $key(a)$ to denote *val*. A parameter *pa*, in turn, is either mandatory or optional. If *pa* with key *key* is mandatory, then, for each action a, there exists a value val_a such that $(key, val_a) \in PA(a)$.

Typical mandatory parameters of an action are a unique identifier *uri* (e.g. a URL) and the function *func* (e.g. add, update, delete) to be executed. Actions are triggered by exogenous or endogenous changes (cf. Figure 6), e.g. when a document on a shared drive is deleted (an *exogenous* change) or becomes outdated (an *endogenous* change). Accordingly, respective events trigger add, update and delete operations on the SN. Therefore, actions adapt internal as well as external events and affect the SN.

As example consider an engineer in the automotive domain conducting a review of product requirements documented in functional specifications. The goal is to improve as well as to approve such specifications. Because of a revision of the review process, an employee from the quality management department replaced an outdated review template. Thus, an action a_1 was triggered with $uri(a_1) =$ "H:/templates/review-v1.xls" and $func(a_1) =$ "delete". Thereby, another action a_2 was triggered with $uri(a_2) =$ "H:/templates/review-v2.xls" and $func(a_2) =$ "add". However, based on new guidelines, the engineer noticed that the template was incomplete (e.g. a required question was missing). Therefore, he adapts it. Thus, another action a_3 is triggered with $uri(a_3) =$ "H:/templates/review-v2.xls" and $func(a_3) =$ "update".

3.2 Property classification

SN maintenance not only requires to consider the object-relation level, but the properties of objects and relations as well. Furthermore, if the author of a document changes; for example, it is not necessary to update the entire object but only relevant parts, i.e. the



Notes: (a) Increase depth; (b) decrease depth; (c) increase breadth; (d) decrease breadth

Figure 5.
SN evolution in depth and breadth

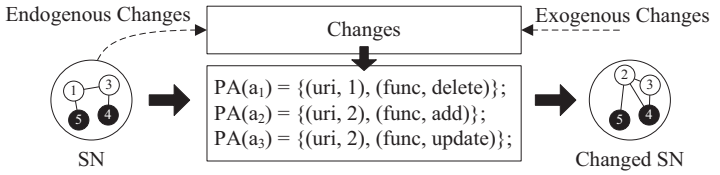


Figure 6.
Actions changing an SN

value of property author and according relations. One can observe that certain properties change over time (e.g. file size), whereas others do not (e.g. *uri*). This can be exploited in maintenance algorithms by focusing on those properties being relevant for a particular operation. To capture this, we categorize properties as follows:

- *Definition 6 (existence and mutability)*: Properties are classified into two categories: *existence* and *mutability*. *Existence* expresses whether a property is *mandatory* or *optional*, where property p with key *key* is mandatory for vertices V (edges E) of an SN, if for each $v \in V$ ($e \in E$), there exists a value val_v (val_e) such that $(key, val_v) \in P(v)$ $((key, val_e) \in P(e))$; otherwise, it is optional. *Mutability,* in turn, expresses whether a property's value is *dynamic* or *static*, where p is *dynamic*, if *val* in (*key*, *val*) can change over time, and it is *static* otherwise.

For example, for $v \in V$, typical mandatory properties are a unique identifier *uri*, a data source *source*, a creation date *cdate*, a modification date *mdate* and a content *cont*. The categories, existence and mutability, can be combined into a matrix comprising four blocks to which we assign the properties (cf. Figure 7).

In the following, we illustrate the assignment of individual properties to different blocks with examples:

- *Mandatory/dynamic*: Some properties are always part of an SN and are dynamic. For example, the modification date *mdate* changes with every update of an object or relation. The content *cont* or the total degree *deg* of an object can change over time as well. Therefore, these properties are mandatory and dynamic.
- *Optional/dynamic*: The *title* of a document can change over time, but some file types (e.g. a text file) do not have a *title*. Therefore, the *title* of a document can be considered as optional and dynamic. A property containing project budgets *invest* might not be available for all vertices or edges and can vary over time as well. Therefore, *invest* can be considered as optional/dynamic as well.
- *Mandatory/static*: An identifier *uri*, a *creator* or a creation date *cdate* exists for all objects and relations and, therefore, is mandatory. As these properties do not change over time, they can be considered as static.
- *Optional/static*: If a property does not change over time and does not exist for every object or relation, it is called optional/static, e.g. the file type of a document.

Based on the property classification, we infer the following for adding, updating and deleting elements of an SN: one must ensure that static/mandatory properties (c) are given as a minimum requirement when adding elements [cf. Figure 8(a)]. When deleting

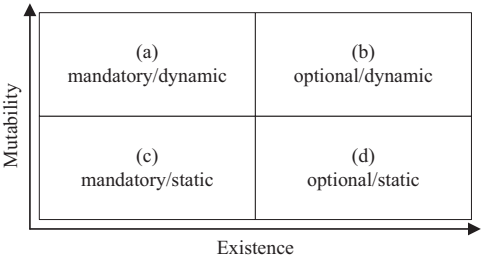


Figure 7.
Property
classification

elements, one has to consider properties within all blocks (a, b, c, d) [cf. Figure 8(b)]. When executing updates, only properties not assigned to the mandatory/static block (a, b, d) must be considered [cf. Figure 8(c)]. Note that the gray background color in Figure 8 indicates affected blocks for each function (cf. Section 3.3).

Therefore, from the evolutionary perspective (cf. Section 3.1), dynamic properties capture changes of existing vertices and edges in an SN, whereas static properties cover changes in terms of added vertices and edges. Additionally, mandatory properties always allow for analyses, like comparisons on the overall set of SN vertices and SN edges. Usually, dynamic properties store process information, whereas static properties usually store metadata.

3.3 Algorithms

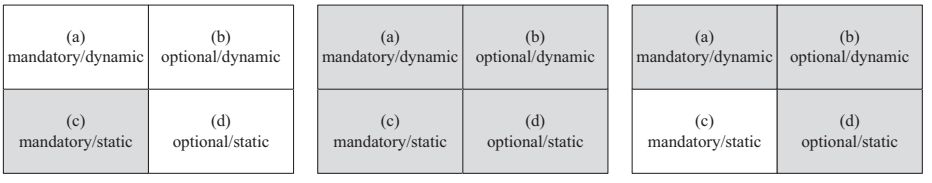
To successfully maintain SNs, we first specify functions (*add*, *delete* and *update*) that can be triggered by actions to perform maintenance operations with the algorithms.

The *add*-function adds a vertex v_{add} and its properties to an *SN* and determines which semantic relations exist between v_{add} and existing vertices. As mentioned above, mandatory/static properties are the minimum input parameter for the *add*-function:

Function add (SN,v_{add})
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN, v_{add} the vertex to be added incl. its properties $P(v_{add})$;
Ensure: *SN* is updated;
$V: = V \cup \{v_{add}\}$;
for each $v \in V$ do
  if $uri(v) \neq uri(v_{add})$ then
    $E: = E \cup$ {new edge/s between v and v_{add}};
  end if
End for

The *delete*-function deletes a vertex v_{del} in an *SN* including existing semantic relations between v_{del} and its total neighborhood. Note that the function considers all blocks of the property classification, i.e. all properties of v_{del} are deleted.

Function delete(SN, v_{del})
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN,
v_{del} the vertex to be deleted incl. its properties $P(v_{del})$;
Ensure: *SN* is updated;



Notes: (a) Add-function; (b) delete-function; (c) update-function

Figure 8. Property classification and functions

v: = get $v \in V$ s.t. $uri(v) = uri(v_{del})$
E: = $E \backslash \{(v, \gamma), (\gamma, v) \mid \gamma \in \Gamma(v)\}$
V: = $V \backslash \{v\}$

The *update*-function takes a vertex v_{upd} as input, which is used to update the vertex v in the SN that is identified by the same uri as v_{upd}. The function also adds, deletes and updates semantic relations between the updated vertex v and existing vertices. Note that mandatory/static properties are not considered in case of objects:

Function update(SN, v_{upd})
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN, v_{upd} the vertex used to update the SN incl. its properties $P(v_{upd})$;
Ensure: SN is updated;
$P(v_{upd})$: = $\{p \in P(v_{upd}) \mid p$ is not mandatory/static$\}$;
v: = get $v \in V$ s.t. $uri(v) = uri(v_{upd})$;
for each (key, val) $\in P(v)$ do
  if (key, val_{upd}) $\in P(v_{upd})$ then
    val: = val_{upd};
    $P(v_{upd})$: = $P(v_{upd}) \backslash \{(key, val_{upd})\}$;
  else
    $P(v)$: = $P(v) \backslash \{(key, val)\}$;
  end if
end for
$P(v)$: = $P(v) \cup P(v_{upd})$;
for each $v' \in V$ do
  if $v' \in \lceil (v)$ then
    E: = update edge/s betw. v' and v;
    E: = $E \backslash \{$obsolete edge/s between v' and $v\}$;
  end if
  if $uri(v') \neq uri(v)$ then
    E: = $E \cup \{$new edge/s between v' and $v\}$;
  end if
end for

Based on these functions, we propose three algorithms for maintaining SNs. The maintenance is based on two main principles: the *push*- and the *pull-principle*. The former can be applied to both exogenous and endogenous changes, whereas the latter can only be applied to exogenous changes.

With the push-principle, the data source pushes information and business processes automatically to an SN when they are added, updated or deleted within the data source. However, with regard to exogenous changes, prerequisite is that the data source is able to send notifications if information and/or business processes have been changed. Regarding endogenous changes, the prerequisite is that the SN detects changes automatically and triggers respective actions.

With the pull-principle, an SN gathers information and business processes from a data source. Such a maintenance process is triggered by time-based schedulers, i.e. the SN is maintained at a certain point in time. The principle is used for data sources which are not capable of sending change notifications (e.g. a document has changed) to the SN.

Altogether, the use of a specific principle depends on the capabilities of a data source (whether the data source is able to send change notifications to the SN or not). For

example, for an enterprise information system which is capable of sending notifications, we use the push-principle, whereas for a shared drive, we use the pull-principle.

For each of these two principles, we introduce a corresponding algorithm (cf. Figure 9). Prerequisite for both principles is that the SN has access to underlying data sources. In case of exogenous changes, the SN transforms information and business processes into a uniform format. In case of endogenous changes, no transformation is necessary.

3.3.1 Push-algorithm. The push-algorithm deals with changes of an SN based on the push-principle, e.g. a policy is no longer valid in 2015 and the corresponding object in the SN has to be maintained accordingly. Thus, the maintenance of the SN is triggered by an action that is applied to the SN by the push-algorithm.

The algorithm works as follows: In the add and update case, we create a vertex v including its properties from the data source affected by the action a (i.e. based on the *uri* of the action). After that, we call the corresponding function. In the delete case, we identify the corresponding vertex $v \in V$ based on the *uri* of the action and call the according function. Hence, the push-algorithm applies endogenous and exogenous changes to the SN by actions.

Algorithm 1: Push-algorithm
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN, a an action;
Ensure: SN is updated;
switch *func(a)* do
  case *add*
    v: = create a vertex incl. its properties from the data source affected by the *uri* of a;
    add(SN, v);
    break;
  end case
  case *update*
    v: = create a vertex incl. its properties from the data source affected by the *uri* of a;
    update(SN, v);
    break;
  end case
  case *delete*
    v: = get $v \in V$ s.t. *uri*(v) = *uri*(a);
    delete(SN, v);
  end case
end switch

3.3.2 Pull-algorithm. The pull-algorithm deals with changes of an SN based on the pull-principle, i.e. data have changed in the data source and need to be gathered by the

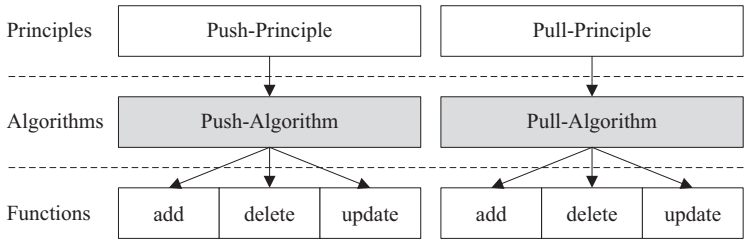


Figure 9.
Push-and pull-algorithm

SN. For example, documents on a shared drive are updated, and therefore, respective changes have to be made in the SN. As aforementioned, the maintenance of the SN is triggered by a scheduler. The algorithm works as follows: First, we create a set of vertices V_{ds} from a data source *ds*. After that, for each vertex $v \in V$ in the SN that was created from *ds* (property *source(v)*), we check whether a corresponding vertex $v_{ds} \in V_{ds}$ exists. If this is the case, we check whether v_{ds} is newer than v (e.g. by comparing the creation and/or modification dates). If v is out-of-date, it is updated with the properties of v_{ds} by calling *update*(SN, v_{ds}). After that, v_{ds} is removed from V_{ds}. If no corresponding vertex exists in the data source, we delete the vertex (v) in the SN using the *delete* (SN, v) function. Finally, we add each remaining vertex $v_{ds} \in V_{ds}$ from the data source to the SN by calling *add*(SN, v_{ds}), which leaves the SN synchronized with the data source. Hence, the pull-algorithm allows for maintaining the SN at a certain point in time. Note that the process has to be repeated for each data source which must be synchronized:

Algorithm 2: Pull-algorithm
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN, *ds* the data source;
Ensure: SN is updated;
V_{ds}: = create a set of vertices incl. their properties from the data source *ds*;
for each $v \in V$ s.t. *source*(v) = *ds* do
  v_{ds}: = get $v_{ds} \in V_{ds}$ s.t. *uri*(v_{ds}) = *uri*(v);
  if v_{ds} then
    if v_{ds} is newer than v then
      update(SN, v_{ds});
    end if
    V_{ds}: = $V_{ds} \backslash \{v_{ds}\}$;
  else
    delete(SN, v);
end for
for each $v_{ds} \in V_{ds}$
  add(SN, v_{ds});
end for

3.3.3 Partial-pull-principle and algorithm. In practice, SNs can comprise a large amount of objects and relations. Maintaining SNs using the pull-principle can, therefore, be a very time-consuming task. In a specific work context, a user might, however, only be interested in a selected part of the SN. For example, during a review, review templates, review minutes, existing reviews or even results of a real-time evaluation (e.g. prioritization of projects in a workshop) are of great importance, whereas checklists and best practices for performing effective project management are less interesting. Thus, it is sufficient to maintain only these objects and relations that are relevant to the user when querying the SN. To capture this, we introduce a further principle, called the partial-pull-principle, where the SN gathers only information and business processes from data sources as requested by a user. These (and only these objects) are then updated on demand.

In contrast to the other principles, the partial-pull-principle is completely *user-driven* because it is triggered by a user request, whereas the push- and pull-principle are machine-driven, e.g. through notifications from other enterprise information systems or schedulers.

Based on the partial-pull-principle, we introduce a third algorithm (cf. Figure 10) as a lightweight version of the pull-algorithm. It does not maintain the entire SN, but only the parts which are relevant for a given request.

The algorithm works as follows: First, we create a set of vertices V_{ds} from affected data sources according to the user request *req*. Then, for each vertex $v \in V$ affected by *req*, we retrieve the corresponding vertex v_{ds} from the affected data sources. Thus, if a corresponding vertex v_{ds} is in the data source, we check whether v_{ds} is newer than v (e.g. by comparing the creation and/or modification dates), and if v is out-of-date, it is updated with v_{ds} by calling *update*(SN, v_{ds}). If there is no corresponding vertex in the data source, we delete the vertex (v) in the SN using the *delete*(SN, v) function. Hence, the partial-pull-algorithm allows for maintaining parts of an SN based on a user request and ensures that all requested objects including their properties are synchronized with affected data sources:

Algorithm 3: Partial-pull-Algorithm
Require: $SN = (V, E, L, W, f_l, f_w)$ an SN, *req* the request to an SN;
Ensure: SN is partially updated;
V_{req} contains the requested vertices;
V_{ds}: = create a set of vertices from the data sources affected by *req*;
for each $v \in V$ *affected by req do*
  v_{ds}: = get $v_{ds} \in V_{ds}$ s.t. $uri(v_{ds}) = uri(v)$;
  if v_{ds} then
    if v_{ds} *is newer than* v then
      update(SN, v_{ds});
      v_{upd}: = get $v' \in V$ s.t. $uri(v') = uri(v_{ds})$;
      V_{req}: = $V_{req} \cup \{v_{upd}\}$;
    else
      V_{req}: = $V_{req} \cup \{v\}$;
    end if
  else
    delete(SN, v);
  end if
end for

4. Validation

This section shows that the algorithms are able to successfully maintain an SN. For this validation, we implemented the algorithms and evaluated their performance considering depth and breadth, i.e. we measured the time needed to add, update and delete objects as well as relations. We further evaluated the relevance as well as the

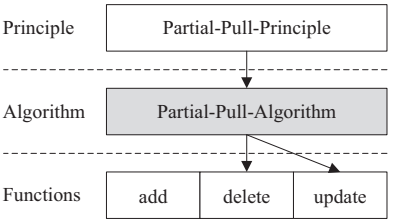


Figure 10.
Partial-pull-algorithm

applicability of the algorithms in the context of a case study which we conducted in the automotive domain.

The validation was guided by three research questions:

RQ1. Is automatic maintenance of an SN feasible considering both exogenous and endogenous changes?

RQ2. How do depth and breadth affect the runtime of the SN maintenance algorithms?

RQ3. How essential is automatic maintenance of SNs in practice?

4.1 Implementation and configuration

Driven by the research questions presented we created the following:

- a well-defined set of SNs for evaluating the performance with synthetic data (cf. Figure 12) to answer *RQ1* and *RQ2*; and
- a specific SN for empirical evaluating the algorithms with real business data to answer *RQ3*.

Note that the objects and relations shared identical properties to enable comparability regarding measurements as well as enable the evaluation of the used configuration in a real-world application.

We realized the prototype as a three-tier architecture. The presentation layer was implemented with the Web application framework Play[1], the Twitter Bootstrap framework[2], the Data-Driven Documents (D3) library[3], jQuery[4], HTML5 templates and Cascading Style Sheets (CSS3). We created the SNs using the semantic middleware iQser GIN Server (v. 2.0.0.36) (Wurzer, 2008). In addition, we developed Java open-source plugins[5] on the logic layer. The data layer was based on a Lucene search index[6] and a MySQL database[7].

Based on the property classification described in Section 3.2, we configured objects and relations of the prototype. As mandatory/static object properties, we chose *cdate*, *uri* and *source* [cf. Figure 11(a)]. Optional properties were *file type* and *title*: the *file type* does not change over time (static), whereas the *title* may evolve (dynamic). Furthermore, the

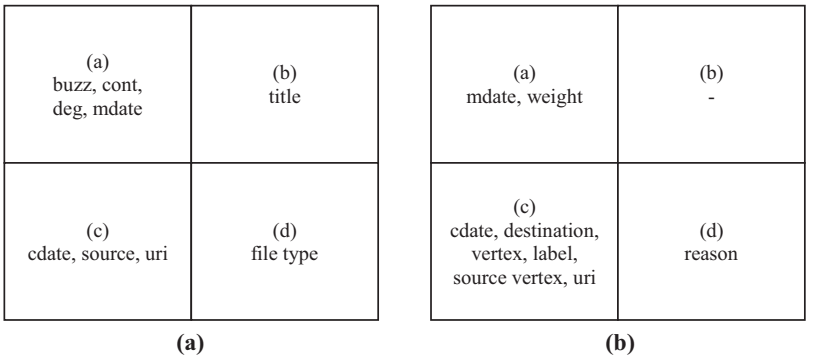


Figure 11.
Property classification implementation

Notes: (a) Objects; (b) relations

title might not be available for every *file type* (e.g. text file), i.e. it is optional. Mandatory dynamic properties included *buzz* (i.e. user activity on objects), *cont*, *deg* and *mdate*.

Analogous to object properties, *uri* and *cdate* were mandatory for relations [cf. Figure 11(b)]. However, relations had additional mandatory properties such as source vertex and destination vertex. The label of an edge was also mandatory and static. However, the weight of an assigned label may vary over time, and therefore, it was configured as mandatory and dynamic. As example, changing *cont* may affect the weight of "is similar to" relations. Additionally, the reason of a relation, which describes why a relation was established (e.g. a particular method that detected an "is similar to" relation), was classified as static and optional. All SNs were created with the following relations: "is author of", "has same title as" and "is similar to".

After setting up the prototype, we first validated the performance of the algorithms (cf. Section 4.2), i.e. the influence of depth and breadth on the algorithms. The performance tests were executed on a machine with an Intel quad-core CPU Intel Core i7 2670Q with 3.1 GHz, 16 GB RAM, 512 GB solid-state drive (SATA 6 Gbit/s) and a Windows 7 64-bit operating system. Then, we evaluated the application of the algorithms in the context of a case study in the automotive domain (cf. Section 4.3).

4.2 Technical validation

The successful implementation and initial tests have already demonstrated that automatic SN maintenance is feasible in general. We now examine the runtime in consideration of depth and breadth. To address research questions *RQ1* and *RQ2*, we investigated the performance of add, update and delete operations for the pull- as well as the push-algorithm (cf. Section 3.3).

Following this, we created SNs comprising 5, 50 and 500 objects, once with small (1 KB) and once with larger (100 KB) text files (cf. Figure 12). To obtain comparable results for the measurements, all objects within an SN were identical (e.g. identical property *cont*). Based on this, we simulated the worst-case scenario with respect to performance: every object being connected with every other object in each SN yielding 40, 4,900 and 4,99,000 relations. Note that only "is similar to" and "is author of" relations were detected, as the property title is not explicit in plain text files, and therefore, no "has same title as" relations were recognized.

Based on the initial SNs, we performed operations (add, update, delete) that refer to a single object using the push- and pull-algorithm (cf. Section 3.3). Each combination of SN, algorithm and operation represented one case to be examined with regard to small and large files, which resulted in 36 cases. An exemplary case may be updating an object

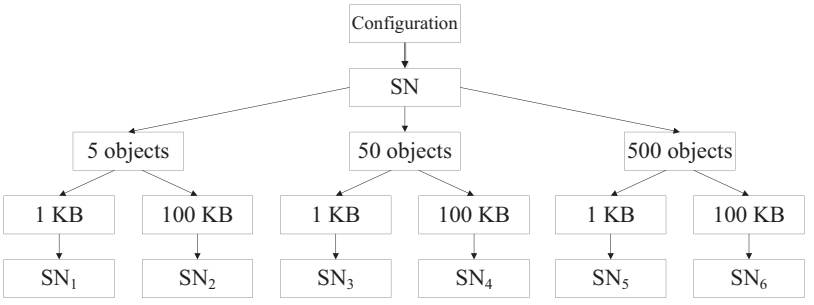


Figure 12.
SN instances for technical validation

with a size of 1 KB using the push-algorithm in an SN with five objects. For each case, the numbers shown in the diagrams (cf. Figures 13-15) correspond to averages over three warm runs. Warm runs were chosen to ensure comparability of the measured values because the iQser GIN Server (Wurzer, 2008) performs several initial background tasks on start-up. Note the logarithmic scale used in the diagrams.

Figure 13 shows that objects can be added to an SN in linear time. Further, depth has an influence on the runtime regarding the number of objects in an SN as well as their size (1 vs 100 KB). Therefore, the actual performance of the algorithms depends on the property values of the vertices. For example, the value of *cont* (i.e. the size of the property in bytes) affects the analysis of similarity relations between the added and the existing vertices. Detecting relations between the added and the existing objects is polynomial in the number of vertices. Note that complexity rises when additional relation labels (m = number of relation types) must be detected between objects (n = number of objects) because, usually, additional algorithms must be executed, which examine properties of vertices concerning a specific relation type. Therefore, the complexity level can reach nm/^2, considering that each relation label is derived by a particular algorithm whose complexity influences the overall algorithmic complexity as well. For example, if such an algorithm is exponential, the overall complexity will no longer be polynomial. However, some algorithms can be used to process multiple relation labels. As example, consider deriving foreign key relationships from a relational database (e.g. "is author of" or "is used by").

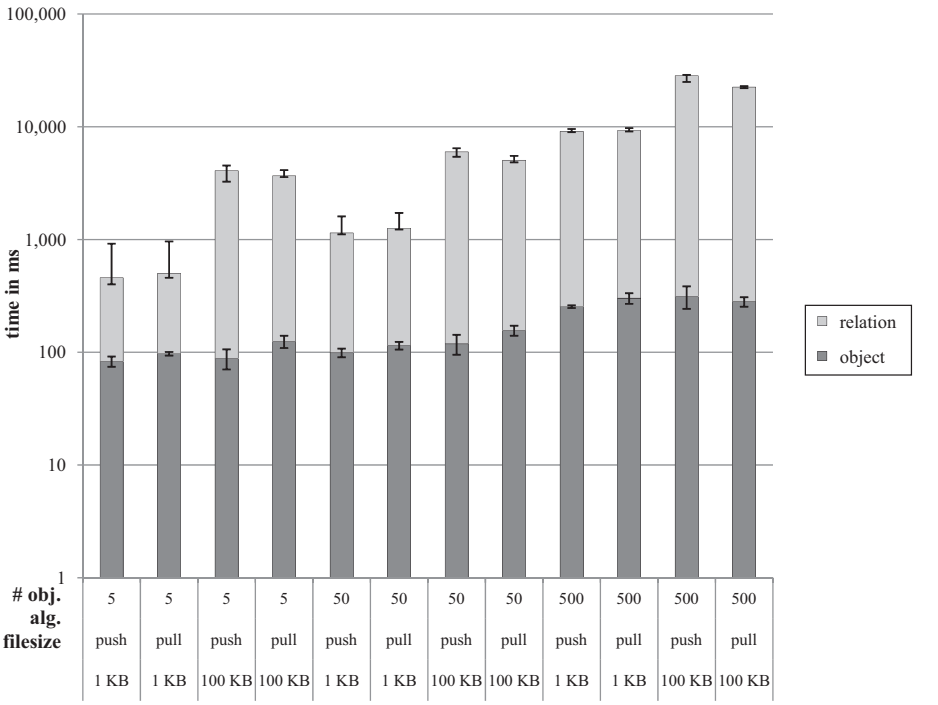


Figure 13.
Effect of depth and breadth on additions

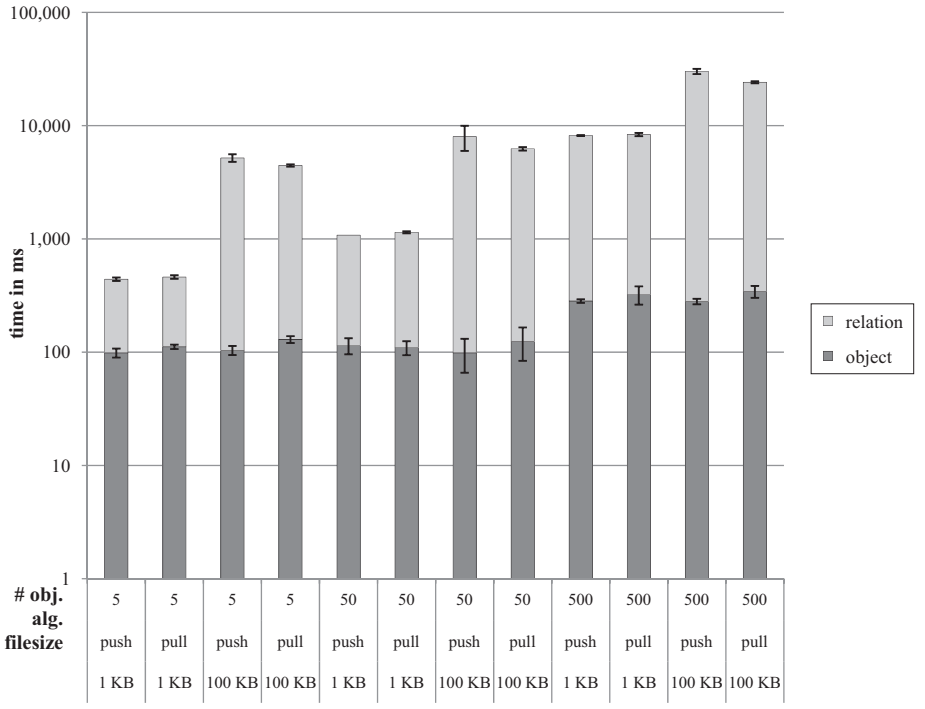


Figure 14.
Effect of depth and breadth on updates

As shown in Figure 14, update operations perform similarly compound to add operations. As opposed to the integration of objects, however, certain operations are not required (e.g. for mandatory/static properties). Comparisons between existing properties need to be performed instead, e.g. to check whether a property has changed. However, we could not detect significant differences concerning costs between add and update operations when applying them under identical initial situations.

As opposed to add and update operations, delete operations perform differently. While the objects can be deleted in linear time, the deletion of relations varies significantly depending on the size of the objects. The reason is that all references, which form the basis of a relation (e.g. extracted concepts and co-occurrences of a specific object in case of "is similar to" relations) must be deleted as well. Note that such "housekeeping" tasks were controlled by the iQser GIN Server. In turn, this might be the reason for variations of the measured values. For example, the cost for deleting an object with a size of 1 KB out of five objects was higher than the cost for deleting an object with a size of 100 KB out of five objects (cf. Figure 15). Furthermore, measurements with the programming language Java can be less accurate compared to native programming languages (e.g. the Java garbage collector cannot be disabled).

Despite the fact that the implemented technology might cause inaccuracies in measurements, we were still able to verify that the maintenance costs caused by the algorithms highly depend on depth and breadth (*RQ2*). However, external components (e.g. a component for the semantic analysis of object properties) as well as their computation cost can influence the performance of maintenance operations

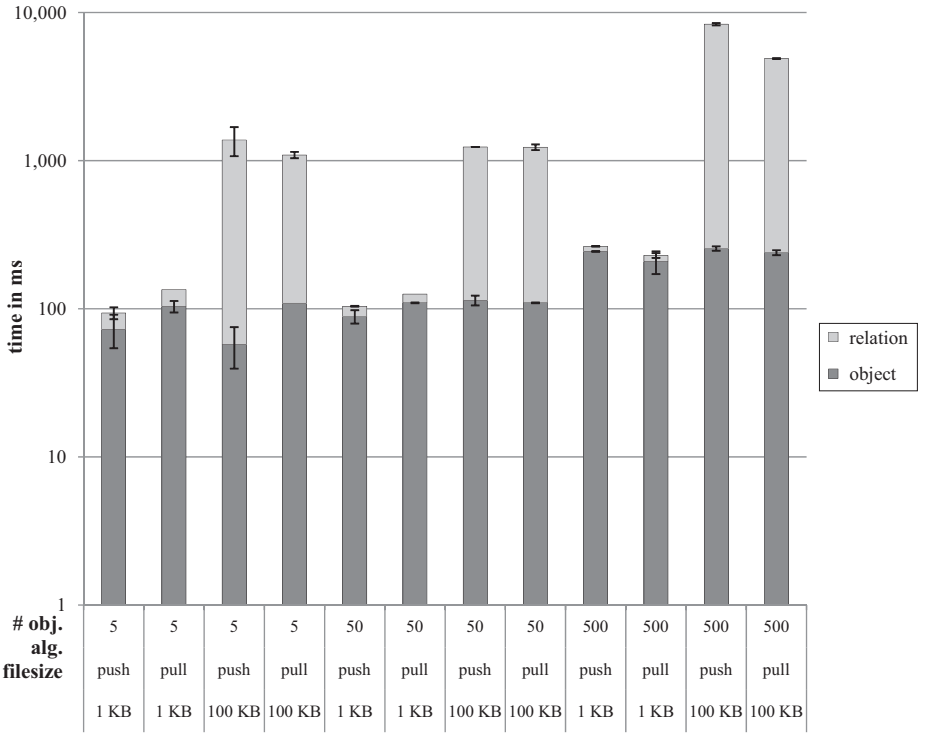


Figure 15.
Effect of depth and breadth on deletes

significantly. In turn, the consideration of the presented property classification (cf. Section 3.2) can have positive effects on performance if only required operations (e.g. only updating properties which are not mandatory/static) are executed.

Altogether, the push- and the pull-algorithm both ensure a synchronized SN regarding affected data sources. Thereby, the technical validation has shown that automatic SN maintenance is feasible (*RQ1*). We further showed that the runtime of the maintenance algorithms is influenced by depth and breadth (*RQ2*). In the following, we evaluate the relevance as well as applicability of automatic SN maintenance in the context of a case study we conducted in the automotive domain.

4.3 Empirical validation

After completing the successful technical validation, we have to confirm the relevance and applicability of the algorithms in a real-world use case (*RQ3*). Following the proposal of Yin (2009) and Kitchenham *et al.* (1995), we chose an empirical case study to evaluate *RQ3* in a characteristic project setting. Data were collected by semi-structured interviews, which allowed us to ask open as well as closed questions (cf. Figure 16). The interviews were based on a questionnaire comprising 60 questions[8]. The layout of each interview was designed applying the time-glass principle (Runeson and Höst, 2009). We started with closed-ended questions allowing for comments from the participants in the first part where we asked general questions about their current work environment and information handling in the context of their processes. Afterward, they

had to perform tasks with the described prototype (cf. Section 4.1) and answer mostly closed-ended questions bound to these tasks with respect to maintenance (i.e. add objects, update objects, delete objects and search for objects and validate property values as well as detected relations). Finally, we asked additional, mostly open-ended questions. This approach allowed us to develop and guide the interview conversation to gain deeper insights through exploration as well as the collection of structured data based on closed questions. Particularly, in the last part of the interviews, the open questions allowed reflecting the tasks with the SN together with the participants. In addition, the approach allowed to further understand the need as well as the requirements for SN maintenance and offered the basis to discuss the benefit of further application scenarios.

Therefore, *RQ1* could be addressed from the user's perspective by the task-specific questions, whereas *RQ3* was investigated by introducing and final questions. The participants were selected based on their expert knowledge regarding the considered case. Two basic roles were involved: knowledge workers and decision-makers from several innovation departments in the automotive domain. Thus, all participants were involved in knowledge-intensive business processes (Mundbrod and Reichert, 2014). No participant was a member of the research team.

The case study was performed in July 2014 with five decision-makers and six knowledge workers from a large automotive manufacturer. Each interview lasted around 90 minutes. For the tasks, we provided an SN with identical configuration regarding the technical evaluation (cf. Section 4.2). However, the underlying corpus (data sources) contained 333 real-world documents from their field of interest (e.g. scientific papers from departments dealing with technology monitoring and technology development). Therefore, the users were familiar with the information represented in the SN and able to judge the containing information and its quality on a certain level.

The initial questions about information handling in the users' current work environment revealed that, except personal contacts (e.g. in meetings or phone calls), information is mostly handled and accessed in a digital way (cf. Figures 17 and 18).

However, information is mostly not well-structured and often distributed across different sources (e.g. information systems or shared drives), which makes it difficult to search for it. These statements are endorsed by the fact that a significant amount of information is stored in files (cf. Figure 19). Remarkable in this context is the usage of visual information. Almost 50 per cent of the participants work with visual information. Figures, which might be of interest for decision-makers and knowledge workers, are often embedded in files (e.g. charts in spreadsheets, models in presentations or technical drawings in patents) or information systems (e.g. reports or dashboards). The participants stated that such drawings usually had additional business value.

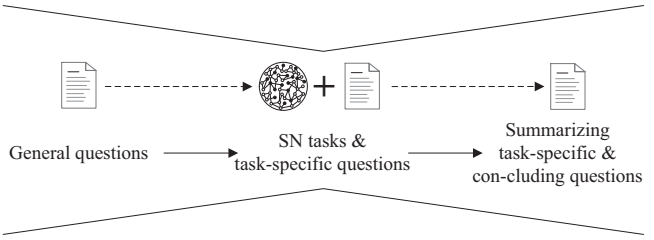


Figure 16.
Interview design

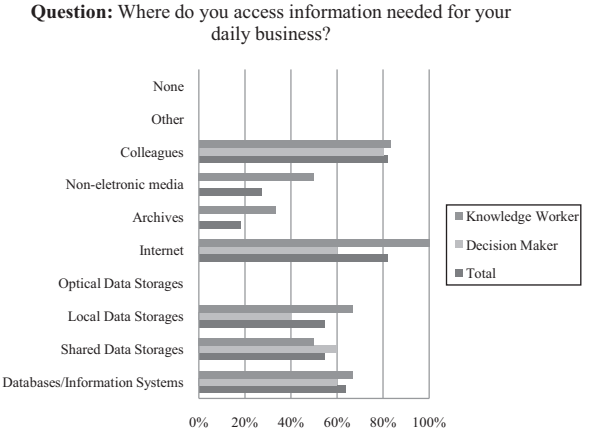


Figure 17.
Information access by source

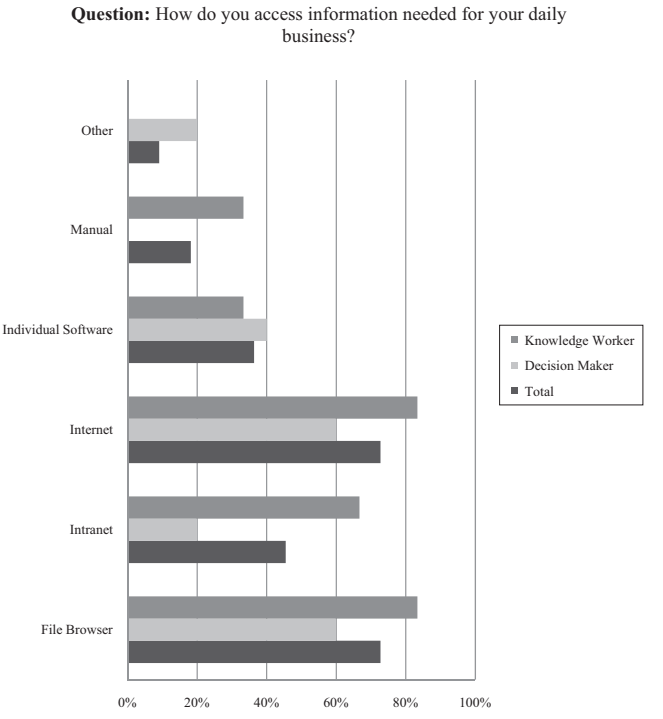


Figure 18.
Information access by software

Considering the enormous file usage, a resulting challenge when working with files in a business environment is to keep track of changes (cf. Figure 20) and to identify interdependencies such as identical or similar documents within different data sources (cf. Figure 21). This is mainly substantiated by the dynamics and the amount of available information as well as a lack of technological assistance (e.g. search over

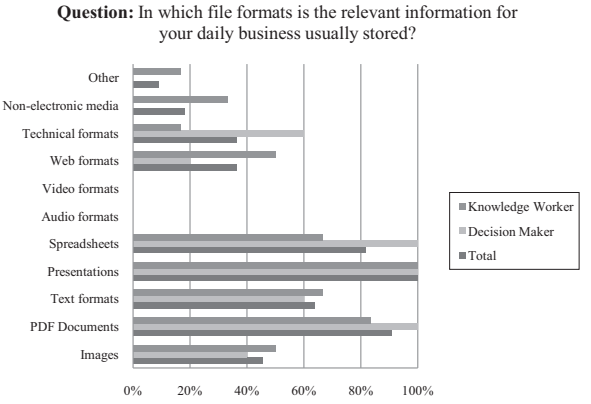
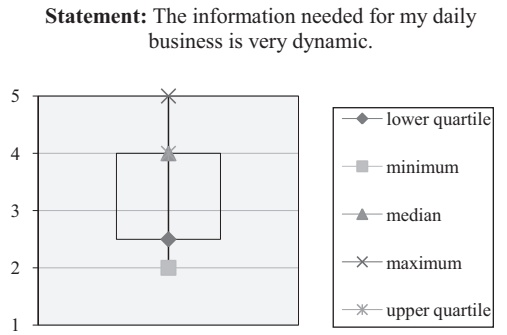
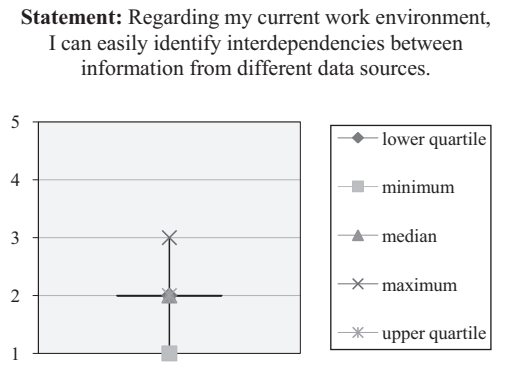


Figure 19.
Information access by file format



Notes: 1: Completely disagree; 2: disagree; 3: neutral; 4: agree; 5: completely agree

Figure 20.
Information dynamics



Notes: 1: Completely disagree; 2: disagree; 3: neutral; 4: agree; 5: completely agree

Figure 21.
Information overview

different data sources, notifications on changed content). Therefore, to be up-to-date, an SN must cover such changes.

To address these challenges (e.g. dynamics of information in distributed sources, heterogeneous file types), we introduced the implemented prototype and asked the participants to perform tasks with the provided graphical user interface (e.g. to check whether property values were updated or to validate the correctness of a relation).

Regarding the concluding questions, every participant stated that all SN maintenance tasks with the prototype could be completed successfully. As objects, relations and properties were adapted based on exogenous and endogenous changes, we can confirm that automatic SN maintenance is feasible in practice.

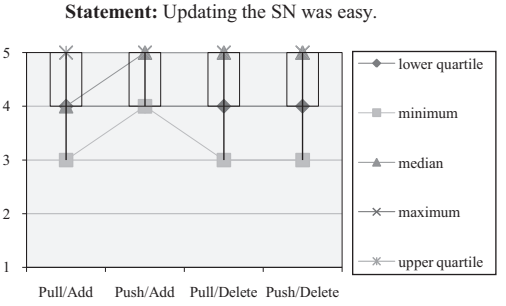
However, certain users could not recognize any relation between the document they added to the SN and others. While no "is author of" and "has same title as" relation existed, the "is similar to" relations were not shown. To enable a more sophisticated user experience, we filtered these relations according to a hard-coded weight-threshold with respect to the overall amount of "is similar to" relations. Nevertheless, with corresponding database entries, we could verify that the relations were set, and took this as an input for further user interface development by allowing users to set the threshold dynamically (i.e. filtering relations by weight).

Additionally, we asked the users about their preferences for push or pull. All users preferred the push-algorithm, as the effects of their tasks were reflected immediately in the user interface. In contrast, the pull-algorithm always has a delay, as it is only triggered at a certain point in time. Obviously, a synchronization every 3 minutes caused too much delay for some users, even when performing other tasks in between. However, the partial-pull-algorithm, which addresses this downside, received positive feedback from the participants. Note that the partial-pull algorithm only considers objects currently existing in an SN.

We interpret the disadvantage of the pull-algorithm as a confirmation of *RQ3* (automatic maintenance is essential) and recommend the push-algorithm if technology permits. Regarding the pull-algorithm, the update intervals can be shortened, depending on the number of files in a data source. Nevertheless, the delay of the pull-algorithm, as observed by the users, is caused by the configured time interval of implemented schedulers. However, its performance compared to the push-algorithm is almost identical (cf. Section 4.2).

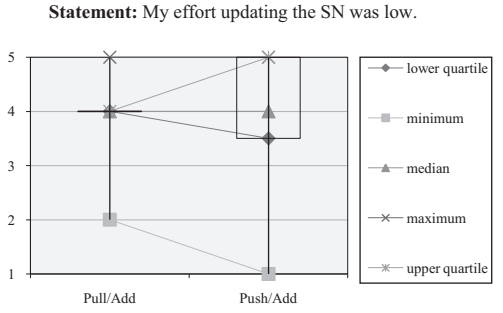
Finally, we asked the users about their impression of benefits for their daily work with respect to SN maintenance. All users stated that integrating and connecting distributed information was easy when using the prototype (cf. Figure 22) and could be done with reasonable effort (cf. Figure 23). Additionally, they confirmed the benefits resulting for their daily work (cf. Figure 24).

More than 90 per cent of the participants stated that the SN provided an enhanced overview on information and that a maintained SN was desirable. In particular, this would be a benefit compared to distributed heterogeneous data sources. One user summarized: "Maintenance and corresponding updates should be automatic. No user interaction for maintenance should be required. Users should focus on working with the SN". Asking about use cases for a maintained SN, knowledge workers as well as decision-makers recognized the potential of the SN to support their daily work.



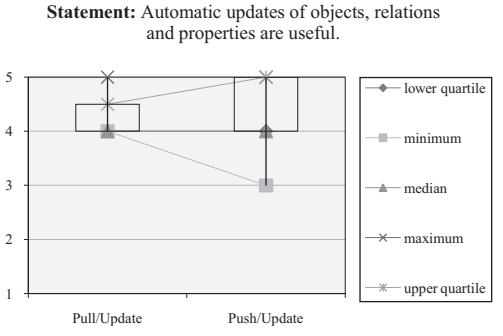
Notes: 1: Completely disagree; 2: disagree; 3: neutral; 4: agree; 5: completely agree

Figure 22. Simplicity of SN maintenance for users



Notes: 1: Completely disagree; 2: disagree; 3: neutral; 4: agree; 5: completely agree

Figure 23. SN maintenance effort for users



Notes: 1: Completely disagree; 2: disagree; 3: neutral; 4: agree; 5: completely agree

Figure 24. User perspective on automatic SN maintenance

4.4 Conclusion

We have shown that automatic SN maintenance with regard to both exogenous and endogenous changes is feasible with acceptable cost (*RQ1*). Furthermore, we examined the effect of depth and breadth on the runtime of the proposed algorithms (*RQ2*). Both algorithms performed satisfactorily when adding, updating and deleting objects. The cost of detecting relations, however, varies significantly with the algorithms used for this task (e.g. expensive linguistic or statistical algorithms).

In the case study, all users appreciated a unified single point of information access, which is up-to-date and allows for searching and filtering distributed information. Many of the interviewed users already experienced a well-maintained SN as an enabler for various use cases (e.g. expert search or gap analysis). In particular, knowledge workers and decision-makers can benefit from maintained SNs. Both groups emphasized that relations between objects could be interesting for various purposes (e.g. navigation within an SN or decision support). For example, SN visualizations can be used to support decisions (cf. Section 5.2).

The case study results on the deficits of information handling in current business environments (cf. Section 4.3) have shown that it becomes increasingly crucial to provide up-to-date, integrated and homogenous views on enterprise information. Moreover, the empirical validation confirms that there is a high demand for a central point of information access in knowledge-intensive processes. Finally, we verified that a maintained SN is essential not only for such business processes but also for a practicable solution (*RQ3*).

5. Use cases

Complementary to the validation of the algorithms in Section 4, we now demonstrate the relevance of SN maintenance in practical settings by considering two use cases from the technology process in the automotive domain.

Radical innovations require high investments and conscientious preparation (Oertelt and Ulmschneider, 2013). The technology process of an enterprise (cf. Figure 25) constitutes the basis for developing and adopting novel technologies, products and processes in a structured way (Oertelt, 2009) to create value for customers with changing demands (e.g. technical and functional innovations, environmental awareness) in a dynamic environment (e.g. volatile raw material prices, legal regulations, competitors or start-ups). The technology process involves tasks like the identification, monitoring, prioritization and controlling of technologies (Oertelt and Ulmschneider, 2013). Typical tasks, for example, include the observation of relevant technologies, the identification of technology enablers and the evaluation of a technology's maturity to determine future application scenarios and evaluate them with regard to costs, potentials and risks. The output of such tasks must be well-grounded, as corresponding decisions might have significant future impact in terms of technological and economic success as well as a company's reputation. The process requires sustainable research activities, interaction and expertise and, therefore, can be characterized as knowledge-intensive.

In consequence, identifying and utilizing relevant information to support the technology process constitutes a challenging task. Usually, the required information cannot be found in one data source and is stored in unstructured or semi-structured data formats. Taking technology monitoring and decision support as examples, we illustrate how a maintained SN could reduce efforts and enhance the technology process.

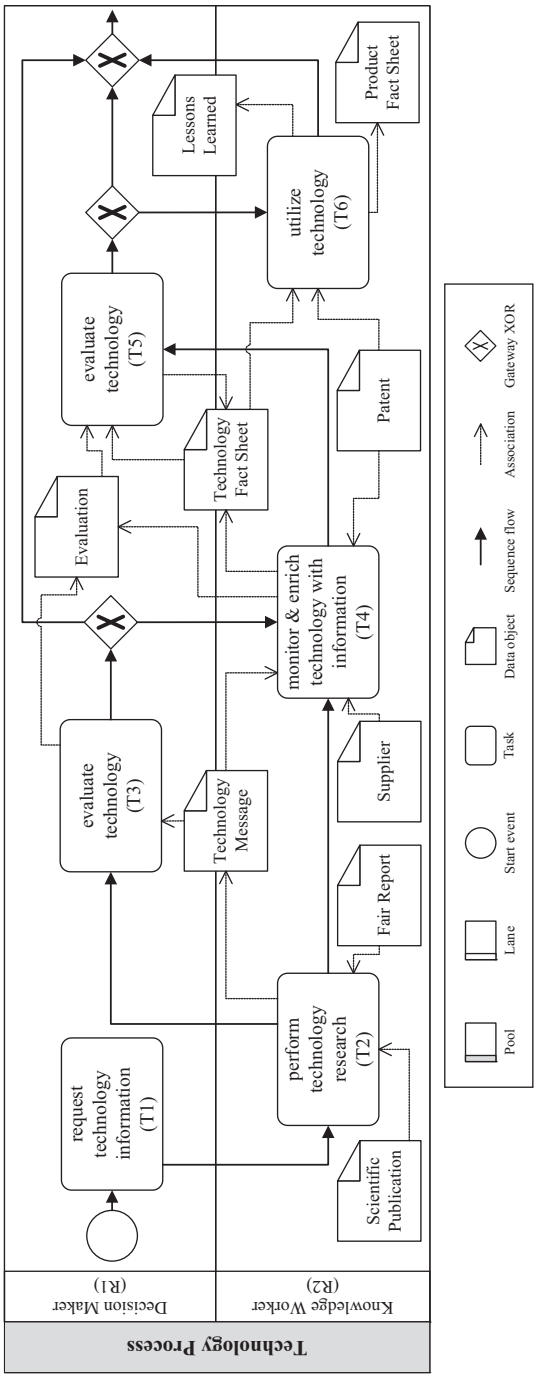


Figure 25.
Technology process

5.1 Use case: technology monitoring

Technology monitoring can help companies to detect scientific, technical or socioeconomic events that have an impact on a company's business (Bradford Ashton *et al.*, 1994). Process tasks include the monitoring of the following:

- patent publications in the area of a specific topic of interest;
- scientific publications (e.g. conference papers, dissertations, books); and
- competitors (e.g. analysis of technology usage announcements).

Usually, such information has already been available in a company, but is physically stored at different places. Consider an engineer performing a search on a specific topic. Assume that he discovers an interesting publication, which can support users when solving a technical problem. The engineer saves the document on a shared drive and forwards the document to a colleague from another department. Another example is a technology analyst looking for information for a particular field of interest. If the research result is unsatisfactory, a technology scout can be hired to monitor significant technology centers, i.e. concentrated domiciles of high-tech corporations. Tasks of a technology scout include visiting fairs or interrogating the personal network. Results are usually reported by email in form of a document or a presentation.

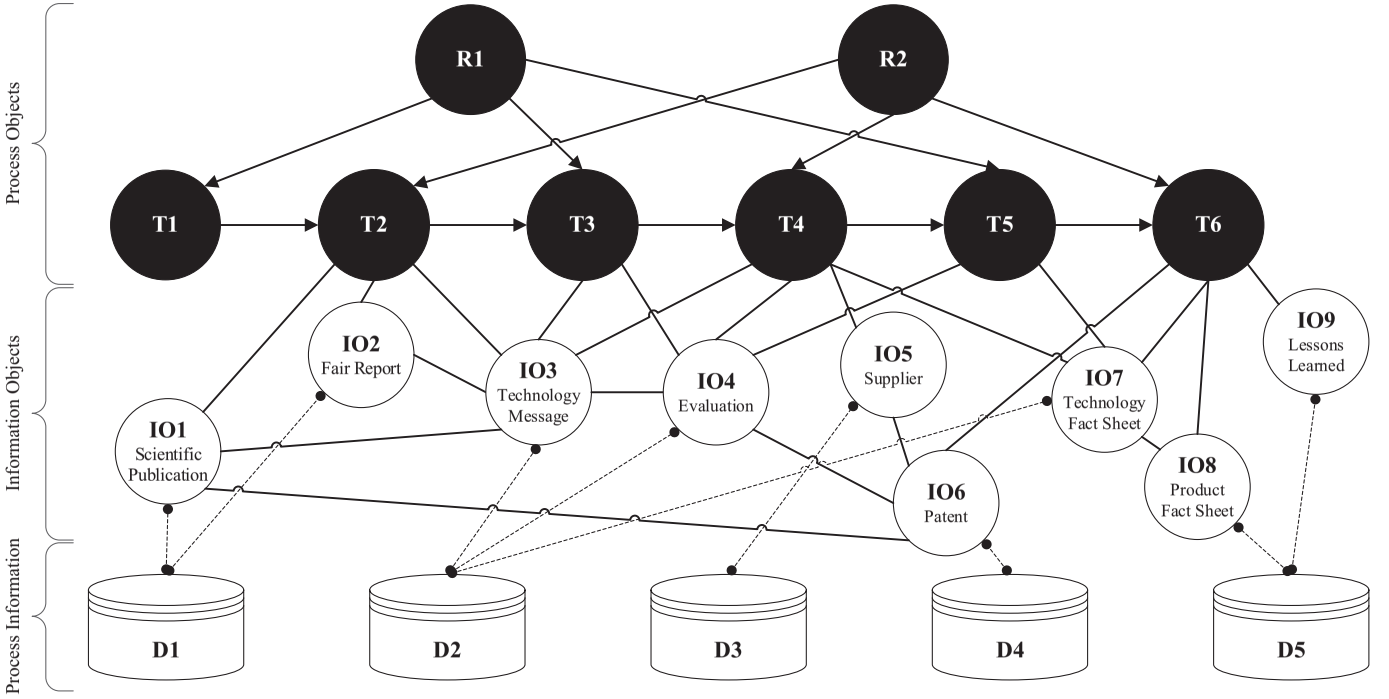
Altogether, as illustrated with the examples, information is, in the context of the technology process, typically gathered and stored by knowledge workers at different locations in various data formats. Important information might not reach every stakeholder and the analysis of all available information is challenging.

The SN approach closes this gap by integrating siloed (technology) information, bringing it into the context of process and other information objects (cf. Figure 26), and keeping it up-to-date.

Therefore, in a first step, the process schema (cf. Figure 25) is integrated (Michelberger *et al.*, 2012b). Process schema elements such as tasks, events, data objects, roles, sequence flows, message flows, associations or gateways are first identified and then used to create the first stage of the SN (R1 and R2, T1-T6 in Figure 26). Afterward, process instances (e.g. request for information, decision for technology usage, allocation of funds) of the integrated process schemas are included as well. Besides the process elements, corresponding metadata, such as author, creation date or modification date are also considered and associated with the process elements. Certain metadata is automatically available (e.g. creation date, modification date, unique identifier), whereas other metadata has to be defined manually (e.g. deadline, project milestone, temporal process constraint or quality gate).

In the second step, process information schemas (e.g. technology fact sheet templates or questionnaires for technology evaluation) and process information instances (e.g. technology fact sheets or completed technology evaluations) are integrated in the SN. Data sources can be folders on a shared drive containing technology information (D1 and D2 in Figure 26; e.g. pdf or office documents), a supplier management system (D3), a patent database (D4) or a project management system (D5). Metadata, such as author, file type or revision number is attached to information objects accordingly.

In the third step, explicit relations are identified and integrated in the SN. Examples include the transformation of relations between process objects (e.g. sequences, role associations derived from BPMN models) or foreign key relations (e.g. author or



Notes: R = role; T = task; IO = information object; D = data source; •----• transformation into information object

Figure 26.
SN object representation of the technology process

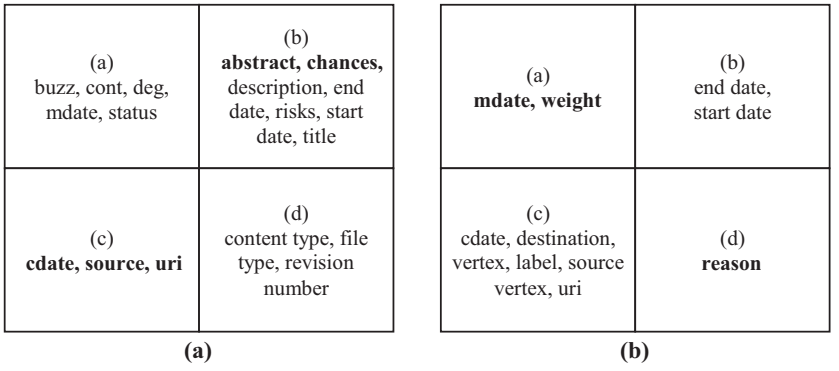
technology fact sheet derived from a relational database). The last integration step is the detection of implicit relations (e.g. similarity relations between information objects).

Overall, the mentioned steps of the illustrated use case result in the property classification (cf. Section 3.2) and configuration of the SN as shown in Figure 27.

Compared to the classification in Section 4.1, we added additional properties. The status property [cf. Figure 27(a)] contains the categorization of an information object based on its type. The status property can, for example, be used to store the legal status of a patent (e.g. applied, granted), the maturity status of a technology (e.g. research, development, ready) or the progress status of ongoing product developments (e.g. open, started, finished, approved). In turn, process objects may use this property to store their process status (e.g. open, warning, problem, canceled). Note that status changes usually trigger further process steps (e.g. "evaluate the application of a technology" when the status is changed to ready).

Optional and static properties are extended by revision number and content type. Revision number is used for tracking changes of process and information objects, whereas content type specifies whether an object contains information about a technology, patent, product or supplier. In most cases, the required information can be directly gathered from the data source (e.g. patent database or technology fact sheet).

The title is used to store names, e.g. a task name, a technology name, a product acronym, a scientific publication, a patent or a supplier name. Usually, the title does not change very often but can change over time. As title might not be available for every object, it is classified as optional/dynamic. Further optional/dynamic properties are abstract, description (e.g. problem statement, possible solutions), chances and risks, which allow users to provide further details about process and information objects. Properties start date and end date, in turn, store temporal information on a process or information object. For example, in the context of a process object, they define when a task must be started or completed. In the context of information objects, the properties can specify the report date, the estimated maturity date of a technology as well as the production period of a product. Further, they can store patent application and expiration dates. Note that such dates allow detecting complex temporal dependencies, e.g. whether a technology is mature, before the production of a new product begins (cf. Section 5.2).



(a) **(b)**

Notes: (a) Objects; (b) relations

Figure 27.
Property classification for the technology process

Analogously, we add properties start and end dates for relations [cf. Figure 27(b)] and use them for temporal restrictions to trigger endogenous changes. Examples include patent expirations, temporary roles (e.g. vacation replacement) or supplier changes.

Furthermore, we enrich the SN with relation labels like "transfers into" or "has invented", which we derive from the business process instances (e.g. technology transfers into a product) and available process information (e.g. patent inventor). For example, transfer information or inventors can be derived from technology fact sheets and transformed to relations accordingly. Inventors of a patent can be directly derived and processed from a patent database, whereas products usually do not have an inventor. Note that the additional semantics of relations will be the basis for further application scenarios (cf. Section 5.2).

After configuration, the SN is ready for use. The first step of the technology process, the technology identification, can be supported by integrating, structuring and interrelating information from underlying data sources (cf. Section 2). Examples of technology related artifacts are illustrated in Table I. Knowledge workers and decision-makers then benefit from a unified single point of information access where they can search and filter available technology information. Consider an engineer looking for information about autonomous driving. He might be able to find a new approach which is currently evaluated by another department. Further, he may follow related information objects and detect a scientific publication describing a novel collision avoidance technology. Another engineer can, starting with a technology of interest, identify technology alternatives by filtering related technologies which are connected by "is similar to" relations. Note that related information objects can provide hints for the solution of a technical problem as well.

The second step, technology monitoring, can be supported by SN maintenance, i.e. by adapting endogenous and exogenous changes (cf. Section 3.1). Consider knowledge workers and decision-makers from different teams who are continuously updating technology information in documents or enterprise information systems. A major breakthrough, such as a solution for a specific technical problem, the discovery or the maturity of a technology, is usually filed, but might not reach all knowledge workers who had interest. The SN captures such changes immediately and sends notifications accordingly (i.e. by role or author associations of similar objects). Decision-makers

Table I. Technology-related artifacts

Artifact	Characteristics
Fact sheets	Strategy, topic field, idea, technology, project, material, patent, competence, product, production technique
Guidelines and templates	Checklist, order, request form (e.g. for funds)
Scientific publications	Scientific paper, master thesis, dissertation, survey, journal article
Reports and protocols	Meeting and workshop protocol, decision protocol, fair report, trend report, customer feedback, internal research and laboratory reports, competitive and market analysis, press release, reverse engineering report
Lessons learned and best practices	Project documentation, workshop preparation, strategies and creativity techniques

receive notifications when a new maturity level of a technology is reached or additional economic information, e.g. on cost, becomes available.

Therefore, the SN captures exogenous and endogenous changes, i.e. if new information (e.g. a patent or information about a technology supplier) becomes available or objects and relations need further processing (e.g. a relation or information object is outdated). Based on such changes, the SN is able to initiate further process steps by sending notifications.

5.2 Use case: decision support

Building on the previous use case, we demonstrate how technology analysis can be enhanced through SN maintenance.

Increased market dynamics, changing customer demands, extended product portfolios, multiple stakeholders or uncertainty (e.g. about consequences of a decision) are examples of challenges which executives have to cope with in enterprises when making decisions. Such factors typically increase the complexity of decisions, making it difficult to determine and decide for the best option. For example, a broader product range and variety on product derivates require higher density of decision-making with prevision and deeper understanding of expected interdependencies (e.g. complexity of interdependencies of products and their components). Usually, many tasks related to the decision process are performed manually which can be very time-consuming. As example, consider the preparation of a strategic roadmap for a management meeting or the request for information from process stakeholders. Thus, providing up-to-date views on integrated and cross-linked information by request can be a business benefit, e.g. to reduce preparation time for decision-makers.

As example, a decision-maker has to consider the maturity level of a technology and compare cost, benefit, quality and availability with other alternatives – if available. Examples of general questions are as follows: does the technology fit into the product portfolio? When will the technology be available? Can the technology be implemented with reasonable cost?

Therefore, the on-demand availability of an integrated view on all information required for analysis is desirable (cf. Section 4.3).

The SN supports this sub-process by unveiling alternatives as well as possible connections to other related information objects (cf. Table I) through semantic relations. To determine a technology's maturity (e.g. status) as well as for capturing temporal facts (e.g. start date, end date), we use the configured properties presented in Section 5.1.

The additional properties allow us to visualize the SN and its underlying objects in a flexible way, i.e. different views on the SN can be generated. One SN view supporting the decision process is a strategic roadmap (cf. Figure 28). A decision-maker can spot temporal aspects of technologies, like expected maturity dates, when arranging technology objects on a timeline.

Additionally, related process information like projects, patents or materials can be added to the view, which allows detecting time constraints (e.g. whether a technology will be ready or whether a technology is still protected by a patent when a project requires this technology). A search and filter functionality limits the result to relevant objects and, therefore, reduces the view to relevant information by using available properties (e.g. creation date, content type, file type or status). Evaluations (e.g. chances and risk analysis) from experts like engineers can support opinion making.

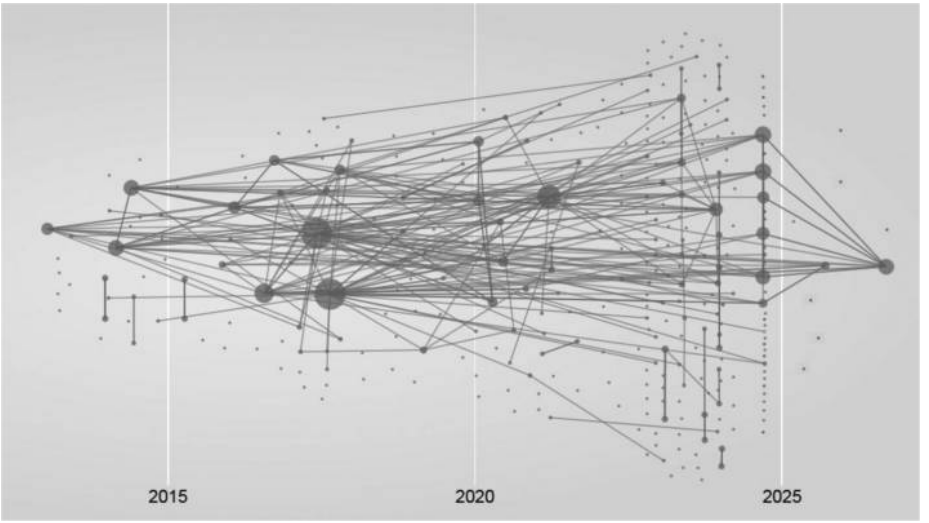


Figure 28.
Strategic roadmap (data masked)

For example, a decision-maker filters technologies with status "ready" and product development projects which have not started yet and only selects "transfers into" relations. By arranging these objects and relations on the x-axis with temporal information, using start and end dates, critical product development processes (in case a technology might not be mature) can be identified. Furthermore, the visualization can be triggered and adapted on demand avoiding manual preparation work.

Such views on the SN are not limited to time-based charts. Consider a portfolio chart which groups technologies by their maturity, expected cost savings or fields of interest (e.g. autonomous driving).

Therefore, SN maintenance does not only allow for the integration and global search of technology information but also enables real-time analysis.

Summing up, a maintained SN supports the technology process by providing access to up-to-date technology information in an integrated, connected and searchable way. Process steps, like technology analysis, technology monitoring, technology controlling and search for alternatives, can be supported. Related technologies can be identified and process steps be triggered by capturing endogenous and exogenous changes, e.g. through notifications in case of a new technology or an updated maturity level. Furthermore, decision-makers can be supported by delivering information through adequate views on demand.

In future work, we will support further process steps to allow rationalizing decisions by socializing information in terms of evaluation (e.g. capturing opinions of engineers on a technology's maturity in a more profound way) and enrich internal information objects with information from the World Wide Web (e.g. selected technology blogs).

6. Related work

An SN represents domain-specific knowledge in a structured and machine-interpretable way (Reichenberger, 2010). Generally, various types of SNs exist. Figure 29 shows the most common approaches, i.e. associative networks, topic networks, fact networks, and

ontologies. The x-axis represents the effort required to create an SN, whereas the y-axis represents the degree of support an SN provides for particular use cases such as search refinement, semantic search, visualization or reasoning. According to Reichenberger (2010), we distinguish between light- and heavy-weighted SNs.

As shown in Figure 29, the effort for creating *associative networks* (Findler, 1979) as well as their degree of support are low. Associative networks are mainly used for search refinement. In turn, topic networks (Park and Hunting, 2002) provide a higher degree of support than associative networks, but the effort for creating them is significantly higher as well. They are often used for realizing a simple navigation in SNs and for visualizing related topics. In turn, *fact networks* (Reichenberger, 2010) provide a higher degree of support (e.g. concepts are supported). They are used for realizing personalized views (e.g. context-aware search) and navigation trees (e.g. moderated search). Finally, the highest degree of support is offered by ontologies (Lacy, 2005; Stuckenschmidt, 2009), which provide good results with respect to conceptualization and delimitation of concepts. However, huge manual effort is needed to create high-quality ontologies. Important uses cases are semantic search and reasoning.

Unlike existing SNs, an SN focuses on information and process objects as well as their relations. Less manual effort is needed to create or maintain an SN (Wurzer, 2008). Note that research in the field of SNs has mainly focused on the representation of domain-specific knowledge in a structured and machine-interpretable form. What has been neglected, however, is the maintenance of SNs. Generally, SNs have in common that they must be maintained. Depending on the type of the SN (cf. Figure 29), however, the level of maintenance effort varies widely. Commonly, the higher the effort to create an SN, the higher the maintenance effort will be. Semi-automatic maintenance approaches are provided, for example, by Čapek (2009), Gargouri *et al.* (2003) and Dinh *et al.* (2014). However, these approaches cannot be directly applied to the SN, as they do not allow for entirely automated SN maintenance as it is necessary in many scenarios.

Moreover, in recent years, various approaches were proposed to tackle the delivery of information to users including data warehousing (Kimball and Ross, 2013), business intelligence (Kolb, 2012), decision support systems (Sauter, 2011) and enterprise content management (Rockley and Cooper, 2012). Data warehousing, for example, rather focuses on the creation of an integrated database (Lechtenbörger, 2001). Opposed to this, an SN deals with the delivery of process information to support the effective and efficient execution of business processes. Traditional business intelligence, in turn, enables data analysis and is usually completely isolated from business process execution (Bucher and Dinter, 2008). Moreover, information supply is often restricted to decision-makers on executive level (Baars and Kemper, 2008; Rouhani *et al.*, 2012).

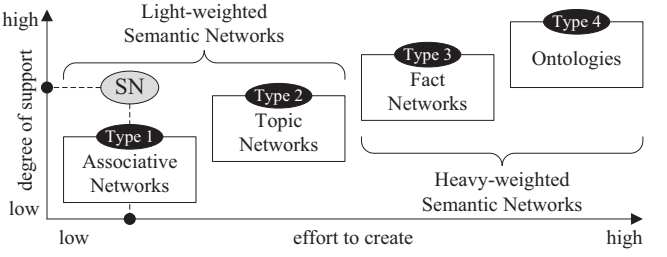


Figure 29.
Types of SNs

Conversely, our approach focuses on the integration and analysis of process information as well as its delivery to both knowledge workers and decision-makers. In contrast, decision support systems support decision-making, i.e. they serve the executive level (Janakiraman and Sarukesi, 2004) and are usually based on structured data (e.g. sales figures). Enterprise content management, in turn, deals with the management of information across enterprises referring to strategies, methods and tools (Cameron, 2011).

7. Summary and outlook

This paper presented an approach for SN maintenance, i.e. for adopting exogenous and endogenous changes. For this purpose, we presented three algorithms based on the property classification, which allows for an efficient SN maintenance.

The validation confirms that the SN maintenance approach is able to keep SNs synchronized with underlying data sources in reasonable time. Moreover, we applied the algorithms to a real-world case, i.e. we validated them based on an implementation in the automotive domain. Furthermore, we illustrated the practicability and usability with two real-world use cases from the technology process.

In future, we will improve the algorithms to support self-learning components with a focus on endogenous changes. Therefore, we will extend the information extraction component to retrieve additional facts (e.g. competitor activities) which can be further processed by a reasoning component. Additionally, we will add taxonomic support so that automatic maintenance is still ensured, i.e. the component will not affect the algorithms. The extensions will allow us to optimize search and filter capabilities (e.g. extended facets for filtering) as well as inferring new facts from properties (e.g. "competitor launched a new product in China"), which may result in further relations (e.g. "is active" relation to an information object describing the Asian markets). Therefore, we will increase the degree of support (cf. Section 6) while keeping the effort low.

Notes

1. www.playframework.com/
2. http://getbootstrap.com/
3. http://d3js.org/
4. http://jquery.com/
5. http://sourceforge.net/directory/?q=nipro
6. http://lucene.apache.org/
7. www.mysql.com/
8. http://nipro.hs-weingarten.de/casestudy

References

Baars, H. and Kemper, H.G. (2008), "Management support with structured and unstructured data – an integrated business intelligence framework", *Journal of Information Systems Management*, Vol. 25 No. 2, pp. 132-148.

Bradford Ashton, W., Johnson, A.H. and Stacey, G.S. (1994), "Monitoring science and technology for competitive advantage", *Competitive Intelligence Review*, Vol. 5 No. 1, pp. 5-16.

Bucher, T. and Dinter, B. (2008), "Process orientation of information logistics – an empirical analysis to assess benefits, design factors, and realization approaches", *Proceedingsc 41st Hawaii Int'l Conference on System Sciences (HICSS'08)*, *Hawaii, HI*, pp. 392-402.

Cameron, S.A. (2011), *Enterprise Content Management: A Business and Technical Guide*, 1st ed., British Informatics Society, Swindon.

Čapek, T. (2009), "Semantic network integrity maintenance via heuristic semi-automatic tests", *Proceedings Raslan Workshop 2009*, *Brno*, pp. 63-67.

Dinh, D., Dos Reis, J.C., Pruski, C., Da Silveira, M. and Reynaud-Delaître, C. (2014), "Identifying relevant concept attributes to support mapping maintenance under ontology evolution", *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 29, pp. 53-66.

Findler, N.V. (1979), *Associative Networks*, 1st ed., Academic Press, New York, NY.

Freund, J. and Rücker, B. (2012), *Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company*, 1st ed., CreateSpace Independent Publishing Platform, North Charleston.

Gargouri, Y., Lefebvre, B. and Meunier, J.G. (2003), "Ontology maintenance using textual analysis", *Proceedings 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, *Orlando, FL*, pp. 248-253.

Hipp, M., Michelberger, B., Mutschler, B. and Reichert, M. (2013), "A framework for the intelligent delivery and user-adequate visualization of process information", *Proceedings 28th Symposium on Applied Computing (SAC'13)*, *Coimbra*, pp. 1383-1390.

Hipp, M., Mutschler, B., Michelberger, B. and Reichert, M. (2014), "Navigating in process model repositories and enterprise process information", *Proceedings 8th Int'l Conference on Research Challenges in Information Science (RCIS'14)*, *Marrakesh*, pp. 1-12.

Hipp, M., Mutschler, B. and Reichert, M. (2011), "On the context-aware, personalized delivery of process information: viewpoints, problems, and requirements", *Proceedings 6th Int'l Conference on Availability, Reliability and Security (ARES'11)*, *Vienna*, pp. 390-397.

Hotho, A., Nürnberger, A. and Paaß, G. (2005), "A brief survey of text mining", *LDV Forum – GLDV Journal for Computational Linguistics and Language Technology*, Vol. 20 No. 1, pp. 19-62.

Janakiraman, V.S. and Sarukesi, K. (2004), *Decision Support Systems*, 1st ed., Prentice Hall, New Delhi.

Kimball, R. and Ross, M. (2013), *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed., John Wiley & Sons, Indianapolis, Indiana.

Kitchenham, B., Pickard, L. and Pfleeger, S.L. (1995), "Case studies for method and tool evaluation", *IEEE Software*, Vol. 12 No. 4, pp. 52-62.

Kolb, J.M. (2012), *Business Intelligence in Plain Language: A Practical Guide to Data Mining and Business Analytics*, 2nd ed., Applied Data Labs, Plainfield, NJ.

Lacy, L.W. (2005), *OWL: Representing Information Using the Web Ontology Language*, 1st ed., Trafford Publishing, Victoria.

Lechtenbörger, J. (2001), *Data Warehouse Schema Design*, 1st ed., Infix, Saint Augustin.

Michelberger, B., Mutschler, B., Hipp, M. and Reichert, M. (2013), "Determining the link and rate popularity of enterprise process information", *Proceedings 21st Int'l Conference on Cooperative Information Systems (CoopIS'13)*, *Graz*, pp. 112-129.

Michelberger, B., Mutschler, B. and Reichert, M. (2011a), "Towards process-oriented information logistics: why quality dimensions of process information matter", *Proceedings 4th Int'l*

Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'11), Hamburg, pp. 107-120.

Michelberger, B., Mutschler, B. and Reichert, M. (2011b), "On handling process information: results from case studies and a survey", *Proceedings 2nd Int'l Workshop on Empirical Research in Business Process Management (ER-BPM'11)*, *Clermont-Ferrand*, pp. 333-344.

Michelberger, B., Mutschler, B. and Reichert, M. (2012a), "A context framework for process-oriented information logistics", *Proceedings 15th Int'l Conference on Business Information Systems (BIS'12)*, *Vilnius*, pp. 260-271.

Michelberger, B., Mutschler, B. and Reichert, M. (2012b), "Process-oriented information logistics: aligning enterprise information with business processes", *Proceedings 16th Int'l Enterprise Computing Conference (EDOC'12)*, *Beijing*, pp. 21-30.

Michelberger, B., Ulmschneider, K., Glimm, B., Mutschler, B. and Reichert, M. (2014), "Maintaining semantic networks: challenges and algorithms", *Proceedings 16th Int'l Conference on Information Integration and Web-based Applications & Services (iiWAS'14)*, *Hanoi*, pp. 365-374.

Mundbrod, N. and Reichert, M. (2014), "Process-aware task management support for knowledge-intensive business processes: findings, challenges, requirements", *Proceedings 3rd Int'l Workshop on Adaptive Case Management and Other Non-Workflow Approaches to BPM (AdaptiveCM'14)*, *Ulm*, pp. 116-125.

Oertelt, S. (2009), – *Ganzheitliches Verfahren zur Priorisierung und Steuerung von Vorentwicklungsprojekten*, 1st ed., Shaker, Magdeburg.

Oertelt, S. and Ulmschneider, K. (2013), "Prozessintegrierter Einsatz virtueller methoden im strategischen technologie und innovationsmanagement", *KnowTech – Wissensmanagement und Social – Markterfolg im Innovationswettbewerb*, Hanau, pp. 485-498.

Park, J. and Hunting, S. (2002), *XML Topic Maps: Creating and Using Topic Maps for the Web*, 1st ed., Addison-Wesley, Boston, MA.

Reichenberger, K. (2010), *Kompendium semantische Netze: Konzepte, Technologie, Modellierung*, 1st ed., Springer, Berlin.

Reichert, M. and Weber, B. (2012), *Enabling Flexibility in Process-Aware Information Systems*, 1st ed., Springer, Berlin.

Rockley, A. and Cooper, C. (2012), *Managing Enterprise Content: A Unified Content Strategy*, 2nd ed., New Riders, Berkely.

Rouhani, S., Asgari, S. and Mirhosseini, S.V. (2012), "Review study: business intelligence concepts and approaches", *American Journal of Scientific Research*, Vol. 50, pp. 62-75.

Rowley, J. (2007), "The wisdom hierarchy: representations of the DIKW hierarchy", *Journal of Information Science*, Vol. 33 No. 2, pp. 163-180.

Runeson, P. and Höst, M. (2009), "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, Vol. 14 No. 2, pp. 131-164.

Sauter, V.L. (2011), *Decision Support Systems for Business Intelligence*, 2nd ed., John Wiley & Sons, Hoboken.

Scheer, A.W. (2002), *ARIS – Vom Geschäftsprozess zum Anwendungssystem*, 4th ed., Springer, Berlin.

Stuckenschmidt, H. (2009), *Ontologien: Konzepte, Technologien und Anwendungen*, 1st ed., Springer, Berlin.

Wurzer, J. (2008), "New approach for semantic web by automatic semantics", *Proceeding 2nd European Semantic Technology Conference (ESCT'08)*, *Vienna*.

Yin, R.K. (2009), *Case Study Research: Design and Methods*, 4th ed., Sage Publications, Thousand Oaks.

Further reading:

Bernstein, P.A. and Haas, L.M. (2008), "Information integration in the enterprise", *Communications of the ACM*, Vol. 51 No. 9, pp. 72-79.

Bradford Ashton, W., Kinzey, B.R. and Gunn, M.E. (1991), "A structured approach for monitoring science and technology developments", *Int'l J of Technology Management*, Vol. 6 Nos 1/2, pp. 91-111.

Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K. and Hanschke, I. (2011), "Requirements for automated enterprise architecture model maintenance – a requirements analysis based on a literature review and an exploratory survey", *Proceedings 13th Int'l Conference on Enterprise Information Systems (ICEIS'11)*, *Beijing*, pp. 325-337.

Fischer, R., Aier, S. and Winter, R. (2007), "A federated approach to enterprise architecture model maintenance", *2nd Int'l Workshop on Enterprise Modelling and Information Systems Architectures*, *Sankt Goar*, pp. 9-22.

Lewandowski, D. (2005), "Web searching, search engines and information retrieval", *Information Services & Use*, Vol. 18 No. 3.

Corresponding author
Klaus Ulmschneider can be contacted at: klaus.ulmschneider@uni-ulm.de