Full Semantic Transparency: Overcoming Boundaries of Applications

Andrea Kohlhase, Michael Kohlhase, Constantin Jucovschi, Alexandru Toader

Jacobs University Bremen, Germany
{a.kohlhase, m.kohlhase, c.jucovschi,
a.toader}@jacobs-university.de

Abstract. Complex workflows require intelligent interactions. In this paper we attack the problem of combining user interfaces of specialized applications that support different aspects of objects in scientific/technical workflows with semantic technologies.
We analyze the problem in terms of the (new) notion of full semantic transparency, i.e., the property of user interfaces to give full access to an underlying semantic object even beyond application lines. In a multi-application case full semantic transparency is difficult, but can be achieved by representing the semantic objects in a structured ontology and actively supporting the application-specific framings of an object in a semantic interface manager.
We evaluate the proposed framework in a situation where aspects of technical constructions are distributed across a CAD system, a spreadsheet application, and a knowledge base.

Keywords: Full semantic transparency; multi-application Semantic Alliance; frame shifts;spreadsheets; CAD systems; semantic services

1 Introduction

As the objects we create and interact with get more and more complex, the corresponding workflows increase in complexity as well. Part of this complexity is mitigated by specialized support systems for specific subgoals. But this only shifts the underlying problem from mastering complexity to mastering sequences of autonomous steps to smoothly perform a given task. Even though subgoals might get simpler to reach, individual subtasks often require distinct support systems. In particular, users need to use multiple applications offering support to address specific rather unpredictable aspects of a task. Therefore, this user-driven workflow, we call it "**useflow**", has become a topic of research interest in recent years.

To clarify the issues, let us look at a concrete multi-application useflow, which will serve as a running example for this paper and for which we implemented a fully semantically transparent interface.

1.1 Running Example

Charles A. is a design engineer in a small company that specializes in the production of small and fast luxury yachts. As it is a small company Charles needs to analyze

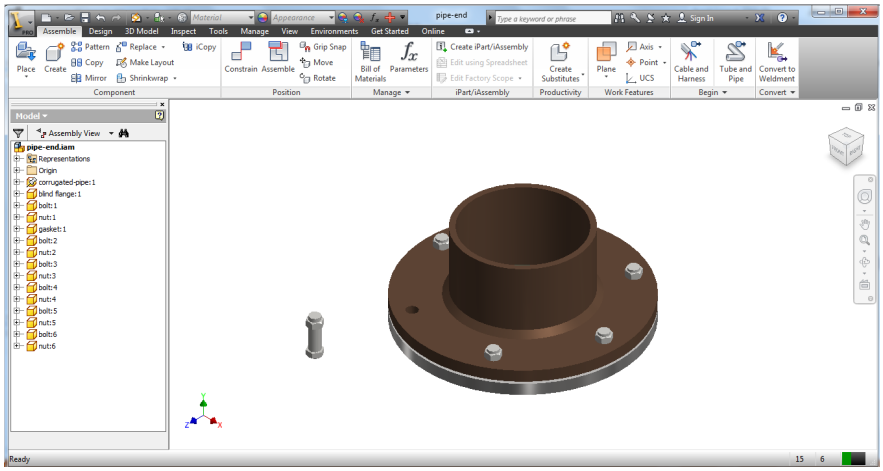


Fig. 1. A Pipe End Representation in a CAD System

customer needs in interviews, has to create design solutions with a CAD system like `Autodesk Inventor`, and also decides on production issues.

The design tasks range from the yacht's shape down to pipe fittings (see Fig. 1 – here a blind flange fitting, a particular pipe end that allows allows access to the piping system for revision and cleaning; we will refer to it as a "**pipe end**"[1] in the following).

As customers often set tight budget restrictions, Charles uses several spreadsheets to inform himself about vendors' pricing schemes for frequently needed items as for example pipe ends. In Fig. 2 we see such a spreadsheet for pricing details of pipe ends

G7 | {= INDEX(INDIRECT("'"&G$5&"'!$F$8:$F$1000");MATCH($A7&$B7&$C7&$D7&$E7;INDIRECT("'"&G$5&"'!$A$8:$

Pricing of Pipe-Ends (Blind Flange Fittings)

Quantity (Pipe Ends): 104

							Vendor A		Vendor B		Vendor C	
Component	Thread	Color	Head	Type	Quantity per flange		by piece	total	by piece	total	by piece	total
bolt	M15	black	machine		6		0,297 €	185,328 €	0,336 €	209,664 €	0,340 €	211,848 €
nut	M15	black			6		0,499 €	311,351 €	0,480 €	299,520 €	0,485 €	302,640 €
gasket				standard	1		2,020 €	210,038 €	1,920 €	199,680 €	2,910 €	302,640 €
flange	M15	black			1		1,069 €	111,197 €	1,344 €	139,776 €	1,071 €	111,372 €
blind flange	M15	black			1		0,879 €	91,428 €	0,192 €	19,968 €	0,776 €	80,704 €
						Price	8,74 €	909,34 €	8,35 €	868,61 €	9,70 €	1.009,20 €

Fig. 2. A Spreadsheet for the Calculation of Pipe End Prices

(e.g. in `MS Excel`). Charles can 'play with the numbers' by entering distinct quantities

[1] This assembly is chosen for purely expository reasons, nothing in this paper hinges on this choice.

or specifying different attributes of the components of a pipe end to compare prices between vendors. The price conditions for components like flanges and bolts are stored in other worksheets in the same workbook (see example in Fig. 3). In this industrial sector it is customary that vendors offer percentual discounts for sale items depending on the quantity of units to be ordered.

Charles' useflow forces him to switch between the CAD program and the spreadsheet system quite frequently.

Price List of Vendor A

Conditions depending on Quantity of Pipe End Units

100	1000	10000	>10000
100,000%	99,000%	95,000%	90,000%

Component	Thread	Color	Head	Type	Basic Price
bolt	M15	silver	carriage		0,450 €
bolt	M15	silver	stove		0,460 €
bolt	M15	black	machine		0,300 €
bolt	M15	silver	machine		0,310 €
bolt	M15	red	machine		0,340 €
bolt	M15	black	machine		0,350 €
bolt	M16	black	machine		0,300 €
bolt	M16	black	machine		0,350 €
nut	M15	black			0,504 €
nut	M16	black			0,498 €
gasket				standard	2,040 €
flange	M15	black			1,080 €
flange	M15	silver			1,080 €
flange	M16	black			1,090 €
blind flange	M15	black			0,888 €
blind flange	M16	black			0,888 €
blind flange	M17	black			0,888 €

Fig. 3. A Spreadsheet as Database

1.2 Contribution and Structure of the Paper

We note that application switches in a useflow become necessary to access different aspects of an object currently in focus (e.g. pricing for the currently active CAD assembly). In particular, Charles would like to have the application he switches to already focused on the target object: that object in the target application that represents the respective aspect of the source object (e.g. the price of the type of bolt focused in the CAD system). Normal application switchers (e.g. Alt-Tab) do not provide this quality.

In this paper, we use sense-making theory to better understand this "aspect switching" resulting in a model of "*frame shifts*" and an interface requirement we call "*full semantic transparency*", i.e., an UI property that provides users access to the full meaning of information objects. Moreover, we present a fully semantically transparent interface. In particular, we strive to support frame shifts via a semantic "mash-up" of user interfaces on a fine-grained object level. Concretely, we make use of an existing semantic interaction framework, which links the semantic objects in applications to concepts in

a structured background ontology. These concepts act as a "pivot table" that allow us to predict and focus target objects in frame shifts.

Before we present the interface solution in Section 4, we survey related work (Section 2) and establish a model for frame shifts and develop the concept of "full semantic transparency" (Section 3). Section 5 concludes the paper.

2 Related Work

Generally there are three ways to address useflow smoothing and application interoperability:

U1 Merge applications into a single large system.
U2 Make applications interoperable on the data level.
U3 Make applications interoperable on the user-interface level.

Option **U1** is only feasible for very established and high-volume workflows. A good example for this option are the office suites that combine word processors, spreadsheets, slide presenters, and drawing applications under one joint interface. Other examples consist in integrated trade-specific solutions that cover the main workflows of a specific industry sector. The workflows covered by such vertically-integrated systems are usually quite rigid, since any change in workflow needs to be reproduced in the system and thus needs a system extension. Thus, such systems cannot cater to the needs of single users; moreover, Charles' small company could probably not afford one of these large-scale integrated systems.

Option **U2** is facilitated by the trend towards offering open APIs that allow access to its underlying domain and event models. In [6] e.g., Diaz et al. describe a method that uses deep annotation, i.e., an ontology representation, to define the data flow from one portlet to another. Therefore, objects are related semantically on a data level. Most existing approaches though don't pay attention to the resulting user-interface challenges, that is a users' access to their full meaning. In [7] this approach is taken up on a conceptual level, stating for example, that a document " will no longer be a mere file, but rather a knowledge base resource", but a closer reading still reveals a similar bias towards machine-understanding. Another approach is the Gnowsis Semantic Desktop framework [15], "a tool for personal information management" [ibd., p. 887]. This collects data from various desktop-based applications in an RDF-based "personal information management ontology" (PIMO), which services can tap e.g. to mash-up data by distinct applications. Even though the data modeling component is similar to what we will present below, the focus of the Semantic Desktop is on information retrieval and semantic information management services (e.g. cataloging or sorting images according to the PIMO) and – to the best of our knowledge – not on UI-level support for useflows, which is what we concentrate on.

Option **U3** has gotten less general attention, except for a call for uniform look-and-feel across applications driven by usability concerns. The underlying reason consists of the idiosyncrasies of user-interfaces based on the particular technologies used and tailored task support. User interface mash-ups only seem simple to realize if the applications share the same base technology, e.g. for web applications. But there are many

non-web applications supporting subtask achievement that are designed as insular systems that excel in their specific domain.

The coordination of multiple visualizations or *mash-up of visualizations* of information is often used when the " information is sufficiently complex to require different types of visualizations for different aspects or layers" [12, p. 715]. North & Shneiderman note that such visualization requirements tend to be rather unforeseeable. Thus, they devised "Snap-Together Visualization" [ibid.], an approach to dynamically support multiple coordinated visualizations based on a relational database model that generates views, which in turn are linked together through different events. In contrast, Bull models the coordination of visualizations via software engineering modeling approaches [2]. Tudorache et al. note that such approaches are essentially based on dynamic communication between independent components, but envisioned beforehand by a software designer and not necessarily employable by an application user [18] involved with situated actions [17,11]. In particular, they are mainly targeting the workflow rather than the useflow.

The unpredictable flexibility requirements for the useflow maybe explained by the difference between a user's task environment and his information environment, the background environment that allows users to acquire more information to take better actions (see [13, p. 20]). Pirolli discusses multiple information foraging models of humans, one of which is the "patch model" [ibd., 31ff]. Here, information is assumed to be distributed in a 'patchy' manner. In our running example, Charles finds information needed in his useflow at different locations in distinct applications. Therefore, we can apply the patch model to Charles' useflow. Obviously, an information forager needs to optimize his use of information patches. In which cases or at which steps in the useflow, for instance, should Charles switch applications in order to get price information about pipe ends? If, for example, Charles wants to minimize the patch distance, he makes use of keyboard shortcuts like "Alt Tab" to switch applications. In contrast, if he wants to maximize information gain on pricing issues for pipe ends, he authors a spreadsheet to support him. To keep customers within their applications, companies draw on this trade-off by enriching environments with added-value services like providing piece-lists of assemblies in grids from within a CAD system. Pirolli shows in [13, p. 38] that reducing between-patch costs improves the overall average rate of information gain and moreover that the optimal gain is achieved by spending less time within a patch.

Therefore, in this paper we concentrate on option **U3** departing from a concept called "semantic transparency" [8], that assumes a user's need for semantic alignments of applications in his useflow, towards "full semantic transparency" based on sense-making theory . In particular, we will apply the notion of "frames" [19] to semantic transparency and propose to address useflow-centered mashing up of user interfaces by semantic technologies.

3 Full Semantic Transparency

If we want to optimize between-patch costs in Charles' information environment, then we have to better understand how humans conceive information. This is a central ques-

tion in *Sense-Making Theory* (see e.g. [5] for an overview). In [19] Weick introduces the terms "**frame**", "**cue**", and "**connection**":

> Frames and cues can be thought of as vocabularies in which words that are more abstract (frames) include and point to other less abstract words (cues) that become sensible in the context created by the more inclusive words. Meaning within vocabularies is relational. A cue in a frame is what makes sense, not the cue alone or the frame alone. Said differently, the substance of sensemaking starts with three elements: a frame, a cue, and a connection.

To anticipate trajectories of information (and act accordingly) Charles needs to make sense of the information in his focus. In a nutshell, this means that he continuously strives to understand the potential connections of the information cues he is perceiving with respect to possibly different frames, i.e., via the UI of applications. Kolko notes the long-term importance of frames in sense-making as a frame can be viewed as "an active perspective that both describes and perceptually changes a given situation" [9].

In the following we suggest a model of sense-making in the process of application switching in UI terms.

Frames via Applications Frames conceptualize knowledge in a way that enables information system stakeholders to make use of the elements of information systems (see [10]). Usually, the terms "frame" and "cue" are employed with respect to people's cognitive abilities, but they are also expressed in the design of their tools. In particular, we can consider applications as tools that frame semantic objects. They are manifestations of frames, thus, we can treat them as frames themselves.

Cues via Information Objects Each application supports information presentation of a specific framing of semantic objects. Therefore, the user interface of the application is composed of UI objects that contain specific kinds of information with respect to the respective representation of semantic objects. Let us call these UI objects "**information objects**". In a CAD system, the assemblies and the parts they are made of are information objects. Consider for instance a hex bolt such as the one on the left of Fig. 1, which is made of a cylindrical shaft and a hexagonal head: while the former is a geometric object the latter two are not. In contrast, in a spreadsheet document cells and tables are information objects: they are mere data containers on a technical level, but their associated properties like the type give their content (e.g. a specific number) a meaning. This meaning of a cell might be implied by other information objects. In particular, the position of a cell within a table consisting of row and column specification carries specific meaning itself. Therefore, we can view information objects with their contents as cues.

Connections via Conventions The information objects in the applications can be seen as constituting a symbolic, domain-specific language for application authors and readers based on common conventions. These can be conventions like the interpretation of a table as a grid having specific properties or application-dependent conventions like cell content being of a specific type. For making sense of information given the connection between information object and application is established with these conventions.

Sense-Making across Frames via Semantic Objects Frame shifts in a certain useflow assume the existence of a meaningful object that aligns these frames. Charles, for example, looks at the object "bolt" from two viewpoints: under the frame of geometric construction from within a CAD system and under the frame of pricing of a product from within a spreadsheet system (see Fig. 4). To stress that an object carries *several* meanings depending on the perspective it is looked at, this is called "**semantic object**". A semantic object functions as a pivot for application shifting, thus, if we establish a semantic object as a node in an according ontology, this node can serve as a pivot for frame shifts.

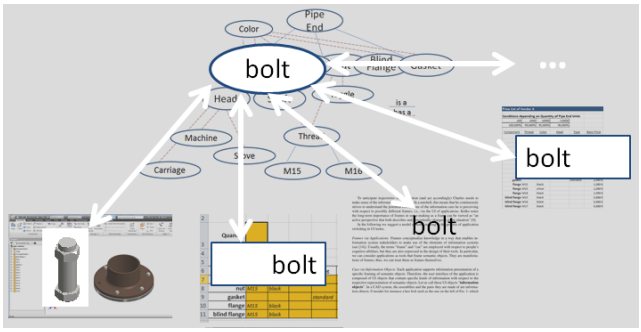


Fig. 4. A Semantic Object as Pivot For Sense-Making

For each frame the representation of a semantic object is stored separately in a document. Each application in turn is a "**player**" of certain document-types (i.e., it plays special documents), it *presents* the representations of semantic objects. The "pipe end" e.g. is represented as `*.iam` file and as `*.xlsm` file. Note that such representations also include representations of other semantic objects like "bolts" or "flanges", that e.g. in the CAD system `Autodesk Inventor` are stored as independent part-files (`*.ips`). In the following we restrict applications to be document players as only those enable application switching as described above.

Sense-Making of Frame Shifts via Full Semantic Transparency [8] introduces the term "**semantic transparency**": "a user interface [is called] semantically transparent, if it enables a user to access its semantic objects and their relations via the corresponding UI objects". But what does "access its semantic objects" mean? In [8] it means access to more information about the semantic object. But in our running example we have observed that respective information objects in the CAD system and the spreadsheet application only handle their very own specific frame of the semantic objects Charles has to deal with in his daily work. In particular, access to a semantic object should be extended to mean access to all *available* frames of a semantic object — enabling a "**full semantic transparency**". Even though – as our example above shows – this is exactly what Charles would need, neither of the user interfaces `MS Excel` and `Autodesk Inventor` involved in our running example allow access to both frames of the object under consideration, that is, neither UI is fully semantically transparent.

Full semantic transparency hinges on the sensemaking process during frame shifts for a semantic object. As we mentioned above, sensemaking has three components: a frame, a cue and a connection. In this paper, we assume that frames are given by particular documents and their players. Moreover, we can view information objects with their contents as cues. But who or what establishes the connection across frames?

Charles aligns the applications, that is, his cognition establishes the connection between a frame and a cue, so that a specific aspect of a semantic object can be communicated. This relation can also be called an **interpretation**, i.e., "the assignment of meaning to data" [1], which in turn can be interpreted as subjective extension or enhancement of the given data [9]. In the following we formalize this connection across frames so that we obtain computer support for frame shifts.

4 Realizing Full Semantic Transparency

The very basic idea to reach full semantic transparency consists in the fact that semantic objects can be modeled in structured ontologies. So instead of trying to connect e.g. the information object "bolt" in the CAD system directly with the information object "bolt" in the spreadsheet system, we formally align both by the detour through a structured ontology (see Fig. 5) which acts as a pivot structure. Semantic technologies handling these ontologies can provide intelligent services and user interfaces.

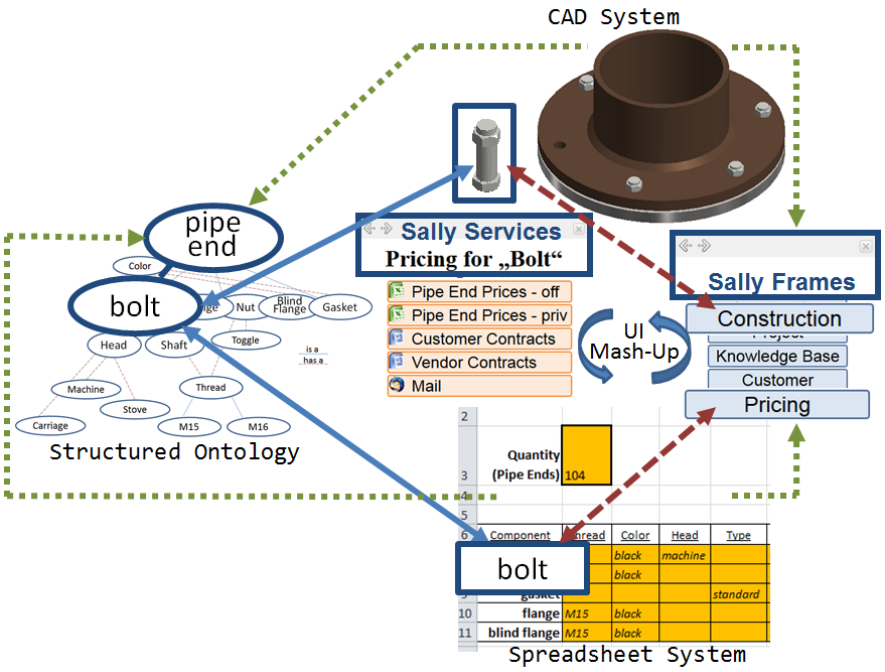


Fig. 5. Mashup of Interfaces

We choose the open source "**`Semantic Alliance`**" framework [3], which allows to superimpose semantic services over an existing (and possibly proprietary) application provided that it gives open-API access to user events as a basis for a join-

ing, fully semantically transparent user interface. The central advantage for our purposes is that the interaction intelligence is encapsulated in an autonomous, application-independent semantic ally called "**`Sally`**", which remains invisible to the user, but which orchestrates the joint user interface. In particular, information objects of applications are linked to concept specifications of the semantic object in a structured background ontology via an "**interpretation mapping**" which can serve as a model for the sense-making component "connection" between a frame and a cue. It can therefore be modelled via this interpretation mapping. If we model frame shifts this way, we can support them in useflows by extending the `Semantic Alliance` framework towards integration of applications of different media types.

4.1 Mashing Up User Interfaces

We have implemented the "**`MA Semantic Alliance`**" ("MA" stands for "multi-application"), an extension of `Semantic Alliance` which allows to integrate multiple applications. It allows to mediate frame switches between applications of different media type in complex useflows.

It has been stressed according to [16, p. 1815] that "sensemakers benefit not just from good sources of information but also from good sources of structure". Moreover, information objects are the UI objects that carry meaning for the user. In particular, clicking those objects is a natural entry-point for access to the semantic object. Therefore, to "unleash [...] sense-making potential" [5, p. 36], the `Sally`-based UI adds interaction elements at two levels:

Sally Services: A service selection menu at the information-object level (Fig. 7[2]). An information object is assumed to be in the user's focus when he left-clicked it, for example the "bolt" in the assembly in Fig. 1. A right-click opens the "Sally Services", a menu for services currently registered with `Sally` and available for the semantic object "bolt" in the current frame. These services provide intelligent user assistance at a very fine-granular level. They can range from a Google search with keyword "bolt" over a definition look-up service in a company-wide knowledge base up to a service in a spreadsheet to calculate prices for the selected bolt in the CAD system (see [3]). Additionally, the respective services can add their own custom interactions, either through custom menus or via popup windows provided by the `Semantic Alliance` framework.

Sally Frames: A frame selection menu (Fig. 6) which allows users to cognitively shift frames very comfortably without necessarily yet shifting applications. The idea is that the user focuses on an information object in one application, which triggers awareness about other meanings of the underlying semantic object. If the interest in another meaning is large enough, he can choose a frame shift via the menu "Sally Frames" and all available services for an information object under the new frame will be shown on demand via Sally Services. `Sally` draws on an ontology of

[2] Note that this layout is still preliminary proof-of-concept realization that will evolve, for instance, Sally Frames could be realized as a dropdown menu that doesn't occupy as much screen space.

frames (including an empty one), which services can register for. Therefore, `Sally` can gather services with respect to a certain frame.

Note that, if the frame for an information object is the regular application frame, then a right-click will deliver the usual context menu - possibly extended by some available services via `Sally`.

To realize the `MA Semantic Alliance`, we modeled the abstract document models (ADM; called "abstract document types" in [3]) for spreadsheets, CAD assemblies, etc. in abstract document ontologies, so that multiple documents and their interpretation mappings can be jointly represented in the form of RDF triples in an in-memory triple store in `Sally`. Whenever a document is opened in the `MA Semantic Alliance`, salient aspects of its contents are exported to the triple store together with information about their interpretations. Instead of direct ADM API calls, `MA Semantic Alliance` services now use SPARQL [14] queries to access the respective information objects. This redesign allows queries across information objects and their interpretations from all open documents. In particular, services can use queries to determine whether they are applicable to a chosen information object (and frame); this information can be used to populate the menus Sally Frames (Fig. 6) and Sally Services (Fig. 7).

Our experiments show that the query-based approach is expressive enough to realize the smooth task-switching capabilities in all studied cases, including our running example which we will now revisit.

4.2 A Use Case with a Fully Semantically Transparent UI

To fortify our intuition about these interface elements on the one hand and to evaluate the expediency of full semantic transparency on the other, let us see how an instance of the `MA Semantic Alliance` combining a CAD system UI, a spreadsheet system UI, and a knowledge base UI transforms Charles' application-switching useflow from our running example to an interaction model allowing full semantic transparency via frame shifts.

We will go through Charles' new useflow step by step (*italicized text*) and discuss the UI repercussions (normal text).

1. **CAD System UI** *Initially, Charles works on the design of a specific pipe end fulfilling some geometric side conditions like strength and pipe diameter. Satisfied, he wants to verify his design in terms of the customer requirements and senses that his pipe end design may violate financial constraints. He has a closer look at the kind of bolt he used (by clicking on one bolt) and verifies that it is a rather expensive carriage bolt (Fig. 6).* Normally, Charles would have had to switch to his spreadsheet application, load the worksheet in Fig. 2, enter part quantities, and evaluate the result.
   (a) **Frame selection** *Now, Charles looks at Sally Frames for the "Pricing" frame.* In this context menu `Sally` offers the user access to all current frames, here e.g. "Pricing", "Knowledge Base", or "Customer". *Under the frame "Knowledge Base" he expects for instance services that deepen his understanding for*

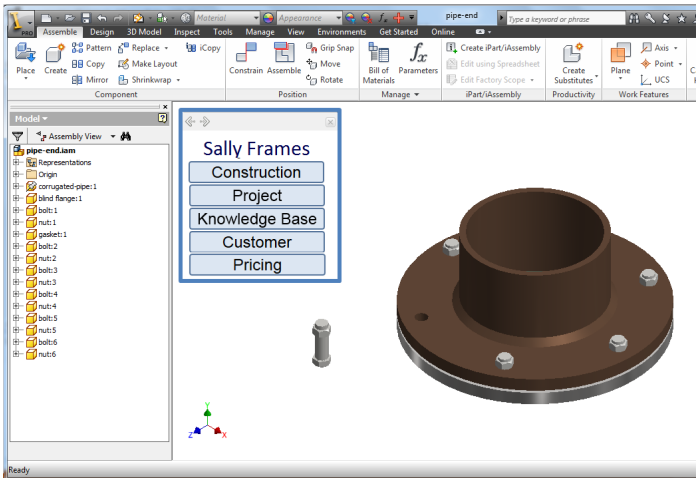


Fig. 6. The Sally Frames Menu for a Pipe End

the concept "pipe end", whereas under "Customer" he wants to get information about exact customer requirements given in the contract for pipe ends. But for now Charles selects the "Pricing" frame. His choice replaces the Sally Frames with the Sally Services menu (containing currently available services for the semantic object "bolt") yielding Fig. 7.

(b) **Service Selection** *Charles studies the Sally Services menu (as seen in Fig. 7).* To determine the available pricing services, `Sally` needs to know the pertinent information object; as Charles has chosen the "bolt" component, `Sally` gathers all pricing services available in an extra window overlaying the CAD application. If those services depend on cue x like "Give me the definition for x", then these services are offered for the content of the selected information object, here for "bolt".

Concretely, Charles has the choice between a company-wide, official spreadsheet for pipe end prices, a private one, that he authored himself as a calculation template, the price conditions for the component "bolt" documented in the customer or vendor contracts for pipe ends, or he can simply check his mail for more informal communications about bolts. Charles opts for the private pricing spreadsheet . . .

2. **Spreadsheet System UI** . . . which orders `Sally` to open the private "pipe end pricing" worksheet with the resp. properties of the selected bolt, which in turn becomes the top most UI, that is, *Charles switched the application (Fig. 8)*. Here, `Sally` makes use of the interpretation mapping for the Sally Services menu in two ways: it can focus on the pertinent worksheet, since the respective table is mapped to the pipe end via the interpretation mapping, and in particular, the quantity cell to the concept of multiplicity of parts in assemblies. *Charles glances over the values and decides that his suspicion was warranted: the design is over the price range set by the customer.*

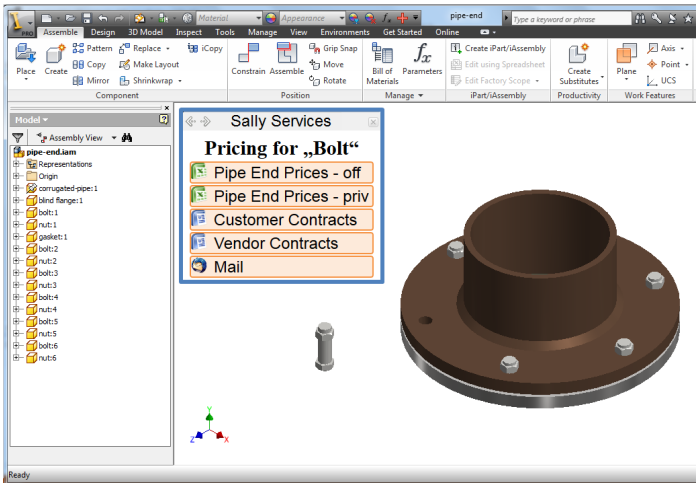
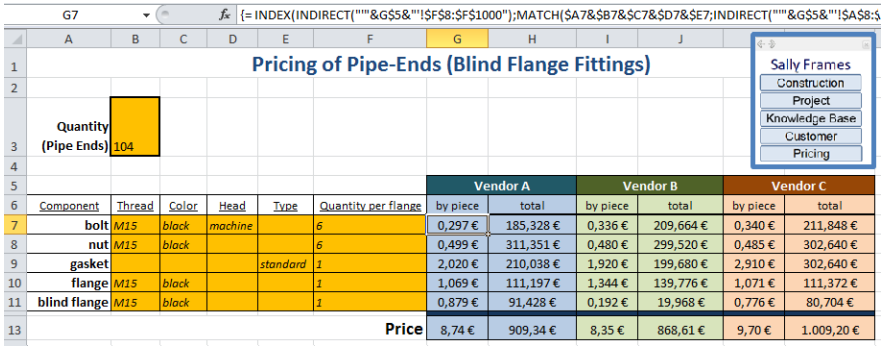


Fig. 7. Sally Services for Selected Bolt under Pricing Frame



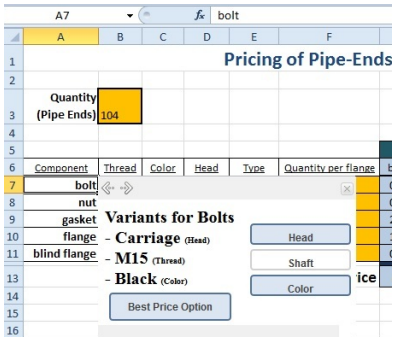
Component	Thread	Color	Head	Type	Quantity per flange	Vendor A by piece	Vendor A total	Vendor B by piece	Vendor B total	Vendor C by piece	Vendor C total
bolt	M15	black	machine		6	0,297 €	185,328 €	0,336 €	209,664 €	0,340 €	211,848 €
nut	M15	black			6	0,499 €	311,351 €	0,480 €	299,520 €	0,485 €	302,640 €
gasket				standard	1	2,020 €	210,038 €	1,920 €	199,680 €	2,910 €	302,640 €
flange	M15	black			1	1,069 €	111,197 €	1,344 €	139,776 €	1,071 €	111,372 €
blind flange	M15	black			1	0,879 €	91,428 €	0,192 €	19,968 €	0,776 €	80,704 €
					Price	8,74 €	909,34 €	8,35 €	868,61 €	9,70 €	1.009,20 €

Fig. 8. Pipe End Pricing with Sally Frames Menu

(a) **Frame Selection** *He tries to reduce the overall price by looking for alternatives for the pricey carriage bolts. Therefore, Charles now selects the frame “Knowledge Base”, since he needs information on homologous bolts, and clicks cell* [A7]. `Sally` in turn opens a window next to cell [A7] offering all available knowledge base services for “bolts”.

(b) **Service Selection** *Here, Charles chooses the “variants” service (see above).*

(c) **Variants Service Menu** The variants service uses a complex user interface of its own inside the `Sally`-originated UI window to determine the full specification of the bolts. In particular, `Sally` replaces the menu in the window by a list of bolt properties that can be altered without affecting functionality. *This list tells Charles that he can use arbitrarily colored bolts with arbitrary head types as long as he stays with the M15 thread.*



(d) **Direct Interaction** *This variants menu stays on top even though Charles now selects* [D7] *and modifies the head type to "machine",* which triggers the spreadsheet system to recalculate the numbers.

(e) **Variants Service Menu** *The price is already better, but he isn't yet satisfied, so he selects the color entry in the variants menu.* In consequence a temporary worksheet is created with a number of tables that display prices for the different color options for M15 machine bolts as in Fig. 9. *Alternatively, Charles could*

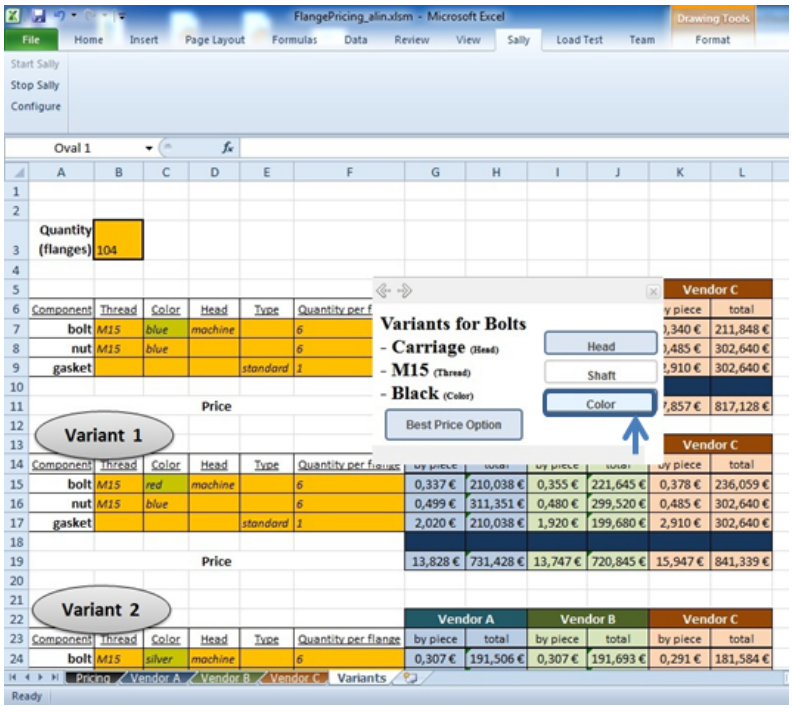


Fig. 9. Variants for Bolts in Terms of Color

have chosen the "Best price option", then the input data from the variant with the best price had replaced the original values.

We see that the interactions in the `Sally`-based interface are triggered by the interpretation mapping and the target concepts in the ontology. The advantage of this approach is that some aspects of the user interface can be declaratively specified in the ontology, but the `MA Semantic Alliance`-based approach also incurs requirements on the ontology and costs, which we will discuss next.

4.3 Requirements for the Ontology

The ontology has to include conceptualizations of the data models of the (information objects in the) applications themselves, e.g. concepts of CAD assemblies that are made of "parts" – which may be geometric objects like nuts or sub-assemblies themselves. Fig. 10 shows the taxonomic part of the ontology used in our example enriched with the "has part" relation (for the structure of assemblies). In our case, the ontology also contains the fact that a bolt is characterized by its head type, thread, and color.

Note that in the development of the ontology, we chose to only represent the object types, not the object instances themselves. For instance, the ontology only has the concept of a machine bolt (and its characteristic dimensions), but not the six individual instances in the pipe end assembly. Similarly, the ontology only specifies the notion of a volume pricing function of an object, but not the actual prices for the objects in the respective assemblies. The instance data stays in the applications themselves: the CAD system for the parts of the assemblies and the spreadsheet for the pricing information. This seems like a generally applicable heuristic for the development of engineering domain ontologies, as this information still justifies data import/export between frames: For the frame switch from the CAD frame to the pricing frame, we only need to know that the six bolts in the pipe end assembly co-reference the M15 machine bolt to export the data into the spreadsheet, which also references it in a functional block that takes the bolt characteristics as inputs.

4.4 Cost-Benefit Evaluation

The `MA Semantic Alliance` architecture consists of a network of semantic services that provide value, but need an interpretation mapping and background ontology to operate. Like any network-based approach, the potential benefits grow in proportion to the size of the network while the costs – often called **network joining costs** – grow with the size of the joining node. Our approach incurs three kinds of joining costs:

a) the applications have to be equipped with a `Semantic Alliance` API,
b) the domain has to be represented in a structured background ontology that specifies the semantic objects concerned, and
c) each document has to be equipped with an interpretation mapping into the ontology.

The costs for *a)* are almost negligible: [3] report that the effort required is in the order of developer weeks, which we can confirm for the API for `Autodesk Inventor` that was necessary for our example (the spreadsheet APIs already existed). Moreover, the API costs are only incurred once per application and are therefore domain-independent.

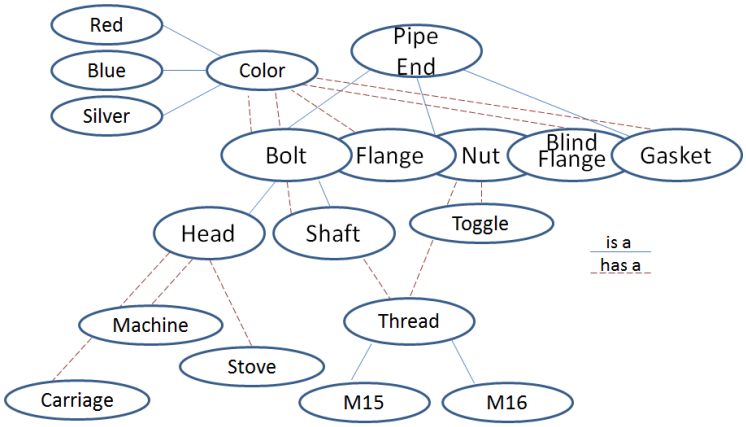


Fig. 10. A Simple Pipe End Ontology

The costs for *b)* are non-trivial, and are incurred by every useflow as long as it involves new semantic objects. But note that the costs for establishing an ontology are network costs, since we can assume the ontology to be shared, either globally or in the specific institution (in our example Charles' company). In our experience, large parts of the ontology concern very general topics (engineering parts, measurements, and legal transactions in our example), and can therefore be shared widely. In the future, they might even be outsourced to existing ontologies like DBpedia [4]. But of course, the useflow-specific parts of the ontology have to be hand-coded.

Finally, the costs for *c)* are document-specific, i.e., they are incurred by every document and service to be added to the useflow. Even though these costs can be somewhat mitigated by the interpretation mapping editors under development in the `Semantic Alliance` context, they have to be taken into account in the evaluation of the framework.

5 Conclusion

In this paper, we have addressed the problem of integrating complex useflows spanning multiple applications at the interface level. We have concentrated our investigation on applications that can be described as document players, where the objects addressed are represented in "documents" and the applications give the user access to these objects – "playing the documents", where each application is responsible for different aspects of the objects under consideration. We analyze the underlying problem as one of (lacking) full semantic transparency: Particular useflows are sense-making processes in which an information becomes sensible via frame (here: application), cue (here: information object) and connection (here: interpretation mapping), and switching between distinct applications force frame shifts with respect to the semantic object in the user's focus. In traditional useflows, the connection between the cues remains in the mind of the beholder and cannot be called upon to integrate the respective user interfaces.

We propose to solve that with (more) semantic technologies and materialize the connection – but not as a direct connection between information objects. Rather we make use of an existing technology that establishes a connection between the application's information objects and a semantic object represented in a structured background ontology and uses this mapping to mash semantic services into existing user applications, effectively turning them into user frames. We show that this architecture can be used for the task-switching and information integration requirements of user interface integration on a real-world engineering use-flow involving a CAD system for construction tasks and spreadsheets for calculations.

The point of this paper is not that user interfaces can be integrated at all, but that the integration can be achieved modularly, where new applications (and semantic services) can be integrated flexibly as the useflows warrant. Indeed most business applications opt for integration in a single system (see **U1** in the introduction) with the well-known lock-in problem for useflows. Our CAD system (`Autodesk Inventor`) already supports the integrations of spreadsheets, but this integration is at the data integration level only (**U2**).

In our semantic integration we can modify useflows easily by adding frames and services or replacing existing frames and services by compatible ones. For instance the spreadsheet-based pricing service in our example workflow could be replaced by a more scalable OLAP-based one if volumes and resources warrant.

Future Work The most significant shortcoming of the work presented here is that, even though our use cases and experiences with the system make it plausible that smooth application switching will enhance productivity in knowledge-intensive useflows, we have not subjected the system to evaluation with real users. We plan to do this once the system has stabilized further. Then we will also see whether our current assumption that semantic objects in the ontology are sufficient for pivoting between applications, or whether we need to extend pivoting to a two-step operation, which can utilize closely aligned semantic objects in the ontology. Our ontology representation already has candidates for such alignments, but we will have to see which ones can be used for pivoting – clearly not all ontology relations would be adequate.

We can improve the reach of our method and the implementation: For instance, one of the key ingredients of the solution proposed in this paper is the materialization of the connection between information objects across frames via interpretation mappings. This is also one of the largest cost factors in deploying it – assuming that the background ontologies will be shared and thus stabilize for a given domain. To reduce the costs we want to experiment with the powerful background ontology framework the `MA Semantic Alliance` employs. This provides a notion of "views" (mappings between semantic object representations) that may perhaps be used to materialize the semantically interesting parts of interpretation mappings into the ontology by representing information objects in the ontology and establishing ontology views to the semantic objects. This would have two consequences, the old interpretation mapping split into two parts: *a)* the (largely trivial) mapping between an information object in an application to its ontology representation and *b)* the intra-ontology view which can be shared by the resp. Community of Practice or the institution. In our experience, the necessary views are often already implicitly present in the ontology. Take for instance

the bolts in our running example: as they are physical objects, they have a time-space extent (their geometry, which is the aspect modeled in the CAD system) and they are possible goods in financial transactions (the aspect modeled in the pricing spreadsheet). Relying on such views made explicit in the ontology would simplify the interpretation mappings sufficiently that they may be amenable to heuristic methods for automating interpretation mapping creation.

The current implementation of the `MA Semantic Alliance` framework restricts the content of the RDF triple store to the contents of already open documents, which enables smooth application switching for a semantic object that is already in the user's focus. This might be sufficient for single-user situations like the one in our running example, but can be improved: If we imagine that `Sally` has access to a persistent triple store, then the collected abstract document ontologies can be used for discovery of relevant data sources (e.g. other pricing resources) and smooth application switching to them.

Acknowledgements We want to thank the anonymous reviewers for their constructive suggestions. This work has been funded by the German Research Council under grant KO-2484-12-1.

References

1. Hugh Beyer and Karen Holtzblatt. Contextual design. *ACM interactions*, 6(1):32–42, 1999. Jan/Feb.
2. R. Ian Bull. Integrating dynamic views using model driven development. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, CASCON '06, Riverton, NJ, USA, 2006. IBM Corp.
3. Catalin David, Constantin Jucovschi, Andrea Kohlhase, and Michael Kohlhase. `Semantic Alliance`: A framework for semantic allies. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics*, number 7362 in LNAI, pages 49–64. Springer Verlag, 2012.
4. DBpedia. `http://www.dbpedia.org`. seen Feb. 2012.
5. Brenda Dervin. Sense-making theory and practice: an overview of user interests in knowledge seeking and use. *Journal of Knowledge Management*, 2(2):36–46, 1998.
6. Oscar Díaz, Jon Iturrioz, and Arantza Irastorza. Improving portlet interoperability through deep annotation. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 372–381, New York, NY, USA, 2005. ACM.
7. Jon Iturrioz, Oscar Diaz, and Sergio F. Anzuola. Toward the semantic desktop: The semouse approach. *IEEE Intelligent Systems*, 23:24–31, 2008.
8. Andrea Kohlhase and Michael Kohlhase. Semantic transparency in user assistance systems. In Brad Mehlenbacher, Aristidis Protopsaltis, Ashley Williams, and Shaun Slatterey, editors, *Proceedings of the 27th annual ACM international conference on Design of communication (SIGDOC)*, pages 89–96, New York, NY, USA, 2009. ACM Special Interest Group for Design of Communication, ACM Press.
9. Jon Kolko. Sensemaking and framing: A theoretical reflection on perspective in design synthesis. In *Design Research Society Conference Proceedings 2010*, 2010.
10. D. B. le Roux and G. P. le Roux. People frames: the social construction of information systems. In *Proceedings of the 4th Symposium on Computer Human Interaction for the*

Management of Information Technology, CHiMiT '10, pages 1:1–1:9, New York, NY, USA, 2010. ACM.

11. Bonnie A. Nardi. *Studying context: a comparison of activity theory, situated action models, and distributed cognition*, pages 69–102. Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.
12. Chris North and Ben Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 128–135, New York, NY, USA, 2000. ACM.
13. Peter Pirolli. *Information Foraging Theory*. Oxford Series in Human-Technology Interaction. Oxford University Press, 2009.
14. Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C Recommendation, World Wide Web Consortium (W3C), January 2008.
15. L Sauermann, G A Grimnes, M Kiesel, C Fluit, H Maus, D Heim, D Nadeem, B Horak, and A Dengel. *Semantic Desktop 2.0: The Gnowsis Experience*, volume 6, pages 887–900. Springer, 2006.
16. Nikhil Sharma. Role of available and provided resources in sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1807–1816, New York, NY, USA, 2011. ACM.
17. Lucy A. Suchman. *Plans and Situated Actions: The problem of human machine communication*. Learning in Doing: Social, cognitive, and computational perspectives. Cambridge University Press, 1994 (1987).
18. Tania Tudorache, Natalya F. Noy, Sean M. Falconer, and Mark A. Musen. A knowledge base driven user interface for collaborative ontology development. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 411–414, New York, NY, USA, 2011. ACM.
19. K.E. Weick. *Sensemaking in Organizations*. Sage Publications Inc, 1995.