A Versioning and Commenting Approach for Enhancing Group Efficiency in Collaborative Web-Based Business Process Modeling Tools

Justus Holler
University of Muenster
ERCIS
Muenster, Germany
justus.holler@ercis.uni-muenster.de

***Abstract*—Business process modeling interpreted as a collaborative act requires resource intense communication and coordination between domain and modeling experts. Therefore, modern web-based business process modeling tools need to provide a shared workspace. Tool users can check and validate the progress of the business process modeling project and coordinate their work. This paper proposes a concept for workspace awareness, which combines version management with a commenting functionality in order to increase the groups modeling efficiency. The process model versions in the workspace from draft to final are documented with the help of a history-in-parent approach. Comments for each model version and comments within the models allow the tool users to collaborate asynchronously, which decreases the need for several workshop and interview iterations for process recording and refinement.**

Keywords-Business Process Modeling Tool; Versioning; Comments; Awareness; Modeling Efficiency.

I. INTRODUCTION AND MOTIVATION

Business Process Management (BPM) deals with managing, transforming and improving organizational operations [1]. One key part is the modeling of these operations in business process models. They are of high interest for public and private organizations as means to stay competitive and attractive in fast changing markets. A variety of methods, modeling languages and tools are used in these endeavors and there is research regarding the suitability of modeling languages for different purposes [2][3]. In every case, a modeling project basically depends on two parties. Firstly, the department worker who holds the knowledge of the organizational domain. Secondly, the modeling expert with particular sharp analytical and modeling skills who facilitates model creation [4-9].

Projects in this field are typically resource intense. This applies on the one hand to the time the department workers have to invest into the workshops or interviews and on the other hand to the budget for the (external) modeling expert who moderates the workshops or interviews and designs and refines the models afterwards [10][11]. Subsequently, the designed models have to be semantically validated by the department workers [6][7] and finalized in an iterative manner.

In order to reduce errors and to enhance the holistic understanding of the business processes within the scope of the project, it is at least beneficial if not necessary to include several department workers or domain experts into the business process modeling project [12][13]. Involving department workers in the process of modeling is reasonable as they accept [12][14] and understand the processes more thoroughly which fosters their critical evaluation and leads to potential process improvements [15]. Furthermore, the risk is reduced that the designed processes are not accepted by the department, which would lead to resource intense re-modeling or a failed project.

In its core, a business process management project with the deliverable of a documented process landscape is based on communication. Project participants contribute to a project outcome no single one of them could solely have accomplished. The necessary integration of the relevant project members can be done with the help of collaborative business process modeling tools [9][14]. In such tools, collaboration is possible but also coordination is necessary as the project participants have to distribute their tasks and synchronize their working objects [16–18] in a common context. Within this context awareness information helps to inform about the task status of other project members tasks and therefore reduces the coordination effort [19][20].

Hence, it is worthwhile to identify and foster drivers for improved integration and communication of the project members. The overall goal of this paper is to increase the project groups' efficiency in designing, reviewing and finalizing the models in the business process landscape. Therefore, this paper proposes a concept for workspace awareness in web-based business process modeling tools. This is done by the combination of a model versioning approach and an integrated commenting function.

Section 2 examines the need for versions in business process modeling projects and the role of comments as means for communication and coordination in information systems. The third Section explains the reasonable combination of model versioning and comments for improving awareness and presents the history-in-parent versioning approach and how it can be applied for a web-based process modeling tool. Section 4 discusses the conceptual design of the management and visualization of the commenting function based on the versioning approach and Section 5 concludes the paper and gives an outlook for future research.

II. Versions and Comments in Business Process Modeling Tools

In the context of modeling, it is difficult to define the quality of process models. Models can only be more or less useful with respect to their purpose [21]. In order to reach a status of usefulness – a final model – it needs the discussion of the good-willing and knowledgeable [22]. Moreover, once this status is reached, it has to be taken into account that organizations, markets and requirements change and that the process models have to be adjusted or re-evaluated on a continuous basis. In professional life in process modeling projects, this is done with IT support. For example, with a web-based process modeling tool which supports the project members in creation, modification, validation and communication of the process models in a collaborative manner. The project members access the models in an online repository in parallel and will either only view the model or also modify their content. The synchronous modification bares the risk of known issues like lost updates or inconsistencies within the models if not appropriate techniques are applied to reduce or prevent the risks. In theory this risk can be managed by displaying the models in real-time but it is questionable if this can be achieved and guaranteed in practical settings with tool users working distributed regarding time and place. Hence, (theoretical) modeling tools with the support for synchronous modeling are not applied in practice [23] and an analysis regarding twelve commercial available modeling tools showed that none of them supported real-time modeling [10].

Modeling in professional life is done in an asynchronous manner. The modeling experts design versions of the model, while the domain experts contribute business knowledge and validate the model versions either in workshops or directly within the tool. In team settings with several domain and modeling experts, there will be discussion regarding the most reasonable model and hence, the need for different versions of one process model. It takes several distinctive versions to reach the final, correct and useful state of one model.

For each new version from draft to the final model of the whole process landscape the reason why the new version was created, e.g., the input by the domain expert has to be saved. Hence, the necessity for textual descriptions or additional documents arises. This meta-information helps to understand what has changed or why something should be changed in a new version of the model. Therefore, comments enhance the understanding of the model users collaborating in one (virtual) workspace regarding the reasons for model changes and reduce the coordination effort [19].

III. Workspace Awareness with Comments and Version Management

People who work together in one room can easily recognize who is dealing with which part of the project. In a distributed or virtual setting, this is not possible. It needs additional information for efficient coordination of work items. Awareness information addresses the “what” and “where” [19]. In the case of comments, the comment itself is the answer to the “what” while the link to the commented object answers the “where”. Comments within process models therefore are particular well-suited regarding awareness, as their information automatically sets a context and the context for the process model element does not have to be set by a user explicitly. In groupware, awareness is a long known research field [24] and in the area of software engineering comments are used because of their awareness effects and are the basis for code awareness or code repository mining [25–27]. Although it is expected that the benefits of the above mentioned collaborative tasks can be transferred to the other collaborative tasks [20][28], there is no particular research in the area of collaborative process modeling with a focus on comments. Only [29] considers the commenting function as possible interface between domain and modeling experts. They can use comments directly in the process model (review comments) as known from software like Adobe Reader or Microsoft Word which allow a precise validation and correction and a dialogue between the users [20]. Domain experts can add information, which was not recorded during the interview or workshop. If the modeling expert creates new versions based on the input and wants to communicate the background for the new version, a management system has to be in place allowing comments (version comments) as meta-information for the specific versions.

A. *Version Management Systems*

The main purpose of version management systems in software engineering is the management of different versions of textual documents (source code) and their consolidation [30]. The most important version management systems like Concurrent Versions System (CVS), Subversion (SVN) and Git incorporate textual descriptions for the specific version or code which is a feature regularly used by software developers [27][31]. The developer does not have to analyze the source code in detail and still can estimate the impact of the changes for his or her work. The same applies to users of web-based business process modeling tools. Domain and modeling experts are aware of the changes made to one model version (version comment) or to a specific model element (review comment) by reading the textual description.

A hurdle to overcome for the application of version management concepts in web-based business process modeling tools is the document format in which process models are stored. Unlike source code in software engineering, process models cannot be compared on the basis of lines of text. Business process models are typically not stored in text files but in a binary format or in the best case in a serialized XML format [32]. Hence, process modeling tools which have an integrated version management functionality usually depend on their proprietary format [10].

The “History-in-Parent” version management approach [33] is proposed here which can directly be applied on a database level. Thus, there is no need for a serialization of the process models. The pre-condition for this approach is that the business process landscape can be represented as tree structure with the process framework as root node and the main and detail processes with their respective process elements as child nodes [34]. In principle, the level of abstraction layers is arbitrary but for readability reasons this paper limits the levels of abstraction to three. As only the history (old versions) is stored, the displayed process is always the most recent version of the model, which leads to good processing time in accessing the model in a web-based context. Also the processing time for executing basic functions as creating, modifying or deleting process elements (nodes) or restoring an old version is nearly optimal [33].

In Figure 1, an example structure is shown with a process framework (root node of the tree structure and 1st abstraction layer) after four modeling steps (t = 4) with links to a “1st main process” and “2nd main process” (both child nodes of the process framework). The “1st main process” is further detailed by a “1st detail process” and “2nd detail process”. Hence, the “1st main process” is the parent node for the two detail processes. For each layer of parent nodes (two in this example), a table is maintained, which stores the historic connections of the parent node to its child nodes. The “historic tables” are represented in Figure 1 as smaller circles in the same color and line style as the respective level of abstraction. The naming of the history nodes in the figures follows on every layer the same principle: The first part of the identifier represents the ID of the parent node (in the example the “Process framework” or “1st main process” with the ID 1). The second part of the identifier after the dot represents the time when the connections form parent to child nodes were valid.

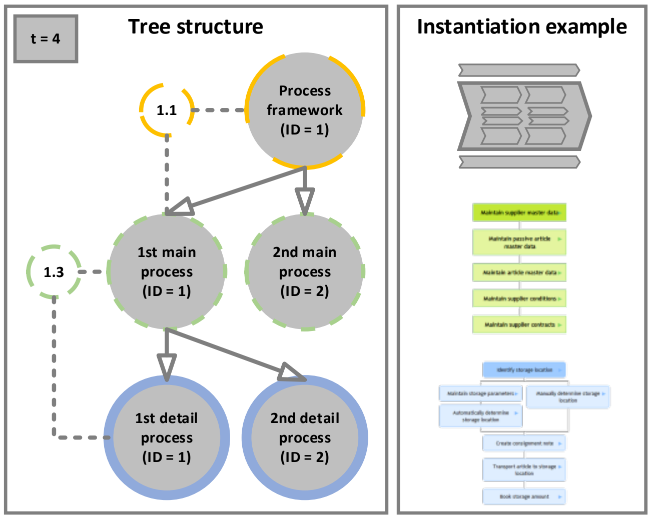

t = 4
Tree structure
1.1
Process framework (ID = 1)
1.3
1st main process (ID = 1)
2nd main process (ID = 2)
1st detail process (ID = 1)
2nd detail process (ID = 2)
Instantiation example

Figure 1. Tree structure and instantiation example of nodes.

Table 1 and Table 2 show the relevant data base tables for the main processes layer of the example. In Table 2, the historic information of the yellow history node can be found in the last row. For readability, the history table for the detail processes which would be analog to Table 2 is not printed and the visualization of technical attributes like primary and foreign keys are omitted in these and all following tables. Also, non-relevant history nodes are not displayed in the figures.

TABLE I. MAIN PROCESSES DATA BASE TABLE

Name	Node-ID	Parent-ID
1st main process	1	1
2nd main process	2	1

TABLE II. HISTORY TABLE ON THE PROCESS FRAMEWORK LEVEL

Framework-ID	Child nodes	Validity
1	NULL	0
1	1	1

B. Creating, Modifying, Deleting and Restoring Process Elements

A version management approach in a modeling tool has to track all relevant changes made to a model by the basic operations create, modify or delete. Furthermore, restoring old model versions has to be supported. In the following, the necessary steps for each of these operations are explained and an example is given in the next subsection.

Saving the links between parent node and its child nodes in the respective history table is the first step for all operations. Three attributes are of relevance: the ID from the parent node *k*, the IDs of the *child nodes*, which were linked to k and the version number (*validity*) before the change took place.

In case a node *o* is *modified*, its parent node is saved as described in the previous paragraph. Then, node *o* is copied as new node *k* with the same parent ID but the modified values (e.g., new identifier). The ID of the new node *k* is (auto-)incremented by the database system. Now, the parent node ID of *o* is set to NULL because *o* has no valid connection to a parent node in the most recent version of the model. Then *o* is saved and all links to *o* from *o*’s child nodes have to be changed to links to *k*.

If a new node k is *created*, first the parent node of k is saved so the old links are saved for the validity t – 1 and then the new node k can be created and is linked to the parent as new process element in the most recent model version.

The procedure of *deleting* a node k is done with the following steps: at first the parent node of k is saved, than the attribute parent node from k is set to NULL and k is saved. Finally, for all child nodes the links to k are set to NULL because k does not exist anymore in the most recent version of the process landscape.

Restoring a model version *x* is always done relatively to a node in the process landscape. The node (*v*) which shall be set back to version *x* is treated as root of a sub tree. This tree is then restored with the connections valid for the specified version (*x*): first, the parent node of *v* is saved (if existing) and afterwards *v* itself is saved. In a recursive step the original *o* of *v* valid for the time *x* has to be identified. Therefore, in the history table of *v* the most recent node *h* has to be selected where *v* is not listed as child node. Three cases are possible: If no node *h* is found, then, there was no change to *v* since version *x*. If *h* is found and the next historic child node *h+1* in the table has the same amount of child nodes as *h*, than the original *o* can be derived by the comparison of the child nodes from *h* and *h+1*. In case *h+1* has only one more child node, than *v* is not the result of a modification but the original itself and the recursion can stop. After going through the recursion at least twice the value for the parent node of *v* is copied to *o* and the parent node of *v* is set NULL, which makes *v* to the new original. All child nodes of *v* have to be saved and their parent node attribute is set to NULL. Now, search for the newest historic node *h* of *v* in the historic table of the same level like *v* which version is smaller or identical to *x* and set the value for the parent node of all in *h* listed child nodes to *v*. For clarification of the above mentioned steps, all operations are used in the following modeling scenario.

C. *Modeling an Example Process Landscape*

In the first step (t = 0), a new process framework "Whole sale" is created (Figure 2).

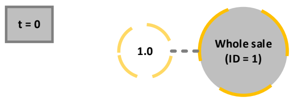


Figure 2. Process landscape (t = 0).

After creating the main processes contracting (t = 1), purchasing (t = 2) and receiving (t = 3) and the detail processes maintain article master data (t = 4), maintain supplier master data (t = 5) and maintain supplier contracts (t = 6) the process landscape in its tree representation looks like depicted in Figure 3 with the process element nodes and the respective history nodes.

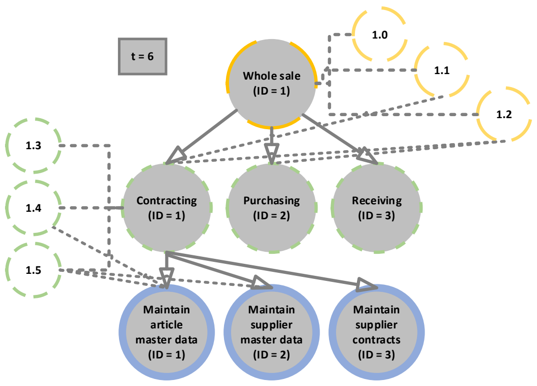


Figure 3. Process landscape after adding main and detail processes (t = 6).

Now the main process element "Contracting" is renamed to "Contrac management". The renamed element is linked to the parent node while the node with the old name ("Contracting") has no link to a parent node anymore (Figure 4, t = 7).

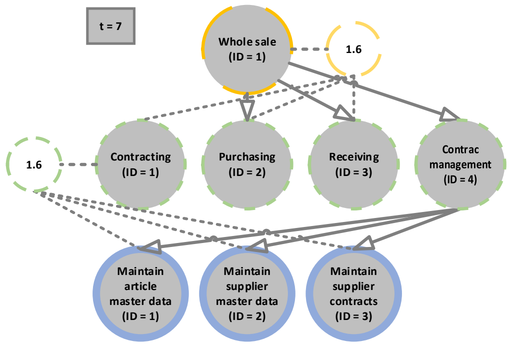


Figure 4. Process landscape after renaming Contracting (t = 7).

TABLE III. MAIN PROCESSES AFTER CHANGING "CONTRACTING" TO "CONTRAC MANAGEMENT" (T = 7)

Name	ID	Framework-ID
Contracting	1	NULL
Purchasing	2	1
Receiving	3	1
Contrac management	4	1

Due to the name change a new entry in the table of the main processes (Table 3) is added (ID = 4) with a new connection to the old parent of "Contracting". The old connection, which was valid for t = 6 is stored in the history table (Table 4).

TABLE IV. HISTORY TABLE ON FRAMEWORK LEVEL AFTER CHANGING "CONTRACTING" (T = 7)

ID	Child nodes	Validity
1	NULL	0
1	1	1
1	1,2	2
1	1,2,3	6

Table 5 lists all detail processes including the ID of their parent nodes. In t = 7, the parent node ID for the detail processes is the same as they all have the same parent node which has changed from ID 1 ("Contracting") to ID 4 ("Contrac Management").

TABLE V. DETAIL PROCESSES DATA BASE TABLE (T = 7)

Name	ID	Main proc.-ID
Maintain article master data	1	4
Maintain supplier master data	2	4
Maintain supplier contracts	3	4

The version information regarding the old parent ID is stored in the historic table on the main process level as this is the parent-level of the detail processes (Table 6).

TABLE VI. HISTORIC TABLE OF THE MAIN PROCESSES (T = 7)

Main proc.-ID	Child nodes	Validity
1	NULL	3
1	1	4
1	1, 2	5
1	1, 2, 3	6

The project manager decides that “Receiving” is not in scope anymore and deletes it (t = 8). Furthermore, the domain expert wants the modeler to correct the typo in “Contrac management” (t = 9) and to consider the business case of canceling a contract. Because this procedure does not fit into the existing “Maintain supplier contracts” the modeler decides to add a new detail process (t = 10, Figure 5).

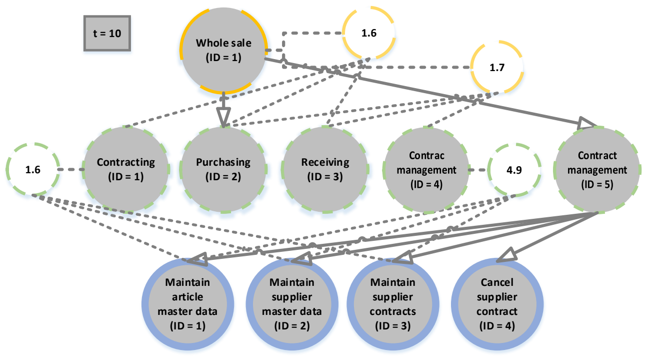

t = 10
Whole sale (ID = 1)
1.6
1.7
1.6
Contracting (ID = 1)
Purchasing (ID = 2)
Receiving (ID = 3)
Contrac management (ID = 4)
4.9
Contract management (ID = 5)
Maintain article master data (ID = 1)
Maintain supplier master data (ID = 2)
Maintain supplier contracts (ID = 3)
Cancel supplier contract (ID = 4)

Figure 5. Process landscape after deleting “Receiving”, correcting “Contract management” and adding “Cancel supplier contract” (t = 10).

In Table 7 the main processes information is stored after deleting the process “Receiving” (Framework-ID = NULL) and correcting the typo in “Contract management”.

The deletion of “Receiving” leads to a new entry in the history table (Table 8), where the link from Framework-ID = 1 to its Child node ID = 3 is not listed anymore.

TABLE VII. TABLE OF MAIN PROCESSES AFTER DELETING “RECEIVING” AND CORRECTING THE TYPO (T = 10)

Name	ID	Framework-ID
Contracting	1	NULL
Purchasing	2	1
Receiving	3	NULL
Contrac Management	4	NULL
Contract management	5	1

TABLE VIII. HISTORY TABLE AFTER DELETING “RECEIVING” (T = 10)

Framework-ID	Child nodes	Validity
1	NULL	0
1	1	1
1	1, 2	2
1	1, 2, 3	6
1	2, 3, 4	7
1	2, 4	8

In Table 9 the detail processes valid for t = 10 are listed with the newly added process “Cancel supplier contract”. The version information are stored in the history table (Table 10) on main process level (parent level of detail processes). The newest validity entry 9, as this refers to the typo correction. The most recent version with the just added detail process is – as always – not stored in this table.

TABLE IX. TABLE OF DETAIL PROCESSES AFTER ADDING “CANCEL SUPPLIER CONTRACT” (T = 10)

Name	ID	Main proc.-ID
Maintain article master data	1	5
Maintain supplier master data	2	5
Maintain supplier contracts	3	5
Cancel supplier contract	4	5

TABLE X. HISTORIC TABLE OF MAIN PROCESSES (T = 10)

Main proc.-ID	Child nodes	Validity
1	NULL	3
1	1	4
1	1, 2	5
1	1, 2, 3	6
3	NULL	7
4	1, 2, 3	8
5	1, 2, 3	9

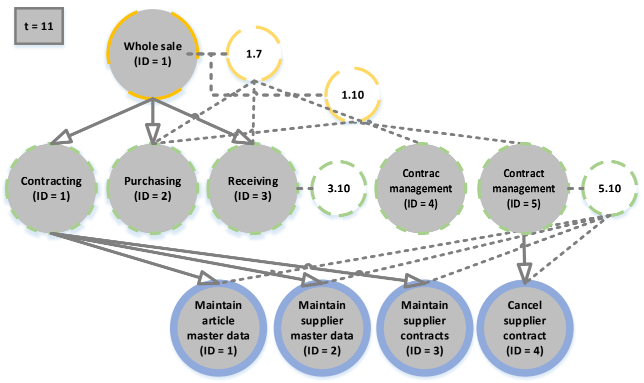

t = 11
Whole sale (ID = 1)
1.7
1.10
Contracting (ID = 1)
Purchasing (ID = 2)
Receiving (ID = 3)
3.10
Contrac management (ID = 4)
Contract management (ID = 5)
5.10
Maintain article master data (ID = 1)
Maintain supplier master data (ID = 2)
Maintain supplier contracts (ID = 3)
Cancel supplier contract (ID = 4)

Figure 6. Process landscape after the status of t = 6 is restored.

In the final presentation of the process landscape (t = 11), the project team decides to restore the version of the process framework which was valid in t = 6. The resulting tree with “Whole sale” as root (Figure 6) is the same as it was in Figure 3.

IV. CONCEPTUAL DESIGN OF THE COMMENT FUNCTIONALITY

The described version management approach from the last section is one technical necessity for the project team to model the process landscape collaboratively. The other important part is the addition of comments at appropriate places. In the sense of [35] the comments are the messages which are send from a transmitter (domain expert) over a channel (modeling tool) to a receiver (modeling expert) and vice versa. This form of communication enables the good-willing and knowledgeable experts [22] to coordinate their activities and efficiently collaborate.

In the following the concept for storing the comments from the technical perspective is described and the possibilities of communicating their message via the modeling tool.

A. Storing the Comments in the Data Base

All comments regarding the process landscape are stored in one comment table (Figure 7, [36]). The link to the specific model (element) is done via: a *comment-ID* identifies the specific comment, an *element-ID* sets the link to the process element or process which is commented (the process element table is omitted for readability reasons), an *author-ID*, an *addressee-ID* can be linked to a specific user, an *attachment* attribute allows a link to additional information, a *timestamp* allows to sort the comments, a *type* indicates if it is a version, review or change request comment and the attribute *comment* contains the textual comment itself. The hierarchical relationship of the comment entity (*comment hierarchy*) allows users to write comments in relation to another comment and therefore create answers or threads.

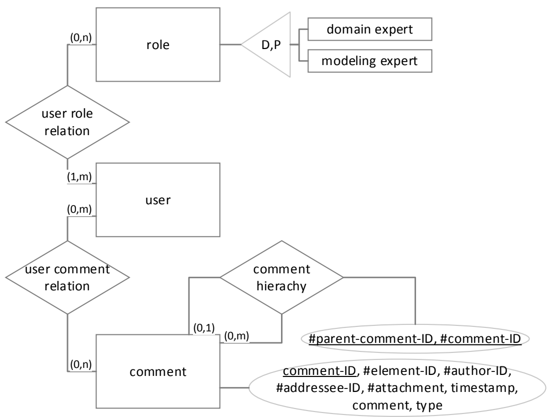


Figure 7. Entity relationship model of comment, user and role table.

Between the *comments* and the *users* a n:m-relationship is proposed. This allows a read / unread functionality for each user separately as known from e-mail clients.

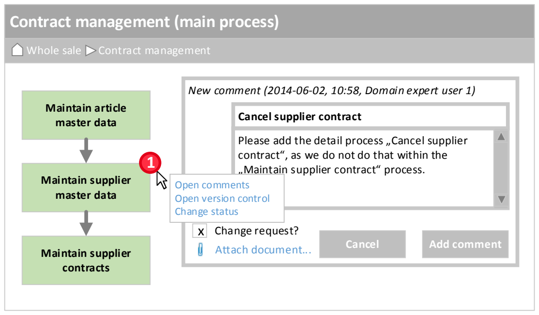
The process and process element tables are not depicted in the Entity relationship model (ERM, [36]), but have an important status attribute. Each instance on framework, main and detail process level has a status indicating if the element is in work, a draft, ready for review or final. Depending on this status, the domain expert can validate the overall structure (draft), check for semantic correctness (ready for review) or set the status of the process (element) to final.

These status are coupled with the role concept. As in the modeling project domain and modeling experts are involved it has to ensure, that only authorized persons can set the appropriate status. For example, the modeler can set the "in work", "draft" and "ready for review" status while only the domain expert can set the status to "final" and therefore validate its semantic correctness from the subjective point of view and if it is fit for use [21].

B. Communication of the Comments

In order to create workspace awareness for the project members it is crucial to communicate the committed changes, their textual descriptions (version and review comments) and the status of each comment. Hence, the comments have to be *visualized*, and *managed* by the modeling tool.

In Figure 8 a domain expert views the main process contract management and has one unread comment regarding the process step "Maintain supplier master data". The amount of unread comments is visualized with the help of a bubble. It can either relate to comments regarding the process element itself or it can indicate the amount of comments underneath the process element. If this bubble or the process element is right-clicked the user can navigate to the comments view (Figure 9), switch to the version management view (Figure 10) or change the process status. Furthermore, in Figure 8 the domain expert writes a change request regarding the missing process step "Cancel supplier contract".


Contract management (main process)

Whole sale ▷ Contract management

New comment (2014-06-02, 10:58, Domain expert user 1)

Cancel supplier contract

Please add the detail process „Cancel supplier contract", as we do not do that within the „Maintain supplier contract" process.

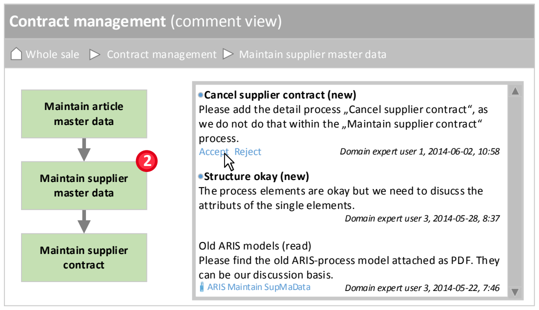
x Change request?

Attach document...

Maintain article master data
Maintain supplier master data
Maintain supplier contracts
1
Open comments
Open version control
Change status
Cancel
Add comment

Figure 8. Adding a comment and right-click effect (domain expert role, t = 9).

While *viewing* the comments in the process model the user can mark the comment as read and depending on the comment and the users' role, it is possible, to "accept" or "reject" a change request. By this means, a domain expert can track if the modeling expert already read the comment and if her change request was accepted and possibly send the modeling expert a reminder. Furthermore, the read / unread functionality helps to filter the visible information: all read and accepted or rejected comments are hidden in the default visualization of the process model for the specific tool user. If there is no new comment, there would be no red bubble in the process model visible. In Figure 9 the bubble indicates the two new comments. In the comment view the comments are displayed in a discussion like style and additional attached documents can easily be accessed by the user [28].


Contract management (comment view)

Whole sale ▷ Contract management ▷ Maintain supplier master data

Cancel supplier contract (new)
Please add the detail process „Cancel supplier contract", as we do not do that within the „Maintain supplier contract" process.
Accept Reject *Domain expert user 1, 2014-06-02, 10:58*

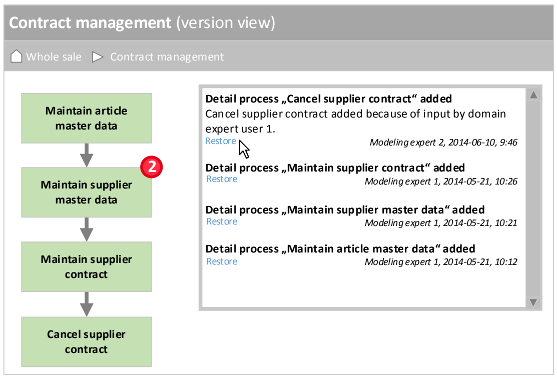
Structure okay (new)
The process elements are okay but we need to disucss the attributs of the single elements.
Domain expert user 3, 2014-05-28, 8:37

Old ARIS models (read)
Please find the old ARIS-process model attached as PDF. They can be our discussion basis.
ARIS Maintain SupMaData *Domain expert user 3, 2014-05-22, 7:46*

Maintain article master data
Maintain supplier master data
Maintain supplier contract
2

Figure 9. Comment view (modeling expert role, t = 9).

In Figure 10 the detail process "Cancel supplier contract" was added which leads to a new version with an explicit version comment by the modeling expert 2 user. Furthermore, the authorized user can restore old versions of the model from here.


Contract management (version view)
Whole sale
Contract management
Maintain article master data
Maintain supplier master data
2
Maintain supplier contract
Cancel supplier contract

Detail process „Cancel supplier contract" added
Cancel supplier contract added because of input by domain expert user 1.
Restore *Modeling expert 2, 2014-06-10, 9:46*

Detail process „Maintain supplier contract" added
Restore *Modeling expert 1, 2014-05-21, 10:26*

Detail process „Maintain supplier master data" added
Restore *Modeling expert 1, 2014-05-21, 10:21*

Detail process „Maintain article master data" added
Restore *Modeling expert 1, 2014-05-21, 10:12*

Figure 10. Version view (modeling expert role, t = 10).

Typically, the management of the comments is done in a passive manner and relies on the project member to log. After the user logs in, a dashboard indicates about activities in the modeling project with a list of recently changed models and new comments. However, for a domain expert, working fulltime in the department, this is problematic as their day to day business does not allow visiting the tool on a regular basis. Hence, for the efficient collaboration between the domain and modeling experts, the comments regarding changes have to be communicated also actively. This is achieved by sending requests for comments or requests for validation via e-mail. In addition, reports on a regular basis summing up all changes and indicating where new comments were made are a possibility of increasing the awareness.

V. Conclusion and Future Work

This paper presented an approach for awareness enhancement in web-based business process modeling tools. The basis for this is a version management instantiation of the history-in-parent approach, which directly works on the database tables. The model versions can be commented and the modeling team is informed about the changes. As the goal of the approach is to foster the efficiency in the collaborative act of modeling the approach defines active (e-mail notification) and passive awareness (indicators for changes in the tool) functionality. Thus, the approach is striving for supporting domain and modeling experts likewise by reducing coordination effort, which leads to savings in time and therefore overall more efficient modeling.

In future research, this approach will be implemented in a web-based business process modeling tool for evaluation. The data base performance over time and feature completeness will be evaluated. Especially the medium (e.g., e-mail, dashboard), frequency (directly vs. summarized) and mode (active vs. passive) of awareness information will be evaluated.

Also, the limitations, e.g., the technical limitation regarding the danger of inconsistencies in case of concurrent modeling will be addressed. One possibility of addressing this can be a locking mechanism. The evaluation will show to which extent this lock hinders the collaborative work, as restricted to the approach all models underneath the model in use would be locked for other modelers.

Furthermore, the evaluation in laboratory settings with students and afterwards in consultancy projects will reveal the appropriateness of the comments. As no user can be forced to actively read or write comments, there is a natural limitation in this approach. It is likely that convenience in writing the comments like templates or auto-completion, similar to the popular version management control systems, will foster comment quality.

References

[1] M. Hammer and J. Champy, "Reengineering the Corporation: a Manifesto for business Revolution," Bus. Horiz., vol. 36, no. 5, pp. 90–91, 1993.

[2] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell, "On the suitability of BPMN for business process modelling," in Business Process Management, Springer, 2006, pp. 161–176.

[3] R. S. Aguilar-Savén, "Business process modelling: Review and framework," Int. J. Prod. Econ., vol. 90, no. 2, pp. 129–149, Jul. 2004.

[4] M. Pendergast, K. Aytes, and J. D. Lee, "Supporting the group creation of formal and informal graphics during business process modeling," Interact. Comput., vol. 11, no. 4, pp. 355–373, 1999.

[5] D. Dean, R. Orwig, and D. Vogel, "Facilitation methods for collaborative modeling tools," Gr. Decis. Negot., vol. 9, no. 2, pp. 109–128, 2000.

[6] P. J. M. Frederiks and T. P. Van der Weide, "Information modeling: the process and the required competencies of its participants," Data Knowl. Eng., vol. 58, no. 1, pp. 4–20, 2006.

[7] M. Weske, Business Process Management. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[8] I. Wilmont, E. Barendsen, S. Hoppenbrouwers, and S. Hengeveld, "Abstract Reasoning in Collaborative Modeling," 2012 45th Hawaii Int. Conf. Syst. Sci., pp. 170–179, Jan. 2012.

[9] P. Rittgen, "Collaborative Modeling - A Design Science Approach," in HICSS '09. 42nd Hawaii International Conference, 2009, pp. 1–10.

[10] K. Riemer, J. Holler, and M. Indulska, "Collaborative Process Modelling - Tool Analysis and Design Implications," in 19th European Conference on Information Systems (ECIS), Paper 39, 2011.

[11] J. Jeston and J. Nelis, Business process management. Routledge, 2014.

[12] D. Dean, R. Orwig, J. Lee, and D. Vogel, "Modeling with a group modeling tool: group support, model quality, and validation," in Twenty-Seventh Hawaii International Conference, 1994, vol. 4, pp. 214–223.

[13] T. Dollmann, C. Houy, P. Fettke, and P. Loos, "Collaborative Business Process Modeling with CoMoMod - A Toolkit for Model Integration in Distributed Cooperation Environments," 2011 IEEE 20th Int. Work. Enabling Technol. Infrastruct. Collab. Enterp., pp. 217–222, Jun. 2011.

[14] P. Rittgen, "Collaborative modeling of business processes: a comparative case study," in Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 225–230.

[15] F. M. Santoro, M. R. S. Borges, and J. A. Pino, “CEPE: cooperative editor for processes elicitation,” in 33rd Hawaii International Conference on System Sciences, pp. 10 vol 1, 2000.

[16] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning, and L. Suchman, “Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings,” Commun. ACM, vol. 30, no. 1, pp. 32–47, 1987.

[17] S. R. Fusseu, R. E. Kraut, F. J. Lerch, W. L. Scherlis, M. M. McNally, and J. J. Cadiz, “Coordination, Overload and Team Performance: Effects of Team Communication Strategies,” in Proceedings of the 1998 ACM conference on Computer supported cooperative work, 1998, pp. 275–284.

[18] C. A. Ellis, S. J. Gibbs, and G. L. Rein, “Groupware - Some Issues and Experiences,” Commun. ACM, vol. 34, no. l, pp. 39–58, 1991.

[19] C. Gutwin, S. Greenberg, and M. Roseman, “Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation,” People Comput. XI, pp. 281–298, 1996.

[20] P. Dourish and V. Bellotti, “Awareness and coordination in shared workspaces,” 1992, pp. 107–114.

[21] J. Becker, M. Rosemann, and C. von Uthmann, “Guidelines of Business Process Modeling,” in Business Process Management: Models, Techniques and Empirical Studies, W. van der Aalst, J. Desel, and A. Overweis, Eds. Berlin et al.: Springer, 2000, pp. 30–49.

[22] W. Kamlah and P. Lorenzen, Logische Propädeutik: Vorschule des vernünftigen Redens, 3rd ed. Stuttgart: Metzler, 1996.

[23] J. Mendling, J. Recker, and J. Wolf, “Collaboration Features in current BPM Tools,” EMISA Forum, vol. 32, no. 1, pp. 48–65, 2012.

[24] T. Gross, “Supporting Effortless Coordination: 25 Years of Awareness Research,” Comput. Support. Coop. Work, vol. 22, no. 4–6, pp. 425–474, Jun. 2013.

[25] C. Gutwin, R. Penner, and K. Schneider, “Group awareness in distributed software development,” in Proceedings of the 2004 ACM conference on Computer supported cooperative work - CSCW ’04, 2004, pp. 72–81.

[26] H. Kagdi, M. L. Collard, and J. I. Maletic, “A survey and taxonomy of approaches for mining software repositories in the context of software evolution,” pp. 77–131, 2007.

[27] A. Chen, E. Chou, J. Wong, A. Y. Yao, and A. Michail, “CVSSearch: searching through source code using CVS comments,” Proc. IEEE Int. Conf. Softw. Maintenance. ICSM 2001, pp. 364–373, 2001.

[28] C. M. Neuwirth, D. S. Kaufer, R. Chandhok, and J. H. Morris, “Issues in the Design of Computer Support for and Commenting,” in CSCW ’90, 1990, no. October, pp. 183–195.

[29] G. Decker and M. Weske, “Towards Collaborative Business Process Modeling,” Cut. IT J., 2009.

[30] W. Tichy, “RCS—a system for version control,” Softw. Pract. Exp., vol. 7, no. July 1985, pp. 637–654, 1985.

[31] Y. S. Maarek, D. M. Berry, and G. E. Kaiser, “An Information Retrieval Approach For Automatically Constructing Software Libraries,” IEEE Trans. Softw. Eng., vol. 17, no. 8, pp. 800–813, 1991.

[32] N. Clever, J. Holler, J. Püster, and M. Shitkova, “Growing Trees – A Versioning Approach for Business Process Models based on Graph Theory,” in Proceedings of the European Conference on Information Systems (ECIS) 2013, Paper 157, 2013.

[33] E. J. Choi and Y. R. Kwon, “An efficient method for version control of a tree data structure,” Softw. Pract. Exp., vol. 27, no. 7, pp. 797–811, 1997.

[34] R. Tarjan, “Depth-First Search and Linear Graph Algorithms,” SIAM J. Comput., vol. 1, no. 2, pp. 146–160, Jun. 1972.

[35] C. E. Shannon, “A Mathematical Theory of Communication, Part I,” Bell Syst. Tech. J., vol. 27, pp. 379–423, 1948.

[36] P. P.-S. Chen, “The Entity-Relationship Unified View of Data Model-Toward a,” ACM Trans. Database Syst., vol. 1, no. 1, pp. 9–36, 1976.