MASTER OF SCIENCE – THESIS IN BUSINESS ADMINISTRATION

Improving Flowmanager

Providing recommendations on the improvement of workflow commissioning and semantic checking

Jens Rothman

April 8, 2015

UNIVERSITY OF TWENTE.

UNIVERSITY OF TWENTE.

Improving Flowmanager - Providing recommendations on the improvement of workflow commissioning and semantic checking

MASTER OF SCIENCE - THESIS IN BUSINESS ADMINISTRATION

Author

Name:	Herrald Jan (Jens) Rothman
Student number:	0208183
Contact:	jensrothman@gmail.com
Programme:	Master Business Administration
Track:	Information Management
Faculty:	Behavioural, Management and Social sciences

Supervisors University of Twente

Name:	Prof. Dr. J. (Jos) van Hillegersberg
Faculty:	Industrial Engineering and Business Information Systems
Department:	Information Systems and Change Management

Name:	Dr. M.E. (Maria-Eugenia) Iacob
Faculty:	Industrial Engineering and Business Information Systems
Department:	Information Systems and Change Management

Management Summary

This is a public version of this thesis. Due to confidentiality, the name of the company involved has been changed to BuildIT and the name of the product has been changed to Flowmanager. Also, some parts of the thesis are marked as confidential and therefore are not included in this public version.

A workflow management system (WfMS) is an information system which controls transactions between activities within a workflow (Stohr & Zhao, 2001). WfMS's are often used by several parties such as banks and insurers to manage the difficult financial processes these companies encounter. BuildIT is looking to improve their own WfMS called Flowmanager. More specifically, BuildIT wants to know how they can improve the workflow commissioning and semantic checking properties of Flowmanager. This request has been the basis on which this thesis is built and has led to the following research question:

How can workflow commissioning and semantic checking in Flowmanager be improved to meet the demands of clients, by applying best practices and standards derived from the theory?

In order to answer this question a literature review and a Delphi panel were conducted. The literature has shown that the field of business rules can be divided in two categories, as stated by Iacob and Jonkers (2009). The first category focuses on operational process rules and entails the topic of workflow commissioning. The second category is about constraints and includes the topic of semantic checking.

Based on these findings, two literature frameworks have been formed for workflow commissioning and semantic checking. For workflow commissioning, the article by van der Aalst, ter Hofstede, Kiepuszewski, and Barros (2003) has been used to create a foundation for the Delphi panel. For semantic checking, the same has been done with the article by Ly, Rinderle, and Dadam (2008).

Based on these literary findings, a Delphi panel has been conducted among 11 experts. These experts are involved daily with Flowmanager´s workflow commissioning options. In three rounds, the experts have created a prioritization for workflow commissioning additions and have indicated the need for semantic checks in several situations.

This thesis concludes its research by discussing the results of the Delphi panel and placing them in the perspective of Flowmanager. For workflow commissioning, this thesis concludes that five workflow patterns should be developed and introduced in Flowmanager in the following order: Multi choice combined with Synchronizing merge, Parallel split combined with Synchronization and Multi merge. For semantic checking, it is concluded that due to the gap between theory and the practice of Flowmanager, and the technical nature of this topic which remains underexposed in this thesis, no direct recommendations can be made for the implementation of semantic checking in Flowmanager. However, semantic checking does play an important role in the introduction of Multi choice and Synchronizing merge in Flowmanager. These two patterns introduce flexibility that is new to Flowmanager. In order to properly manage the implementation of Multi choice and Synchronizing merge in Flowmanager, the implementation should be accompanied by the introduction of the principles of workflow templates and instances by Ly et al. (2008).

Finally, two suggestions for future academic research are made. First, it is suggested that for both workflow commissioning and semantic checking research needs to be done on the implementation of these theoretical concepts into practice. This can lead to the development of a roadmap for the implementation of new workflow patterns and semantic checking. Second, it is suggested to research the presence of the methodology by Ly et al. (2008) in other Workflow Management Systems. This

research assesses the practical relevance of the theory and can provide guidance for further introduction into Flowmanager.

Acknowledgements

In front of you lies the thesis titled “Improving Flowmanager - Providing recommendations on the improvement of workflow commissioning and semantic checking”. This thesis is the final assignment of the Master program Business Administration at the University of Twente. I have created this thesis in the period from September 2014 up to and including February 2015.

I have written this thesis at the request of BuildIT B.V. The exact assignment outline has been created in association with my external supervisors. At this point I would like to show my gratitude towards them for their constant advice, feedback and assistance throughout the entire process the last six months.

I would also like to thank my supervisors at the University of Twente, Jos van Hillegersberg and Maria Iacob. Jos has provided me with a lot of useful feedback which improved the overall quality and scope of my thesis. Maria has really assisted me in structuring the thesis and improving the literature review. Both of the supervisors were always available for questions, either at their offices or online through Skype. Also, I would like to thank Ton Spil for his feedback on my research proposal, helping me narrowing down my research project.

The performed research has proven to be challenging due to the technical aspect, which was pretty far out of my comfort zone as a Business Administration student. Nonetheless, with the guidance of Jos and Maria on the research topics, I managed to gain a lot of knowledge on the topics as discussed in this thesis.

I would like to thank Spilter for letting me use their decision making software free of charge. The software saved me a lot of time by providing a ready to use format for the Delphi panel.

Finally, I would like to thank BuildIT and its employees for providing me with a great environment in which I could execute my thesis. I was working between the other employees and really had the opportunity to see what it is like to work at BuildIT. Also, several employees have assisted me in the selection of relevant participants among the clients of BuildIT. From day one I felt at home and valued by BuildIT.

Jens Rothman

Hengelo, 23rd of February 2015

Table of contents

1. Introduction

This is a public version of this thesis. Due to confidentiality, the name of the company involved has been changed to BuildIT and the name of the product has been changed to Flowmanager. Also, some parts of the thesis are marked as confidential and therefore are not included in this public version.

This chapter provides an introduction to several subjects which are essential to a complete understanding of the context in which this thesis is completed. Firstly, a comprehensive overview of BuildIT, the Flowmanager platform, workflow commissioning and semantic checking is created. Secondly, the assignment on which this thesis is based, is described. Thirdly, the goal, research question and sub question are portrayed. Fourthly, the methods used in this thesis are briefly described. Fifthly, the relevance of this thesis is discussed. Finally, the outline for the rest of this thesis is provided.

1.1. About BuildIT, Flowmanager, workflow commissioning and semantic checking

1.1.1. About BuildIT

This part of the thesis has been marked as confidential.

1.1.2. About Flowmanager

This part of the thesis has been marked as confidential.

1.1.3. About workflow commissioning and semantic checking

This part of the thesis has been marked as confidential.

1.2. Problem Description

BuildIT regularly gets requests from clients for improvements of the functionalities of Flowmanager, including additions to workflow commissioning and semantic checks. These request vary from client to client, depending on their needs and their usage of Flowmanager. This variety makes it difficult for BuildIT to determine which improvement of Flowmanager should receive priority. It is also unclear whether additions of Flowmanager are supported amongst multiple clients, since different departments of BuildIT each communicate with their own client. Taking these issues and the expenses of developing these new features into account, there is a clear need of guidance and structural decision making in the development of new workflow commissioning features.

The second issue which is addressed in this thesis, is the development of semantic checking in the workflow environment of Flowmanager. At this point, no functionality of semantic checking exists in Flowmanager, although it is a much desired feature, both by clients and BuildIT. With semantic checking, the workflow commissioning capabilities of clients improves and BuildIT has a more complete product to sell. Before BuildIT can initiate the development and implementation of semantic checking in Flowmanager, it is necessary to know in which situations semantic checks can provide added value. Researching this issue creates a scope for the development of semantic checking tools.

1.3. Goal, research question and sub questions

Based on the provided information and the problem description in paragraph 1.2, the goal, research question and sub questions for this thesis have been created, as described in this paragraph.

1.3.1. Goal

Summarizing these research topics, the goal of this research is *advising on the improvement of workflow commissioning and semantic checking in Flowmanager by applying best practices and standards derived from theory and aligning them with the demands of BuildIT' clients.*

1.3.2. Research question and sub questions

The described goal can be translated to the following research question:

How can workflow commissioning and semantic checking in Flowmanager be improved to meet the demands of clients, by applying best practices and standards derived from the theory?

In order to answer the research question, the following sub-questions are addressed:

1. How does Flowmanager work? Explaining the platform, its workflow and semantic checking.
2. What are the best practices of workflow commissioning and semantic checking?
3. How does the workflow commissioning and semantic checking of Flowmanager compare with the best practices, what are possible improvements?
4. What is the opinion of clients about the possible improvements of workflow commissioning and semantic checking in the context of Flowmanager?
5. How do possible improvements of workflow commissioning and semantic checking in Flowmanager, derived from the theory, compare with the opinions of clients?
6. Which found improvements of Flowmanager have the highest priority of implementation?

1.4 Methodologies used in Thesis

1.4.1. Structure method

The structure of this thesis has been created based on the Design Science Research Methodology (DSRM) as provided by Peffers, Tuunanen, Rothenberger, and Chatterjee (2007). In their paper, Peffers et al. (2007) emphasize the need for a widely held "*methodology for conducting design science research*" in the field of Information Systems. As a response to this need Peffers et al. (2007) present DSRM, a commonly accepted method of executing design science in the field of Information Systems. Peffers et al. (2007) make a distinction between four types of research in the field of IS. This thesis is classified as an Objective-centered solution, since the incentive for this thesis is based on a need coming from BuildIT.

DSRM consists of six consecutive activities, which are implemented in this thesis. Figure 2 shows how the model by Peffers et al. (2007) is applied to this thesis. The Design & Development and the Demonstration stages require additional explanation which can be found in the next two paragraphs.

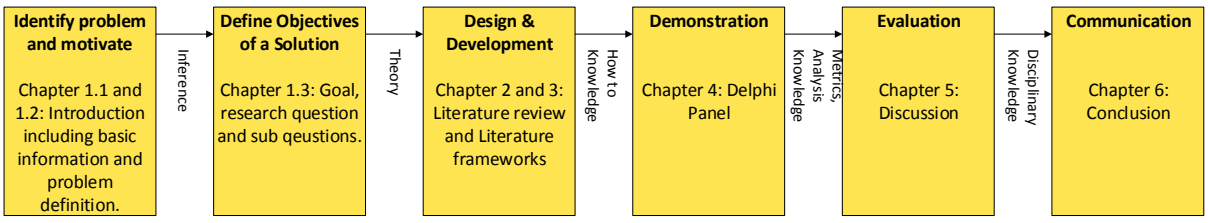

Figure 1: Application of DSRM by Peffers et al. (2007) to thesis.

1.4.2. Literature review method

As stated in figure 2, a literature review and literature frameworks on the topics of business rules, workflow commissioning and semantic checking are presented in chapter 2 and 3. For the selection and processing of the relevant literature, the Grounded Theory Literature Review Method (GTLRM) is

used, as illustrated by Wolfswinkel, Furtmueller, and Wilderom (2013). Due to the double research agenda of the literature review, this thesis is in need of structure and an efficient method for analyzing sources. GTLRM meets this need for structure and efficiency by introducing five phases for a literature review: *define, search, select, analyze* and *present*. Paragraph 2.1.1 elaborates further on GTLRM and its associated methods.

1.4.3. Empirical method

For the empirical part of this thesis, the Delphi panel is selected as the method to use. A Delphi panel's goal is to measure and possibly improve the consensus of participants on a certain subject by providing room for discussion and interpretation of each other's choices and opinions. These properties match the requirements of the empirical research for this thesis: input needs to be gathered on the usability of the theoretical concepts in the context of Flowmanager. The group of experts exists of persons who use Flowmanager daily for workflow commissioning.

Other properties of a Delphi Panel match the requirements for this thesis as well. A Delphi panel requires a relatively small number of participants, something which is the case for this thesis (11 participants). A Delphi panel is also flexible in its setup (time and location), which is useful taking into account the spread of the participants over the Netherlands and their busy schedules. Chapter three elaborates further on the motivation and set up of the Delphi panel in this thesis.

1.5. Relevance of Thesis

1.5.1. Academic

The academic relevance of this thesis can be found in the empirical aspect. As stated in the research question and sub questions, theories are compared with a system which operates on a daily basis. This gives us an indication which theories can actively contribute to the day to day works of businesses. This comparison also shows if certain theories are outdated, should receive some additions or should even be completely reviewed in order to be in line with the current developments. It also tells what specific parts of said theories should undergo these potential changes. Overall, this thesis outlines which theories can be used in practice and which are less useful, it indicates the usability of the discussed theory in practice.

1.5.2. BuildIT

The relevance of this thesis for BuildIT is logical: it gives BuildIT the opportunity to cross the t's and dot the i's for workflow commissioning of the Flowmanager framework. This thesis also provides guidelines for the first implementations of semantic checking. The advice leads to a more complete, user friendly and attractive product for the client. In its turn, this could lead to a higher rate of client satisfaction with the current clients. Also, when the proposed changes are implemented based on the surveys amongst clients, the clients feel like being listened to. This improves the relationship between BuildIT and the clients.

Finally, from an academic point of view, BuildIT has the opportunity to compare their work on semantic checking and controls with relevant and recent theory. This gives them insight into where they stand compared to the standard and best practices and it provides them with specific guidelines on which improvements can be made.

1.6. Thesis Outline

This paragraph describes the structure of the remainder of this thesis, as displayed in figure 3. Chapter two describes the literature review and the used methodologies. In chapter three, two literature frameworks provide an in depth analysis on the subjects workflow commissioning and semantic checking. In chapter four the Delphi Panel is presented. This chapter provides a brief explanation on

Delphi Panels, describes the used methodology and set-up of the panel, followed by a presentation of the results. Chapter 5 presents the discussion of a comparison of the literature review with the results of the Delphi Panel. Finally, Chapter 6 provides a conclusion for this thesis, among which recommendations are made to BuildIT.

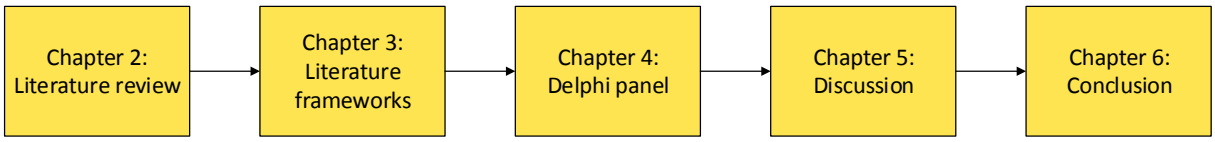

Figure 2: Thesis outline.

2. Literature review on business rules

This chapter aims at creating a scientific foundation which is required for the upcoming chapters of this thesis. As stated in the introduction, the following chapters focus on conducting a Delphi panel and the analysis of this panel. The theory presented and discussed in this chapter provides contents for the execution and analysis process of the Delphi Panel. In order to create a proper foundation, the literature review is created based on the 'Grounded Theory Literature Review Method' as described by Wolfswinkel et al. (2013).

The remainder of the chapter is structured as follows: Firstly, an elaboration of the Grounded Theory Literature Review Method is provided. Secondly, the methodology for the literature review of this thesis based on the Grounded Theory Literature Review is described. Finally, the field of business rules and its connection to workflow commissioning and semantic checking is presented by comparing several articles on the topic of business rules.

2.1. Grounded Theory Literature Review Method

2.1.1. Explaining Grounded Theory Literature Review Method

This thesis requires a broad literature review since advice on improvements is given in two different fields. Such an extensive review needs a lot of structure in order to create well-argued and concrete guidelines and best practices for each of the discussed topics. The Grounded Theory Literature Review Method (GTLRM) can provide such a structure: "*The aim of using a Grounded Theory approach to literature reviewing is to reach a thorough and theoretically relevant analysis of a topic.*" (Wolfswinkel et al., 2013). GTLRM does this by introducing five stages which guide the composition of the literature review, as can be seen in figure 4:

Five-stage grounded theory method for reviewing the literature in an area.

Number	*Task*
1. DEFINE	
1.1	Define the criteria for inclusion/exclusion
1.2	Identify the fields of research
1.3	Determine the appropriate sources
1.4	Decide on the specific search terms
2. SEARCH	
2.1	Search
3. SELECT	
3.1	Refine the sample
4. ANALYZE	
4.1	Open coding
4.2	Axial coding
4.3	Selective coding
5. PRESENT	
5.1	Represent and structure the content
5.2	Structure the article

Figure 3: GTLRM stages (Wolfswinkel et al., 2013).

Define

The first of the five steps has its emphasis on increasing the efficiency by forming a clear scope for the consecutive search phase. In the Define stage, criteria are defined for the inclusion or exclusion of

certain articles (for example limitations in publication date and types of publications). The next task is to guide the search to the correct research field, for instance information systems or workflow management. The third task provides a scope in the selection of academic sources. Researchers should ask themselves questions such as: Which fields are interesting for my research? Which fields have overlap with my fields? The final task of the Define step is the selection of search terms, which should "*cover the entire scope of the research area."* (Wolfswinkel et al., 2013).

Search

Based on the Define stage, the search can be initiated. During this stage it can become apparent that certain terms were missing or obsolete. It is important to keep track of the changes made in this stage, as they might have consequences for other parts of the thesis.

Select

In the Select stage, choices are made about which articles to include or exclude. As shown in figure 5, every article has to go through several 'filters'. Articles are checked for doubles, after which they are filtered based on title and abstract: if criteria as formulated in the Define stage are not met, sources are excluded. The next step is to read the full text and repeat the inclusion / exclusion process. Also, citations need to be checked in order to find more relevant and closely linked articles. If new articles arise, the entire process has to be executed again.

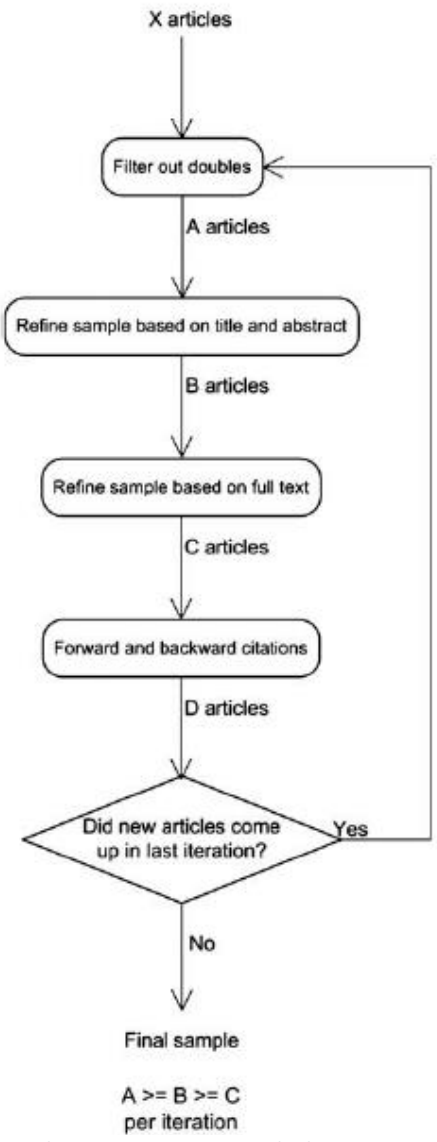

Figure 4: Select stage in reviewing the literature in an area.

Analyze

"*The procedures of grounded theory are designed to develop a well-integrated set of concepts that provide a thorough theoretical explanation of social phenomena under study."* (Corbin & Strauss, 1990). In their article, Corbin and Strauss (1990) state that grounded theory is not just descriptive, it also has an explanative aspect. In contrary to other methods, the grounded theory approach takes into account and explains specific conditions of single cases and uses these conditions to create a complete and detailed overview of all the relevant concepts.

Other methods often focus on a specific article and provide an overview of what each article entails. Grounded theory on the other hand, is a continuous review process which makes an analysis based on the concepts found in the various information sources. The concepts are grouped in categories, which can be compared with each other, eventually leading to a core category, which "*represents the central phenomenon of the study*." (Corbin & Strauss, 1990).

Grounded theory's end goal is to provide a complete overview of concepts in a specific theoretical field around a core category, whilst

keeping their respective context in mind. This end goals makes grounded theory an appropriate method to use for the literature review of this thesis, since the output should consist of best practices and guidelines in two specific fields.

In order to achieve this end goal, grounded theory makes use of a number of canons and procedures, basic laws about data collection and analysis which need to be followed in order to perform a successful grounded theory method. Among these canons several topics are discussed, such as the interrelation of data collection and analysis, development of concept categories and comparisons of cases and categories (Corbin & Strauss, 1990).

The successful arrival at the end goal of grounded theory is guided by three types of coding, being open coding, axial coding and selective coding. Babbie (2007) defines coding as "*the process whereby raw data are transformed into standardized form suitable machine processing and analysis."*

Open coding is the first step in the coding process. Its emphasis lies on a first analysis, classification and labelling of concepts which are derived from the data (Babbie, 2007). By open coding, researchers can break down data for a better understanding and the creation of new insights. The concepts which are found by this type of coding are given a preliminary label and are placed into categories (Corbin & Strauss, 1990).

The next step, axial coding, makes use of the categories formed in the open coding process. Its goal is to detect the concepts which are the most important for the research. The transition from open coding to axial coding is not very strict. Open coding can still occur next to the axial coding process, should new theory emerge. Thus, new (sub)categories can still be formed and data can still be relocated to other categories (Babbie, 2007). In addition, Corbin and Strauss (1990) state that "*categories are related to their sub categories and the relationships tested against data*" in the axial coding stage. After the process of labelling data and concepts in the open coding stage, the input in categories is compared and checked for similarities. These core concepts, which can be found in multiple sources, provide a first glance at the best practices which are required for the rest of this thesis (Corbin & Strauss, 1990).

During the selective coding process, one central core category is selected to which all the other categories, sub categories and their concepts are related. Selective coding usually occurs in the final phase of analyzing the data. It is this category which can answer questions such as: "*What is the main analytic idea presented in this research? What does all the action/interaction seem to be about?*" (Babbie, 2007).

Present

The final stage of the GTLRM is the presenting of the findings from the previous steps. The type of presentation depends on the findings that have been done. Emphasis can be on a diversity of subjects, such as the core category in case of clear relations among the various categories. On the other hand, if there are a lot of exceptions which differ from the category, emphasis could be on these exceptions. Presentations of findings often benefit from visual output, such as tables and diagrams (Wolfswinkel et al., 2013).

2.1.2. Applying GTLRM

This chapter describes the application of the previously mentioned methods in this chapter.

2.1.2.1. Inclusion / Exclusion criteria for literature

An important step in the creation of a solid literature review is the inclusion / exclusion of sources. The following criteria have been used in the development of the review as described in paragraph 2.2.2:

- The scientific fields surrounding the topics of information services are subject to a high degree of change due to innovations in the respective fields. In order to cope with these transformations, a guideline has been set for acquiring literature: sources dating from before 2000 are excluded, unless the amount of citations exceeds 250.
- In order to be included, the selected study has to contain best practices and/or guidelines for either semantic checking, business rules and/or workflow management. The following definitions apply for these search terms:
    - Semantic checking: verification of the semantic correctness of a process in a workflow, by monitoring the correct application of constraints in a workflow process (Ly et al., 2008).
    - Workflow management: "*An approach to the problem of controlling, monitoring, optimizing and supporting business processes.*" (Salimifard & Wright, 2001).
    - Business Rules*: "A statement that defines or constrains some aspect of the business."* (Charfi & Mezini, 2004).
- Schwarz and Russo (2004) have created a list of the top 50 journals in the field of Information Systems, presented in figure 6. Although publication in one of these journals is not essential, it provides a good indication of the quality of the document.

WORLD RANK	TITLE	WORLD RANK	TITLE	WORLD RANK	TITLE
1	*MIS Quarterly*	18	*Communications of the AIS*	35	*Journal of Information Systems*
2	*Communications of the ACM*	19	*IEEE Computer*	36	*The Information Society*
3	*IS Research*	20	*Journal of Strategic IS*	37	*Journal E-U Computing*
4	*Journal of MIS*	21	*Admin. Science Quarterly*	38	*Info Resources Mgmt Journal*
5	*Management Science*	22	*Academy of Mgmt Review*	39	*Interfaces*
6	*IEEE Transactions* (various)	23	*Int'l Journal of E-Commerce*	40	*EM - Electronic Markets*
7	*Harvard Business Review*	24	*ACM Computing Surveys*	41	*Journal of CIS*
8	*Decision Sciences*	25	*Accounting, Management & IT*	42	*European Journal of OR*
9	*Decision Support Systems*	26	ACM SIG Publications	43	*Operations Research*
10	*Information and Management*	27	*IT and People*	44	*Int'l Journal of H-C Studies*
11	*European Journal of IS*	28	*IBM Systems Journal*	45	*Journal of the ACM*
12	*Sloan Management Review*	29	*OMEGA*	46	*Australian Journal of IS*
13	*ACM Transactions* (various)	30	*Journal of the AIS*	47	*Org. Behavior and Human Dec.*
14	*Data Base*	31	*Journal of Org., Comp. and EC*	48	*Behavior and IT*
15	*Organization Science*	32	*Human-Computer Interaction*	49	*Scandinavian Journal of IS*
16	*Information Systems Journal*	33	*Information Systems Management*	50	*Computer Journal*
17	*Academy of Management Journal*	34	*Int'l Journal of Man-Machine Studies*		

Figure 5: Top 50 in IS Journals.

2.1.2.2. Search methodology

In order to retrieve the required articles, the renowned academic search engine Web of Science by Thomson & Reuters[1] has been used. This engine is known for delivering results from highly regarded sources and their filtering options. One issue with Web of Science is that it occasionally is unable to deliver the required articles. In these cases, Google Scholar has assisted by delivering the articles which were originally found with Web of Science.

Another important part of the methodology are the search terms which were used for finding results. The search terms are based on the previously stated disciplines and on the research questions, as suggested by Schwarz and Russo (2004).

[1] http://www.isiknowledge.com

The following search terms have been used:

Workflow	Semantic checking
IT Process management	Continuous auditing
Workflow management	Continuous monitoring
Model checking	Semantic checking
Business rules	Automatic compliance
Automatic compliance	Semantic controls
	Conformance analysis

Table 1: Search terms per literature review topic.

2.1.4.5. Presentation of relevant literature

The coding procedure as illustrated in the previous paragraph, requires a constant revision of the concepts and categories in order to provide a well-founded and complete overview of best practices and guidelines for each of the two topics. This thesis however only entails the end results of the literature review and coding procedure: only the articles which make a contribution to this thesis are discussed. Presenting the separate stages of the literature review results in clutter, confusion and irrelevant information.

2.2. Literature review

This paragraph centers on a literature review of the research fields relevant to this thesis. First, a number of definitions is provided which aids to a full understanding of the literature review and literature frameworks. Second, the comprehensive field of business rules is introduced, which provides valuable information on both the topics of workflow commissioning and semantic checking. Finally, a conclusion is provided, connecting business rules to workflow commissioning to semantic checking.

2.2.1. Definitions for literature review

- Business process: A process consisting of a "*sequence of activities. It has distinct inputs and outputs and serves a meaningful purpose within an organization or between organizations."* (van der Aalst et al., 2003).
- Business process management: The collection of tools and methods used by organizations in understanding, managing and improving their process portfolio, and in identifying and quantifying processes with outsourcing potential (zur Muehlen & Indulska, 2010).
- Business rules management: The identification, definition and management of business rules using technology such as Business Rules Management Systems (zur Muehlen & Indulska, 2010).
- Activity: "*A discrete process step performed either by a machine or a human agent. An activity may consist of one or more tasks."* (van der Aalst et al., 2003).
- Workflow: A workflow "*is a process in which documents, information or tasks are passed from one participant to another. It is a set of activities involving the coordinated execution of multiple tasks performed by different processing entities and covers the flow of information and control within and between organizations.*" (Salimifard & Wright, 2001).
- Workflow Management System (WfMS): An information system which controls transactions between activities within a workflow (Stohr & Zhao, 2001).
- Workflow instance: A single case in a workflow process (Verbeek, Basten, & van der Aalst, 2001).
- Semantic correctness: The analysis of the logical relations between two or more activities in a workflow process (Ly et al., 2008).

- Semantic conflict: A situation where two or more activities in a workflow process collide with each other, based on conditions defined in the constraints (Ly et al., 2008).
- Semantic checking: Verification of the semantic correctness of a process in a workflow, by monitoring the correct application of constraints in a workflow process (Ly et al., 2008).

2.2.2. Literature review on business rules

A key subject for this thesis is business rules, a field which encompasses both workflow commissioning and semantic checking. In this section, an overview is provided of several properties of business rules. In the next chapter, two articles which provide an in-depth view on workflow commissioning and semantic checking are explained.

In their article, Charfi and Mezini (2004) describe a business rule as "*a statement that defines or constrains some aspect of the business."* Having business rules provides a company structure and the opportunity to control its business processes. Also, by applying business rules, parts of a process can be updated or adapted without influencing the entire process (Charfi & Mezini, 2004).

Many different papers have categorized business rules in their own way. These various methods show similarities in the way business rules are classified. Four different papers are briefly discussed after which one method is selected which is most appropriate for this thesis.

First of all, The Business Rules Group (2000) describes four categories of business rules. The first category is *definition of business terms* and entails business rules that put emphasis on describing the various aspects of a business rule so that it can be categorized and interpreted properly. The second category is *facts relating terms to each other* and consist of business rules focusing on describing the structure of a process. The third category is known as *constraints*. Business rules in this category provide limitations for the commissioning of a workflow. The fourth and final category is *derivations*. These business rules focus on the derivation of business rules from other business rules (The Business Rules Group, 2000).

Both Charfi and Mezini (2004) and Iacob and Jonkers (2009) make a distinction between two types of business rules. Charfi and Mezini (2004) describe these types as (1) *constraints* and (2) *if conditions then action*. Iacob and Jonkers (2009) refer to them as (1) *constraints* and (2) *rules that influence the operational process*. The first type of business rules provides limitations for the business process, the latter type focuses on structuring the business process. Since these two articles provide nearly similar definitions, this thesis will make use of the categorization by Iacob and Jonkers (2009). This categorization fits well with the two remaining research topics of workflow commissioning and semantic checking.

In their article, zur Muehlen and Indulska (2010) also make a distinction between different types of business rules, but provide more specific categories to which a business rule can be allocated: *Integrity*, *transformation*, *derivation*, *reaction* and *production* rules. These categories show similarities with the categories as described by The Business Rules Group (2000) and Iacob and Jonkers (2009). A comparison of the categories is displayed in table 2.

	Categories by Iacob and Jonkers (2009)	
	Constraints	**Operational process rules**
Categories by The Business Rules Group (2000)	Constraints	Definition of business terms
	Derivations	Facts relating terms to each other
Categories by zur Muehlen and Indulska (2010)	Integrity rules	Derivation rules
	Transformation rules	Reaction rules
		Production rules

Table 2: Comparing categories of zur Muehlen and Indulska (2010) and The Business Rules Group (2000) with Iacob and Jonkers (2009).

zur Muehlen and Indulska (2010) acknowledge two modeling types of Workflow Management Systems (WfMS's). These are characterized by the way they represent a workflow and the business rules. Both types have the same goal: presenting a business process as realistic and complete as possible. Although they have the same goal, their properties are very different. These two types show resemblances with the two categories of business rules identified by Iacob and Jonkers (2009).

The first model type of WfMS's is the process modeling language and shows resemblances to the rules that influence the operational process as described by Iacob and Jonkers (2009). This type of WfMS is known for its focus on creating the sequence in which activities take place and the different paths which can be taken. The process modeling language has difficulties dealing with constraints and conditions. This is where the second model type comes in: the rule modeling language can fill this gap. The rule modeling language puts emphasis on defining constraints and conditions, similar to the constraints category as described by Iacob and Jonkers (2009).

In their article, zur Muehlen and Indulska (2010) conclude that there is no simple way of implementing properties of the rule modeling language into process modeling language. Further research needs to be conducted in this field. This conclusion suggests that it is more feasible to conduct two separate literature reviews for workflow commissioning and semantic checking.

Conclusively, business rules can fall in two different categories. The first category entails business rules that focus on the creation of structure and sequences in a workflow process, so called operational process rules (Iacob & Jonkers, 2009). These can be classified as an aspect of the process modelling language (zur Muehlen & Indulska, 2010). The second category consists of business rules focusing on relations between different aspects of a workflow process, so called constraints (Iacob & Jonkers, 2009), which fit in the rule modeling language (zur Muehlen & Indulska, 2010).

Workflow commissioning and semantic checking are two practical applications of the two modeling types as previously described. Workflow commissioning is a typical example of an activity which is associated with process modeling language: the focus lies on creating sequences and structure. Paragraph 3.1 elaborates further on a large variety of ways of commissioning workflows by discussing a review article by van der Aalst et al. (2003). Although Flowmanager appears to be a system which makes uses of the process modeling language, there is a need for influences from the rule modeling language in the form of semantic checks as described in the introduction. Therefore, the application of semantic checks in a variety of situations as described by Ly et al. (2008) is discussed in paragraph 3.2.

3. Literature frameworks on workflow commissioning and semantic checking

This chapter elaborates on the usage of the two practical applications belonging to two different types of business rules, as described in the previous chapter. First, emphasis is put on the subject of workflow commissioning by discussing the most important findings of van der Aalst et al. (2003). Second, the paper by Ly et al. (2008) is discussed by presenting their findings on the application of semantic checks in six different situations.

3.1. Literature framework on workflow commissioning

The search for best practices in the field of workflow commissioning by applying the Grounded Theory Literature Review Method, delivered two articles which include a review of several WfMS´s and their properties.

Workflow Management Systems analysis

First of all, Georgakopoulos, Hornick, and Sheth (1995) made a comparison of five WfMS's which were considered best practices at that time. However, this method focuses on describing the properties concerning the end results which can be achieved with the method, instead of describing the actual commissioning of the workflow. van der Aalst et al. (2003) provide us with a more recent comparison of WfMS's. In their article, they use 20 workflow commissioning patterns to distinguish between 15 WfMS´s. van der Aalst et al. (2003): "*The challenge, which we undertake in this paper, is to systematically address workflow requirements, from basic to complex, in order to (1) identify useful routing constructs and (2) to establish to what extent these requirements are addressed in the current state of the art."* The following two tables represent the results accomplished by van der Aalst et al. (2003):

	Product							
Pattern	Staffware	COSA	InConcert	Eastman	FLOWer	Domino	Meteor	Mobile*
1 (seq)	+	+	+	+	+	+	+	+
2 (par-spl)	+	+	+	+	+	+	+	+
3 (synch)	+	+	+	+	+	+	+	+
4 (ex-ch)	+	+	+/−	+	+	+	+	+
5 (simple-m)	+	+	+/−	+	+	+	+	+
6 (m-choice)	−	+	+/−	+/−	−	+	+	+
7 (sync-m)	−	+/−	+	+	−	+	−	−
8 (multi-m)	−	−	−	+	+/−	+/−	+	−
9 (disc)	−	−	−	+	+/−	−	+/−	+
10 (arb-c)	+	+	−	+	−	+	+	−
11 (impl-t)	+	−	+	+	−	+	−	−
12 (mi-no-s)	−	+/−	−	+	+	+/−	+	−
13 (mi-dt)	+	+	+	+	+	+	+	+
14 (mi-rt)	−	−	−	−	+	−	−	−
15 (mi-no)	−	−	−	−	+	−	−	−
16 (def-c)	−	+	−	−	+/−	−	−	−
17 (int-par)	−	+	−	−	+/−	−	−	+
18 (milest)	−	+	−	−	+/−	−	−	−
19 (can-a)	+	+	−	−	+/−	−	−	−
20 (can-c)	−	−	−	−	+/−	+	−	−

*Note that the modeling language of Mobile is extensible. The results only indicate the standard functionality. Many of the design patterns described in this paper can be added to Mobile.

Table 3: Main results of van der Aalst et al. (2003), part 1.

	Product						
Pattern	MQSeries	Forté	Verve	Vis. WF	Changeng.	I-Flow	SAP/R3
1 (seq)	+	+	+	+	+	+	+
2 (par-spl)	+	+	+	+	+	+	+
3 (synch)	+	+	+	+	+	+	+
4 (ex-ch)	+	+	+	+	+	+	+
5 (simple-m)	+	+	+	+	+	+	+
6 (m-choice)	+	+	+	+	+	+	+
7 (sync-m)	+	−	−	−	−	−	−
8 (multi-m)	−	+	+	−	−	−	−
9 (disc)	−	+	+	−	+	−	+
10 (arb-c)	−	+	+	+/−	+	+	−
11 (impl-t)	+	−	−	−	−	−	−
12 (mi-no-s)	−	+	+	+	−	+	−
13 (mi-dt)	+	+	+	+	+	+	+
14 (mi-rt)	−	−	−	−	−	−	+/−
15 (mi-no)	−	−	−	−	−	−	−
16 (def-c)	−	−	−	−	−	−	−
17 (int-par)	−	−	−	−	−	−	−
18 (milest)	−	−	−	−	−	−	−
19 (can-a)	−	−	−	−	−	−	+
20 (can-c)	−	+	+	−	+	−	+

Table 4: Main results of van der Aalst et al. (2003), part 2.

On the top of both tables, respectively eight and seven WfMS can be found. On the left, 20 workflow commissioning patterns are displayed. The explanation of all these patterns is provided in this paragraph and examples are given as well. The + symbol indicates that a certain WfMS has the possibility to commission the associated pattern. The – symbol indicates that a WfMS lacks the possibility to commission the associated pattern. A combination of both symbols (+/-) shows that a pattern is not directly available in the WfMS, but by combining other patterns such a pattern can be simulated.

Each of these WfMS's has their own interpretation on how a workflow is commissioned. Every WfMS has their own specific language which in most cases is developed specifically for the WfMS. This makes it difficult to compare WfMS languages and to learn from each other. If a certain pattern is missing, it does not mean that it can easily be copied from another WfMS. What is possible on the other hand is to analyze the differences in performance of the different WfMS's by using the tool provided by van der Aalst et al. (2003). Flowmanager can also be analyzed with this performance measurement tool, which indicates what patterns currently are not implemented yet.

At first sight, the article of van der Aalst et al. (2003) may seem outdated, due to the large span in time between the publishing of their article and the writing of this thesis. However, there are several arguments why this article is still relevant and up to date for this thesis. First of all, many of the workflow management systems mentioned are still used today and still contain the same possibilities in workflow commissioning. van der Aalst et al. (2003): "[…], *we have found that most new versions of workflow products bring new features in areas of performance, integration approaches, new platform support, etc. and feature minimal changes to the workflow modeling language which forms the core of the product. Therefore, many of the results presented in this paper will also hold for future versions of this product."* In short, the various uses of a WfMS may alter or improve over time, the basic foundation of workflow commissioning has reached a state of maturity and will most likely remain the same.

Secondly, the article by van der Aalst et al. (2003) is still cited frequently and is the most recent available analysis of workflow commissioning to date. According to Web of Science, the article is cited 690 times of which 72 citations date from the years 2013 and 2014, all in relevant journals in the field of Information Systems. Web of Science provides us with the following graph:

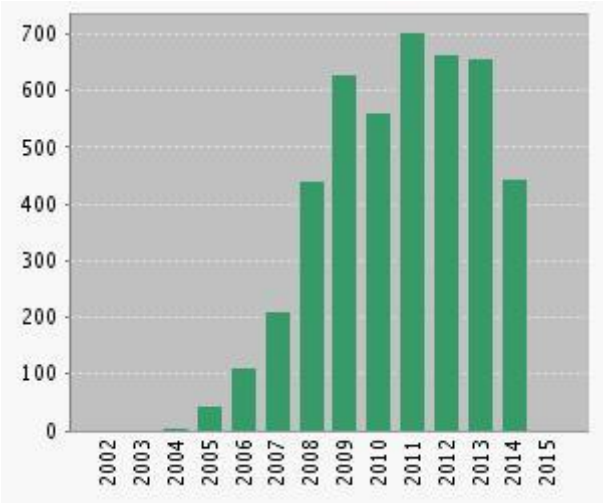

Figure 6: Citations of van der Aalst et al. (2003) (Thomson & Reuters, 2014).

Authors of articles often use the findings of van der Aalst et al. (2003) as part of the foundation of their research. Examples are often cited articles such as Ludascher et al. (2006), Russell, van der Aalst, ter Hofstede, and Edmond (2005) and recent publications such as Viriyasitavat, Xu, and Martin (2012) and Xu, Viriyasitavat, Ruchikachorn, and Martin (2012). Also, in recent articles Haddar, Makni, and Ben Abdallah (2014), Lanz, Weber, and Reichert (2014) and many others have acknowledged that the research done by van der Aalst et al. (2003) still provides a solid foundation for researching and analyzing WfMS's. Therefore, the article provides a useful tool for the analysis of Flowmanager.

The final remark concerns the usage of the results as provided by van der Aalst et al. (2003): Some of the WfMS's which were involved in the analysis have seized to exist or have become outdated. These WfMS's are excluded, a full explanation is located at the end of this paragraph, where the framework is adapted.

Workflow Patterns

In this section, the 20 patterns as used by van der Aalst et al. (2003) are briefly described and associated examples are provided. Each pattern is illustrated by a figure. The examples and figures have been devised specifically for this thesis. The examples and figures are based on the information provided by van der Aalst et al. (2003) and are applied to relevant situations from the financial sector. The patterns are classified into six groups: basic control flow patterns, advanced branching and synchronization patterns, structural patterns, patterns involving multiple instances, state based patterns and cancellation patterns.

Basic control flow patterns

This section describes the fundamentals, the basic straightforward patterns which can be found in WfMS's.

Pattern 1 – Sequence (seq)

After the completion of an activity in a process, the subsequent activity is initiated.

Example: After the a mortgage request is completed by an applicant (complete_mortgage_request), the bank checks the applicants details, such as social security number, name, birth date, etc. (verify_applicant).

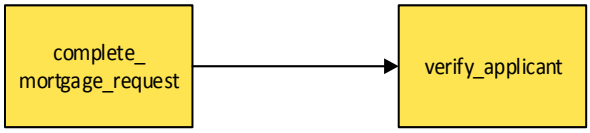

Figure 7: Example of a sequence pattern.

Pattern 2 – Parallel split (par-spl)

After the completion of an activity in a process, two or more subsequent independent activities are initiated.

Example: After the applicant is verified, a confirmation is sent to the applicant that his request is processed (inform_applicant) and the process of determining the maximum of the mortgage (assess_maximum_mortgage) is initiated.

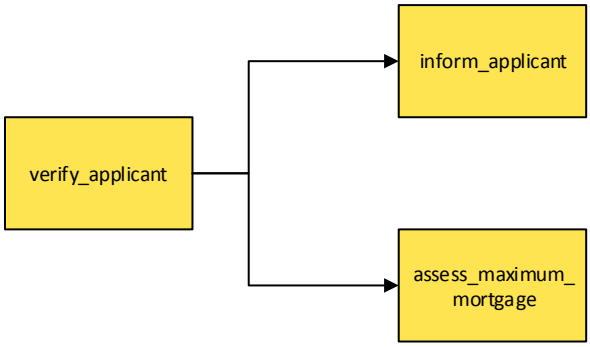

Figure 8: Example of a parallel split pattern with two subsequent independent activities.

Pattern 3 – Synchronization (synch)

An activity in a workflow process where multiple prior activities converge to, the activity synchronizes two or more preceding activities. Assumed is that the preceding activities only have one occurrence.

Example: After an applicant for a loan has been assessed as a suitable candidate (applicant_approved) and the applicant has agreed to the proposal (proposal_agreement), a contract needs to be signed (sign_contract).

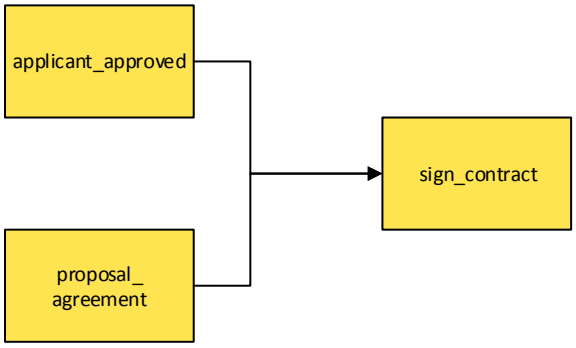

Figure 9: Example of a synchronization pattern with two preceding activities.

Pattern 4 – Exclusive choice (ex-ch)

A point in a workflow where, based on preceding activities, a decision is made between two or multiple subsequent activities. The decision can either be based on the judgment of an employee or on data from the preceding activities of the workflow process. In this pattern, only one of the alternatives can be selected. This is also known as a XOR-split. A XOR-split can be defined as an activity of a workflow process, where the process is forced to proceed in only one particular direction. XOR splits consist of multiple (at least two) potentially interesting paths (van der Aalst et al., 2003).

Example: When someone applies for a loan, the next activity depends on whether the applicant is known by the bank (customer_recognition). If the bank knows the applicant, only an internal check needs to be applied. If the applicant is new to the bank, an external check needs to be executed.

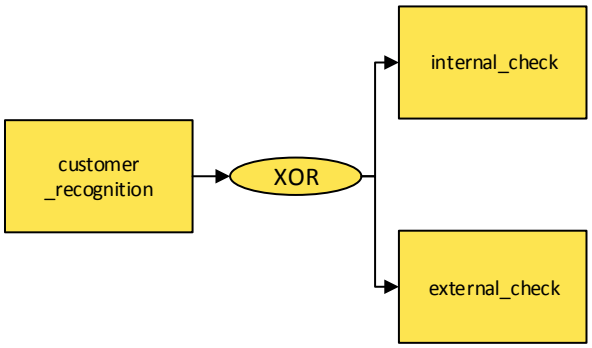

Figure 10: Example of an exclusive choice pattern.

Pattern 5 – Simple merge (simple-m)

Two or more activities come together in an activity without synchronization. van der Aalst et al. (2003): "*It is an assumption of this pattern that none of the alternative branches is ever executed in parallel.*" This pattern is also known as a XOR-join. Pattern 8 and pattern 9 provide alternatives in which branches are executed at the same time.

Example: An applicant has to provide a copy of either a passport (id_passport), ID card (id_card) or a driver's license (id_drive) for verification purposes.

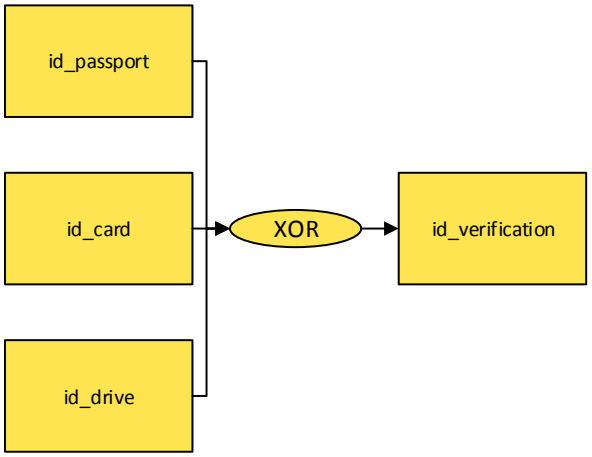

Figure 11: Example of a simple merge pattern.

Advanced branching and synchronization patterns

The patterns in this section are more advanced than the patterns in the previous section. The upcoming patterns provide specific options to the user in commissioning a workflow.

Pattern 6 – Multi-choice (m-choice)

Where Pattern 4 puts emphasis on making a decision between two or more activities (XOR-split), the Multi-choice pattern is less strict. It is possible to select multiple subsequent activities, which can operate independently (also called the OR-split). An OR-split is defined as an activity of a workflow process, where the process can proceed in multiple directions if required (van der Aalst et al., 2003).

Example: If the activity of verifying the identification has failed (id_failed) due to for instance a bad scan of the document or a scan of the wrong document, several subsequent activities could be executed. The client could be sent an e-mail (contact_mail) or given a phone call (contact_phone). If the employee of the bank has serious doubts about the reliability for the document, he could also involve the fraud department (involve_fraud_dept). Multiple activities could be selected, depending on the urgency of the case.

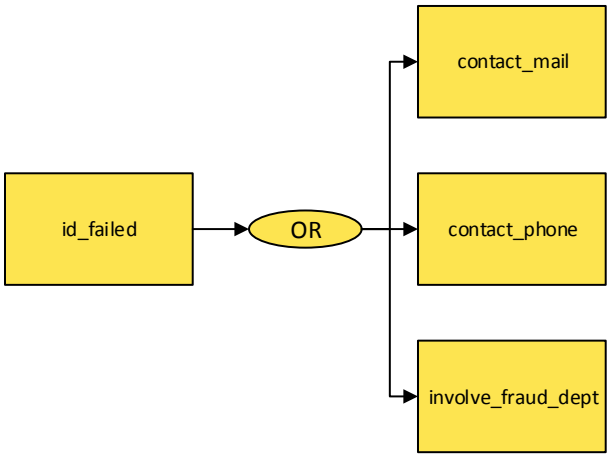

Figure 12: Example of a multi-choice pattern.

Pattern 7 – Synchronizing merge (sync-m)

Two or more activities come together towards one activity. It is possible that only one of the activities is completed, in that case the other simultaneous activities will converge. It is also possible that two or more activities are completed. Should this occur, then the activities which were executed need to be synchronized.

Example: After the applicant of pattern 6 has been contacted through phone and mail, he hands in a better copy of his ID by e-mail. This ID card is approved (id_approved) and the three activities are converged and no longer require actions to be undertaken.

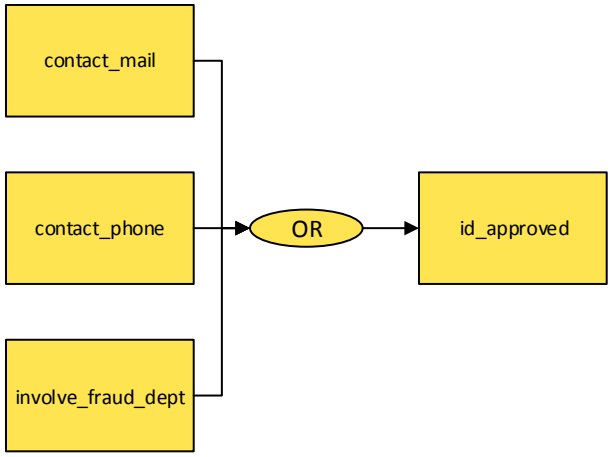

Figure 13: An example of a synchronizing merge pattern.

Pattern 8 – Multi-merge (multi-m)

A multi merge pattern occurs when two activities emerge from a single previous activity and converge towards a single activity, without being synchronized. This can happen when synchronization is irrelevant for the two activities, but they share the same subsequent activities.

Example: In order to make a final decision for an application of a loan (approve_application), various types of checks need to be executed. For instance: A creditworthiness check (credit_check) and a check for personal details (personal_check). Both these paths require an extra round of checking (final_check) in order to minimize the possibility of human errors.

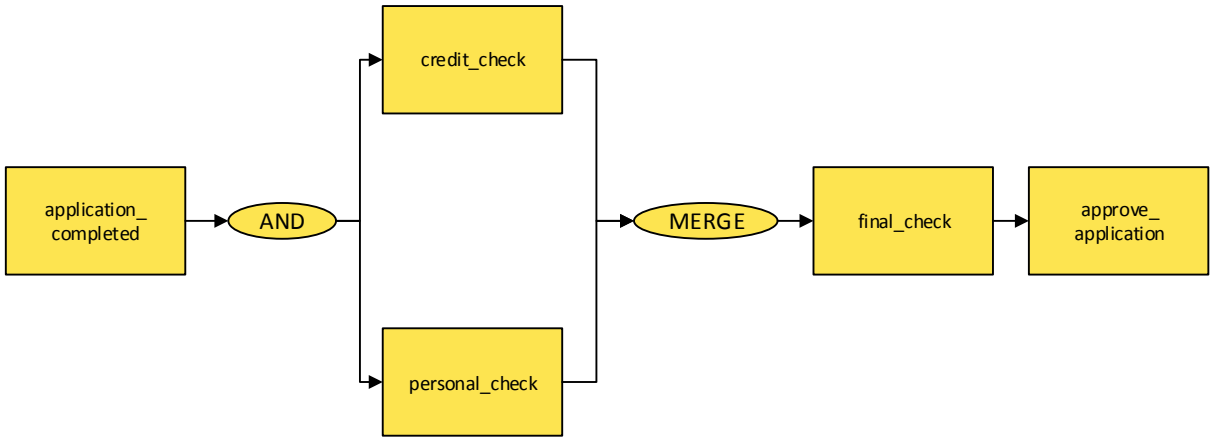

Figure 14: An example of a multi-merge pattern.

Pattern 9 – Discriminator (disc)

The discriminator pattern is a function in a workflow which waits for one of the preceding activities to complete itself, after which it triggers the next activity. The function does not wait for all the preceding activities to complete and provide data. It is also possible to gather data from multiple preceding activities. For instance, the next activity can be initiated if two out of three preceding activities are completed.

Example: For checking the credit worthiness of an applicant, three checks are conducted (credit_check_1, credit_check_2 and credit_check_3). If a client passes two out of three checks, he has passed the credit check.

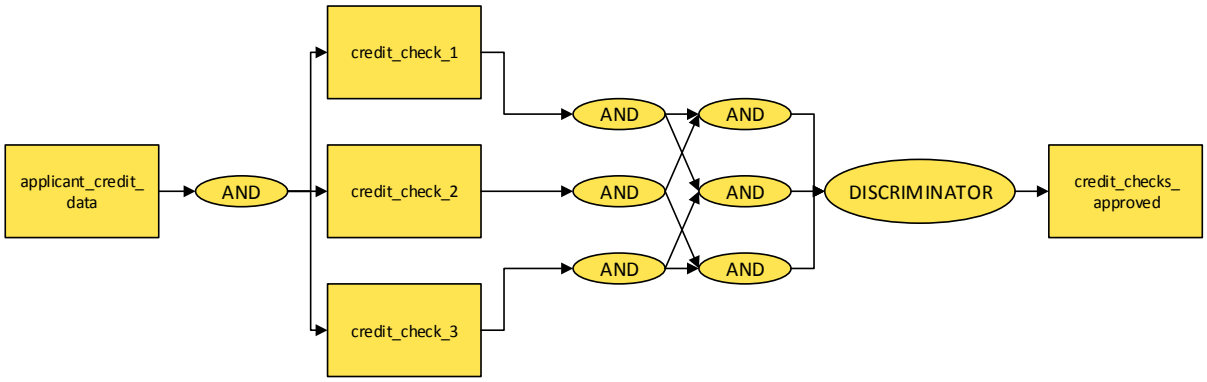

Figure 15: An example of a two out of three discriminator. Every combination of the three checks is possible.

Structural patterns

This section introduces two patterns that have a focus on implementing structure in an arbitrary environment. These two patterns show how difficult procedures can be structured, including the structuring of judgment issues and the exclusion of irrelevant processes.

Pattern 10 – Arbitrary cycles (arb-c)

The ability in a workflow to generate a loop, enabling patterns to repeat themselves. This can be done by making use of Pattern 4, Exclusive choice, one of the exclusive choices in this case will loop back.

Example: An employee of a bank notices during a check that some details of the applicant are incomplete. He sends feedback towards the client and the client has to alter the information. This loop can continue until the employee judges that the entered information is sufficient.

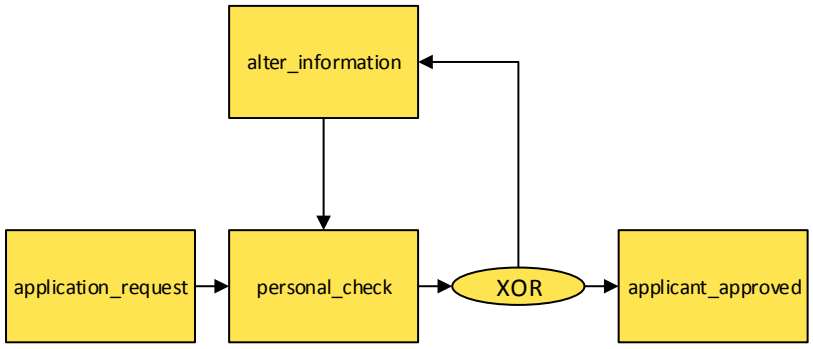

Figure 16: An example of an arbitrary cycle pattern.

Pattern 11 – Implicit termination (impl-t)

This pattern terminates a sub-process, when there are no running activities left in the workflow, nor are there activities available whose status can be changed to an operating one. The pattern is relevant for workflows which are not in 'deadlock'.

Example: Consider the example of the improved scan of the ID (pattern 6). The scan is approved and the next activity in the workflow can be initiated. After contacting the client through phone, he sends in an implicit termination, the activities contact_mail and involve_fraud_dept can be cancelled.

Patterns involving multiple instances

These patterns put emphasis on enabling the execution of multiple paths running simultaneously. The focus is on the converging of one activity to multiple concurrent activities with or without a priori knowledge and the diverging of multiple concurrent activities towards one activity.

Pattern 12 – Multiple instances without synchronization (mi-no-s)

Within a single path of a workflow, multiple instances of an activity can be generated which follow the same procedures. These activities operate independently and do not require synchronization.

Example: A customer of a bank applies for a mortgage and a loan at the same time (double_application). In the workflow, two threads are created (thread_controller) for each of the applications (mortgage_application and loan_application). Both of the paths require a creditworthiness check (credit_check) before proceeding to creating an offer for the customer (create_offer).

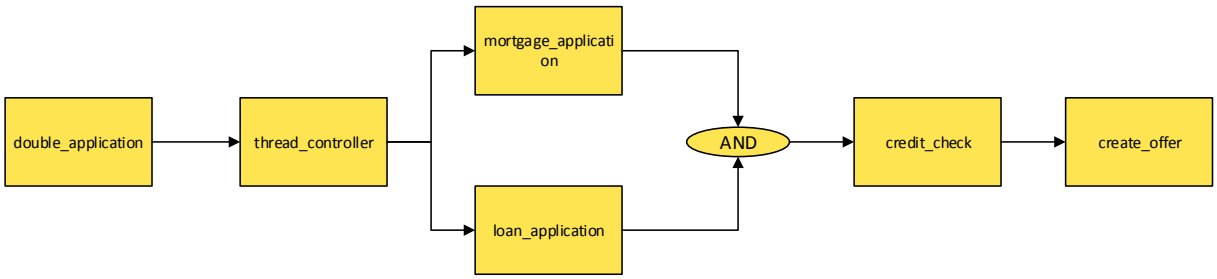

Figure 17: An example of a multiple instances without synchronization pattern with two instances.

Pattern 13 – Multiple instances with a priori design time knowledge (mi-dt)

For a single process, an activity has to be executed multiple times. The number of recurrences is known beforehand.

Example: Before making an offer (create_offer) to a loan applicant, the data provided by the applicant needs to be checked (check_1 and check_2) by two different employees in order to rule out human errors (four-eye principle).

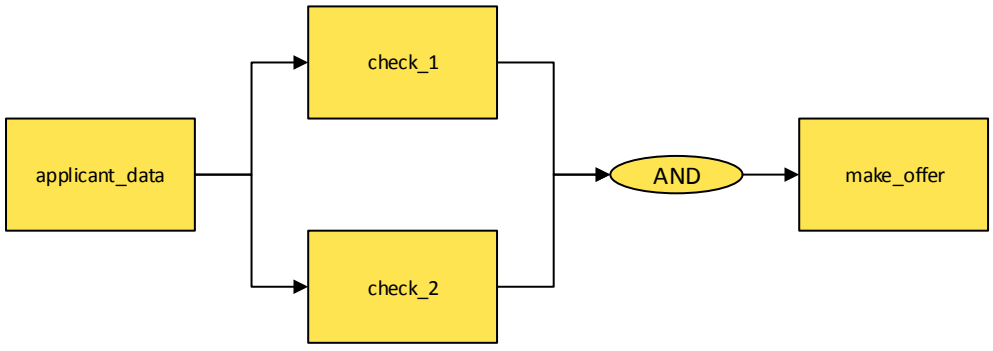

Figure 18: An example of a multiple instances with a priori design time knowledge pattern.

Pattern 14 – Multiple instances with a priori runtime knowledge (mi-rt)

For a single process, an activity has to be executed multiple times. The number of recurrences is not known before the process is initiated, but is known before the instances have to be created.

Example: If company A decides to apply for a loan, its financial status is checked. Depending on the provided information and status of the company, several checks can be executed, like liquidity (check_liquidity), solvability (check_solvability), return on investment (check_roi), etc. The number of activities is determined before they are generated and executed.

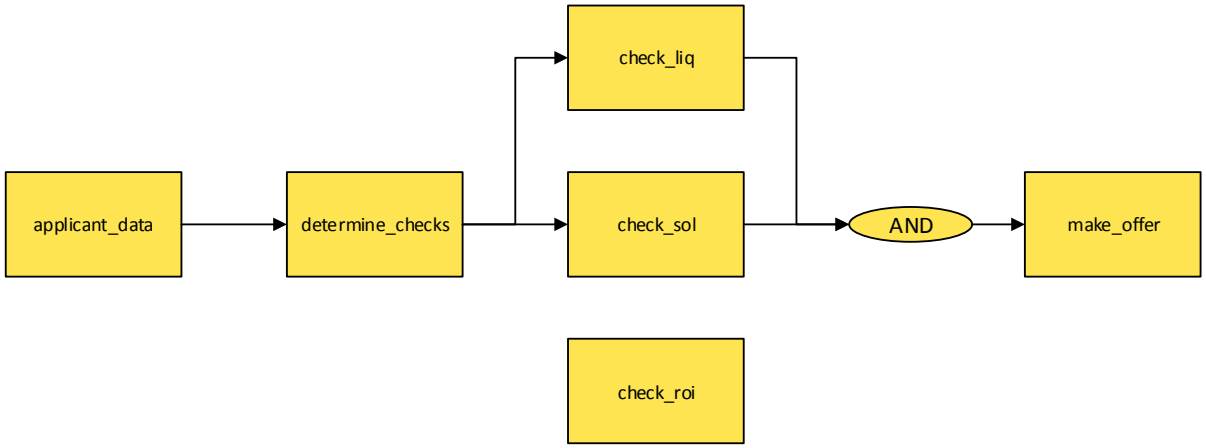

Figure 19: An example of multiple instances with a priori runtime knowledge, with two performed checks.

Pattern 15 – Multiple instances without a priori runtime knowledge (mi-no)

This pattern is similar to pattern 13 and 14, with the difference that no prediction can be made if and when new activities arise.

Example: van der Aalst et al. (2003) provide a good example: If an insurance claim is issued, it is possible to include reports of eye witnesses. Beforehand and during the processing of these reports it is not possible to predict the amount of witnesses.

A visual representation for this example is equal to figure 20, with the addition that the amount of activities cannot be predicted.

State based patterns

The patterns in this section put emphasis on making a choice between activities. Where the previous section had activities running simultaneously, the next three patterns are about selecting and activating the appropriate activity.

Pattern 16 – Deferred choice (def-c)

By using this pattern, a style of decision making is introduced in a workflow, which is not as explicit as the use of a XOR-split. Eventually, a choice is made between two or more alternative paths and the patterns which are not used are withdrawn. The choice between paths is postponed as long as possible, there is no strict deadline and the choice is made based on availability between the paths.

Example: A credit check (credit_check) for a mortgage application (mortgage_application) can be executed by two employees (employee_a and employee_b). The decision for either of the employees is based on availability, the path of the employee not executing the check is cancelled.

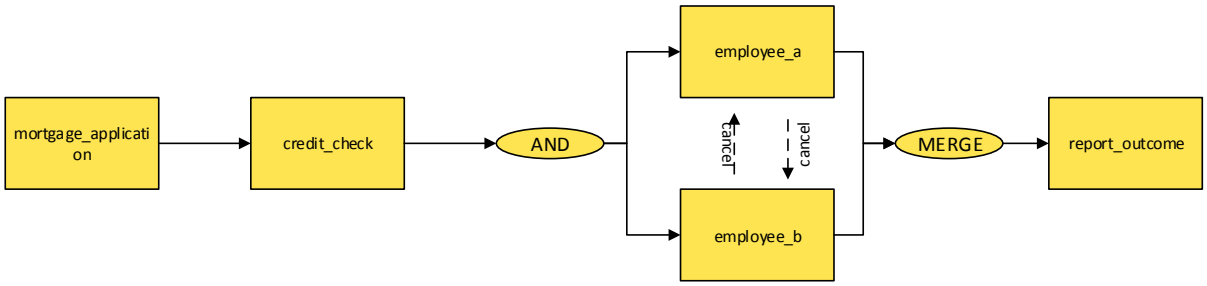

Figure 20: An example of a deferred choice pattern.

Pattern 17 – Interleaved parallel routing (int-par)

A pattern where two or more activities are executed in no particular order. The activities cannot be executed at the same time. In order to prevent simultaneous execution, tools managing interleaved sequences are used.

Example: After a first version of a mortgage proposal has been created (first_proposal) based on an application, its details have to be checked by two different employees (employee_a and employee_b). Both employees have the ability to edit the details of the mortgage, thus both employees need to execute their tasks non-simultaneously.

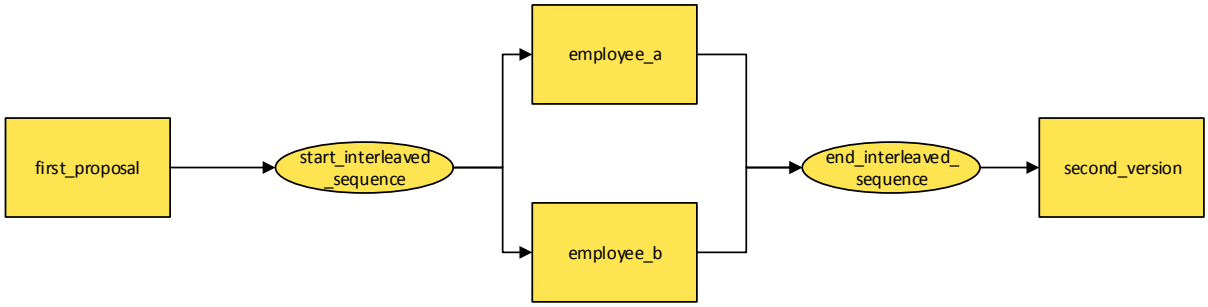

Figure 21: An example of an interleaved parallel routing pattern.

Pattern 18 – Milestone (milest)

The milestone pattern only initiates activities, if certain conditions have been reached or if certain preceding activities have been completed.

Example: A customer has applied for a loan (loan_application) at a bank. The bank has various actions to perform before it can issue an offer towards the customer. Among these actions are a liquidity check (check_liquidity) and a solvability check (check_solvability). If the application passes the liquidity check (M), it is a good indication that they will also pass the solvability check, thus another employee can start creating the offer (create_offer). The status M, indicates a milestone which has to be completed in order for create_offer to be initiated.

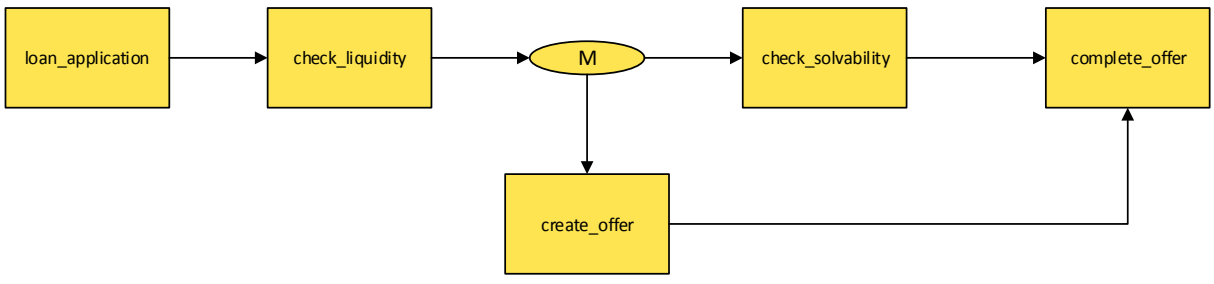

Figure 22: An example of a milestone pattern.

Cancellation patterns

The final category of patterns describes patterns that are created for canceling an activity or an entire workflow entry.

Pattern 19 – Cancel activity (can-a)

Activities that have been initiated, can be cancelled if their execution is no longer of use.

Example: A client has applied for a loan and a mortgage. However, the client has decided to withdraw their application for the loan. Activities concerning the loan have to be cancelled.

Pattern 20 – Cancel case (can-c)

The complete cancellation and removal of a case. Pattern 20 takes the cancellation a step further than pattern 19, since none of the details involved in the case are required in other parts of the workflow.

Example: After a client has put in a request for a loan and a credit check has been initiated, the client has changed their mind and has decided to cancel the application. Therefore, the entire case concerning the application has to be cancelled.

Conclusion

The framework provided by van der Aalst et al. (2003) provides an analysis tool for WfMS's which is essential for this thesis. Based on this tool, a new model is generated which is used to analyze Flowmanager. Some of the products mentioned by van der Aalst et al. (2003) are excluded in the new model. This dismissal of a workflow management system can have two reasons: either the system no longer exists or the system is no longer supported by its creator. For these reasons, the following six WfMS's are excluded: Inconcert, Eastman, Meteor, Mobile, Forté and Change. The rest of the WfMS´s remain in the table since they provide an indication to the current performance of Flowmanager.

The exclusion of these systems has led to the table as displayed at table 5. In this table, two changes have occurred. First, the six WfMS´s as described in the previous paragraph have been excluded. Second, a column has been included for Flowmanager for illustration purposes, since it is analyzed based on the twenty patterns as described by van der Aalst et al. (2003).

Patterns	Systems									
	Staffware	COSA	FLOWer	Domino	MQSeries	Verve	Visual WF	I-Flow	SAP/R3	Flowmanager
1 - Sequence	+	+	+	+	+	+	+	+	+	
2 - Parallel split	+	+	+	+	+	+	+	+	+	
3 - Synchronization	+	+	+	+	+	+	+	+	+	
4 - Exclusive choice	+	+	+	+	+	+	+	+	+	
5 - Simple merge	+	+	+	+	+	+	+	+	+	
6 - Multi choice	-	+	-	+	+	+	+	+	+	
7- Synchronizing merge	-	+/-	-	+	+	-	-	-	-	
8 - Multi merge	-	-	+/-	+/-	-	+	-	-	-	
9 - Discriminator	-	-	+/-	-	-	+	-	-	+	
10 - Arbitrary cycles	+	+	-	+	-	+	+/-	+	-	
11 - Implicit termination	+	-	-	+	+	-	-	-	-	
12 - M.i.* without synchronization	-	+/-	+	+/-	-	+	+	+	-	
13 - M.i. with a priori design time knowledge	+	+	+	+	+	+	+	+	+	
14 - M.i. with a priori runtime knowledge	-	-	+	-	-	-	-	-	+/-	
15 - M.i. without a priori runtime knowledge	-	-	+	-	-	-	-	-	-	
16 - Deferred choice	-	+	+/-	-	-	-	-	-	-	
17 - Interleaved parallel routing	-	+	+/-	-	-	-	-	-	-	
18 - Milestone	-	+	+/-	-	-	-	-	-	-	
19 - Cancel activity	+	+	+/-	-	-	-	-	-	+	
20 - Cancel case	-	-	+/-	+	-	+	-	-	+	

*Table 5: Analysis table for Flowmanager, partially derived from van der Aalst et al. (2003). *: M.i. = Multiple instances.*

3.2. Literature framework on semantic checking

The importance of flexibility and adaptivity in the usage of WfMS's is acknowledged by a number of articles. First of all, Ly et al. (2008) recognize that companies need to keep up with their business processes, due to frequently changing conditions in the global market: "*… adaptivity is the key factor for the successful application of process management technology in practice.*" (Ly et al., 2008). This point of view is shared by other authors, including Rinderle, Reichert, and Dadam (2004a) and Verbeek et al. (2001). These two articles focus on providing tools for the diagnosis of dynamic changes in WfMS's.

The main article used for this part of the literature review is provided by Ly et al. (2008). In their paper a framework is presented which aims at "*supporting semantic knowledge integration and semantic process verification in the context of changes at the process instance and at process template level".* Process template level is defined as an overarching layer which contains underlying instances. Changes at template level are dominant, underlying instances have to follow the changes (Ly et al., 2008).

For the answering of the research question, we need to look at the current state of development on the field of semantic checking in Flowmanager. As stated in the introduction, BuildIT is still trying to figure out how and when to use semantic checks. From a Business Administration perspective, emphasis in this stage of development lies on indicating what semantic checks are and when semantic

checks can contribute to the functioning of a workflow process. Even though the principles of a template level and associated instances by Ly et al. (2008) cannot be applied directly to Flowmanager (Flowmanager does not work with a template or instances), the article provides input for these two research topics. This article is relatively rare in its context since it provides information on semantic checking which can also be interpreted from a Business Administration point of view. The nature of the vast majority of articles covering semantic checks is far more technical, discussing topics such as the development and implementation of semantic check systems as described by for example Reichert and Dadam (1998) and Rozinat and van der Aalst (2008).

As stated, Ly et al. (2008) emphasize the importance of an adaptive WfMS and acknowledge the progress which has been made in the development in these systems thus far. On the other hand, Ly et al. (2008) recognize the difficult situation WfMS's are in: it is impossible to predict and model all exceptions for a WfMS. This leads to the modification of single instances in the workflow process. These exceptions could be semantically incorrect, regardless of their syntactical (in)correctness. Syntactical correctness is defined as the analysis of the syntax correctness of a workflow process (Ly et al., 2008).

This is one of the issues which Ly et al. (2008) address with their framework. An example is provided in figure 24. On the left, a number of constraints are provided. These requirements show certain demands an instance has to meet in order to be implemented successfully. When transforming instance I1 into instance I1', a new activity is introduced: Activity F is implemented between B and C. The constraints however, indicate that this is semantically incorrect, since Constraint 2 (C2) tells us that activity F cannot precede activity D.

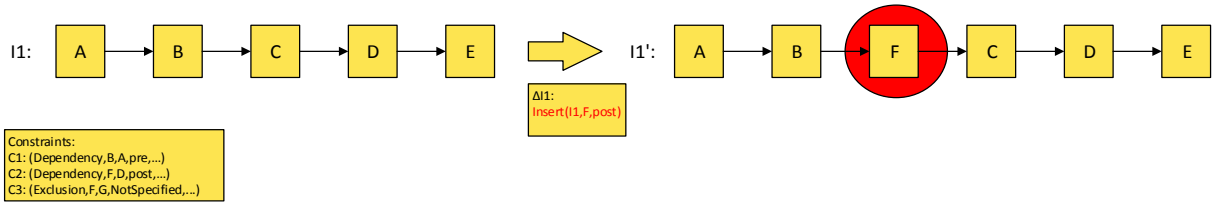

Figure 23: Example of a semantic issue at the instance level.

The second issue for which the framework is designed, is located at the template level. When a process template is updated (this could happen for several reasons, for instance new regulations or a change in strategy), this could have consequences for the existing instances, especially when these instances have been edited with an exception. An example is provided in figure 25.

When template S1 is updated to S2, this could have consequences for the (modified) instances which have been commissioned under the template and constraints of S1. Template S2 differs from template S1 in the addition of the activities F and G. This could lead to semantic issues, as can be deduced from the constraints: if activity F occurs, then activity B cannot be executed. Instance I1 does not encounter any semantic issues, since this instance takes the path of C instead of B. Instance I2 on the other hand experiences these issues, since the path of this instance in S1 included activity B and now activity F has been added.

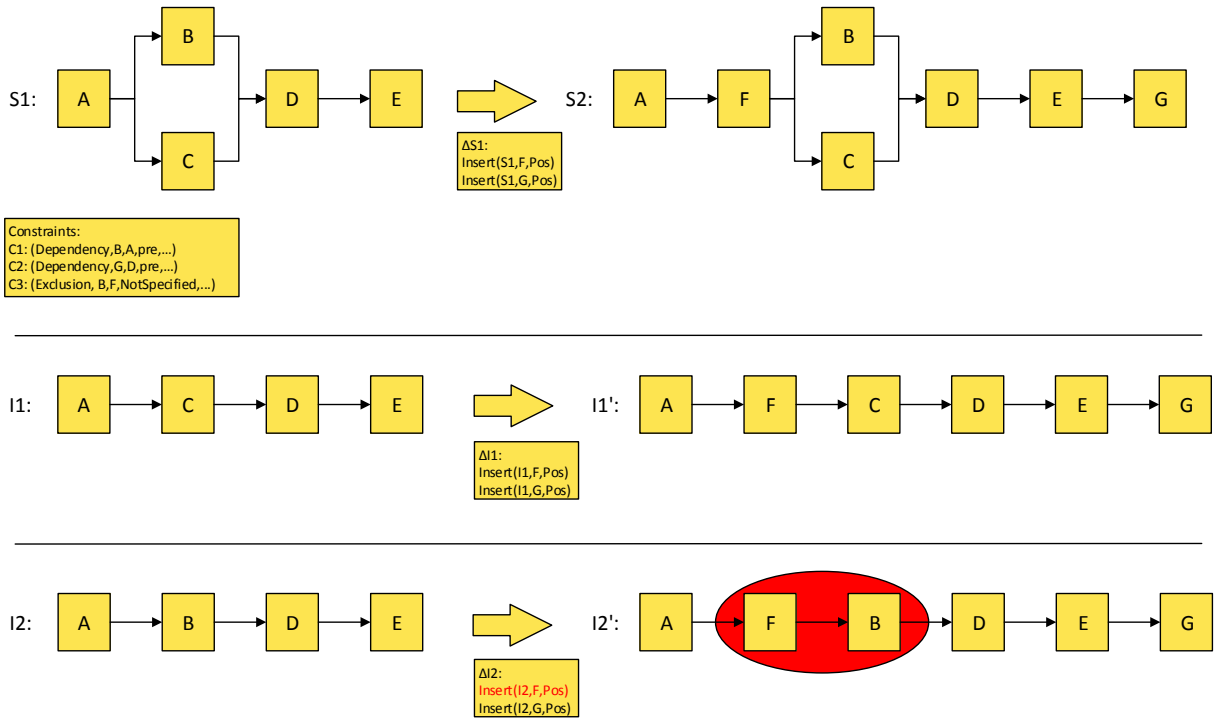

Figure 24: Example of a semantic issue at the template level.

The remainder of this paragraph presents the framework as created by Ly et al. (2008). Firstly, two important factors are discussed, dependence and mutual exclusion constraints. This elaboration is followed by a description of semantic correctness and how instances and templates can be checked for semantic correctness.

Dependence and mutual exclusion constraints

In their article, Ly et al. (2008) make a distinction between two types of constraints. The first of the two is the dependency constraint. Dependency constraints are defined as constraints in a workflow process which indicate set relations between two or more activities (Ly et al., 2008). This type of constraint states that the execution of an activity depends on the execution of another activity, in other words the two activities need to be carried out together. An example is the requirement of a credit check before making a mortgage offer to a client.

The second option is the mutual exclusion constraint. Mutual exclusion constraints are defined as constraints in a workflow process which prevent two or more activities from occurring together (Ly et al., 2008). This type of constraint indicates that two activities cannot be executed together. An example of this type of constraint is that a patient in a hospital waiting for brain surgery cannot be admitted aspirin, since this could interfere with the operation. These two types of constraints are used to shape the workflow process and to identify limitations in the commissioning.

A number of variations can be created of the two mentioned constraints. The syntax for the constraints is as follows: cX: (Type, Source, Target, Position, UserDefined), whereas:

- cX is constraint number X.
- Source is the original activity where the constraint refers to.
- Target is the activity to which the original activity is linked to by the constraint.

- Position is the determination of the order of the Source and Target in the process. Three alternatives can be issued, being Pre (Target precedes Source), Post (Source precedes Target) and NotSpecified (The positions are not relevant or unknown).
- UserDefined are comments from the creator of the constraint which can assist in the understanding and the correct implementation of the constraint.

Semantic correctness on the instance and template level

Ly et al. (2008) provide procedures for the semantic verification for two types of changes as stated previously in this paragraph: process instance changes and process template changes. An instance or process is semantically correct if all the demands defined by the dependency and mutual exclusion constraints are met.

On the level of process instance changes, three types of changes in instances can lead to semantic issues*: inserting*, *deleting* and *moving* of activities. The syntax of these changes is similar to the syntax of the constraints: ΔIX: Task(IX,a,Position), whereas:

- ΔIX is the change in Instance X,
- Task is one of the tasks insert, delete or move,
- IX stands for instance number X,
- a is the activity for which the change is relevant and
- Position is the new position of the new activity in the instance.

This notation of changes has great benefits for the detection of semantic issues. The examples in this thesis are simple and short workflow processes. In reality, workflow processes can have a number of activities much larger than the 5-10 which have been used in the examples, running up to over a 100 activities. Checking all these activities for semantic correctness would obviously take up a lot of time. The inspection of the entire process can be prevented by the notation of changes. Each of the three types of changes has its own need for checks which can be derived from the syntax (Ly et al., 2008).

Insert

When inserting an activity, the dependency constraints containing the *a variable* from the Task syntax as source need to be checked. Dependency constraints which indicate the *a variable* as the target also require a check for semantic issues, since the target has influence on the source. A distinction between the mutual exclusion constraints cannot be made, since both the source and target could influence the end result of a constraint. Both of the mutual exclusions types need to be included in the check.

Assuming that the rest of the process has been commissioned in a syntactic and semantic correct way, the remainder of the constraints (not using the *a variable*) can be *excluded* from the semantic check, since they experience no influence from the *a variable* (Ly et al., 2008).

From a practical point of view, the number of constraints which need to undergo the semantic check can be limited even further. If a constraint has included the *a variable* and the other involved task (either source or target) is a task that is not included in the running instances (for example instances taking only one path where multiple are available in a process), there is no direct need to mark this constraint as violated. This approach meets the demand of the present, but does not take into account future instances in this process which may alter from the current instances. If an instance is altered to follow a different path in the process, one which contains activities which are affected by a constraint previously ignored, a new check for semantic issues has to be executed.

Delete

When deleting *variable a* from an instance, the dependency constraints which have *variable a* as a source will have no issues with the alterations, since the constraints only have an effect on the deleted activity. These constraints can be excluded from the check. Ly et al. (2008) state that similar to inserting a new activity, the dependency constraints with *variable a* as a target need to be included. Mutual exclusion constraints cannot encounter issues by deleting an activity, since one of the two affected variables is deleted from the instance. Thus, mutual exclusion constraints can be excluded from the check. The same exclusion for constraints based on usage by current instances -as discussed with the inserting of a new activity- can be applied when looking for constraints to include in the semantic checking.

Moving

When a task is moved, it is actually being deleted and inserted in a new location. Therefore, all the constraints which need to be checked according to the Insert and Delete changes, need to be taken into account when moving a task to a new location (Ly et al., 2008).

Semantic correctness of instances in a changing template environment

As stated, one of the two types of changes which can be made in processes in a WfMS, are changes in templates. The previous paragraph presents how a change in process template can semantically be verified. This paragraph elaborates on a more complex question: When a process template is changed, how can running instances of a process be checked for semantic issues when they follow the same change paths?

In order to answer this question, a distinction is made between six different instance types, as described by Rinderle (2004) and Rinderle, Reichert, and Dadam (2004b). In their articles, these authors used the models as presented in figure 26 to detect issues in syntactical compliance between instances and templates. Since semantic checks investigate the same basic data as syntactical checks, the same model can be used. Ly et al. (2008) state that applying the model to all the instances involved in the template change can be very time consuming and in many cases irrelevant. The following paragraphs discuss the different types of instances. Afterwards a table is presented that determines which additional semantic checks should be executed, followed by a brief explanation.

The first distinction is made between biased and unbiased instances. Instances are unbiased if they have not received any individual alternations, for example I_1 in figure 26. Obviously, these instances do not require any additional semantic checks in addition to the ones executed in the template change.

The biased instances can be divided into the disjoint bias and the overlapping bias, based on the overlap between the template and the instance. If an instance is altered in such a way that these changes differ *completely* from the changes made on the template level (I_6), it is characterized as a disjoint bias.

Instances whose changes do not differ completely from the changes on template level are placed in the overlapping bias. In its turn, the overlapping bias can be divided into three biases: the equivalent bias, subsumption equivalent bias and the partially equivalent bias. The equivalent bias (I_2) has the exact same changes as the template. The subsumption equivalent bias is divided in two biases (I_3 and I_4): one where the changes in the instance (ΔI) subsume the changes in the template (ΔS) and vice versa. The final bias is the partially equivalent bias, which has one or more changes which are in line with the changes at the template level, but also has changes which differ from the changes on template level.

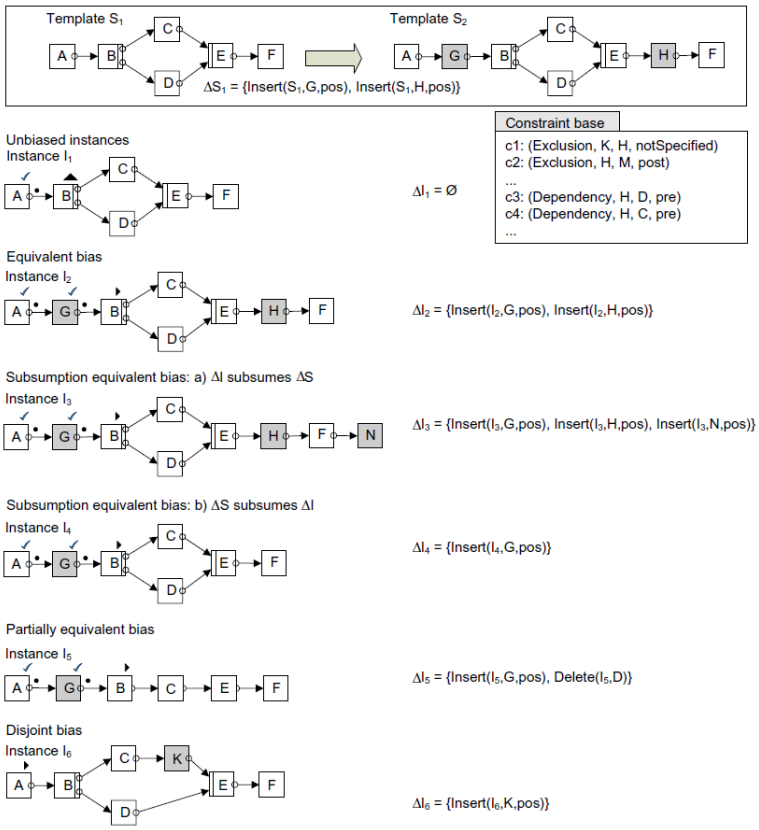

Figure 25: A template change for six different instance types (Ly et al., 2008).

The following table shows which biases require additional semantic checks compared to the checks already performed, followed by an explanation.

Instance type	Additional checks to be performed
Unbiased	None
Equivalent bias	None
ΔI subsumes ΔS	None
ΔS subsumes ΔI	None
Partially equivalent bias	ΔS
Disjoint bias	ΔS\ΔI

Table 6: Additional semantic checks for different biases, based on Ly et al. (2008).

As stated, the instances which can be characterized as unbiased, require no additional checks. Instances in the equivalent bias have the exact same changes as the template and do not require any extra checks. The instances which fall under the ΔI subsumes ΔS bias basically contain the equivalent bias and the addition of tasks which are not included in the changes of the template. The equivalent bias requires no additional checks and neither does the addition of a new task, since it is assumed that a modification of a single instance is semantically checked when it is originally implemented. As stated in the name of the ΔS subsumes ΔI bias, all the changes of I are included in ΔS and additional changes in I will be made based on ΔS. No additional checks are required.

The first type of instance changes which requires additional checks, is the partially equivalent bias. This type of instance has (a number of) changes in common with the template change but also has some unique alterations. If these instances are updated to the new template version, semantic conflicts could arise, caused by applying the changes made in S on the individual changes of the instance. Only

the constraints which contain changes in S which are new to the instance need to be semantically checked.

The second type of instance changes in need of supplementary checks is the disjoint bias. Given that the changes of an instance and the changes of a template differ completely, additional semantic checks need to be executed. All the constraints which contain either the instance modification or template modification need to be checked (Ly et al., 2008).

This part of the thesis has been marked as confidential.

Figure 26: Illustrating template changes in the context of Flowmanager.

As stated, the partially equivalent bias and the disjoint bias are the only two biases which need additional semantic checks. The following table by Ly et al. (2008) provides an overview of which constraints need to be checked when modifications are made to the two aforementioned biases. Alterations in the instances can be found on top, alterations in the template can be found at the left.

Table 3
Potentially violated constraint types depending on the applied change operations at template and instance level

	Insert(I, a, pos)	Delete(I, a)	Move(I, a, pos)
Insert(S, b, pos)	(Exclusion, $b, a, \ldots$) (Exclusion, $b, a, \ldots$)	(Dependency, $b, a, \ldots$)	(Exclusion, $b, a, \ldots$) (Exclusion, $a, b, \ldots$) (Dependency, $b, a, \ldots$)
Delete(S, b)	(Exclusion, $a, b, \ldots$)		(Dependency, $a, b, \ldots$)
Move(S, b, pos)	(Exclusion, $b, a, \ldots$) (Exclusion, $a, b, \ldots$) (Dependency, $a, b, \ldots$)	(Dependency, $b, a, \ldots$)	(Dependency, $b, a, \ldots$) (Dependency, $a, b, \ldots$) (Exclusion, $b, a, \ldots$) (Exclusion, $a, b, \ldots$)

Table 7: Summary of constraints to check, based on applied changes in template and instances (Ly et al., 2008).

Summary

This section has provided a framework which describes how to determine what parts of a workflow process need to be checked for semantic issues. Semantic checks can be executed by analyzing the constraints which are included on the template and instance level.

Two different types of constraints are defined: mutual exclusion and dependency constraints. Changes at the instance level can occur in three different ways: by inserting, deleting or moving one or multiple tasks. Each of this three modifications has its own consequences with their associated semantic checks which need to be performed. Changes at the template level follow the same procedure. But the adaption of the related instances follows a different procedure, since some instances can be individually modified. When updating these individual cases to the new template, six different types of biases of instances can be distinguished: unbiased, equivalent bias, ΔI subsumes ΔS bias, ΔS subsumes ΔI bias, partially equivalent bias and the disjoint bias. The last two types of biases require additional semantic checks. The partially equivalent bias only needs to check the alterations which differ from the template changes and which affect the involved constraints. Instances in the disjoint bias need semantic checks for all the alterations which affect the involved constraints.

4. Delphi Panel

This chapter elaborates on the Delphi Panel performed for the empirical input of this thesis. As stated in paragraph 1.4.3, the properties of a Delphi panel match the requirements of the empirical research of this thesis. The ability to measure and improve consensus amongst the participants, the room for discussion and the flexibility are a few of these properties which advocate for the use of a Delphi panel. Additional motivation for the selection of the Delphi panel is provided in paragraph 4.1.2.

The questioning in this Delphi Panel is based on the literature frameworks on workflow commissioning and semantic checking as presented in chapter 3. The workflow patterns as described by van der Aalst et al. (2003) are used to analyze the improvement opportunities in Flowmanager. The six different situations in workflow processes and their associated degree of semantic checking as described Ly et al. (2008) are used to map the opinions of the daily users of Flowmanager on the application of semantic checks in practice.

This chapter has the following outline. First, basic information about the use of Delphi Panels is provided followed by a motivation on the application in this thesis. Second, the method provided by Okoli and Pawlowski (2004) is displayed, which is used for the Flowmanager Delphi Panel. Third, the configuration of the Flowmanager Delphi Panel is provided. Fourth and finally, the results of the Flowmanager Delphi Panel are presented followed by an analysis of these results.

4.1. About Delphi Panels

In this section, basic information is provided on the properties of Delphi Panels.

4.1.1. Basic information

In their article, Okoli and Pawlowski (2004) provide a description of Delphi Panels which hold most of the characteristics of a Delphi Panel: *"Delphi may be characterized as a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem. To accomplish this "structured communication" there is provided: some feedback of individual contributions of information and knowledge; some assessment of the group judgment or view; some opportunity for individuals to revise views; and some degree of anonymity for the individual responses."*

A Delphi Panel is a "*widely used method to obtain input from a group of experts"* (Diamond et al., 2014). In short, a Delphi Panel is a qualitative empirical research method which uses a series of questionnaires and controlled feedback to let a group of experts arrive at consensus about a specific subject.

4.1.2. Motivation

A number of reasons have led to the selection of the Delphi Panel method:

- This thesis is in need of a consensus of clients on improvements of Flowmanager in order to answer the research question. A Delphi Panel provides room for discussion and the needed methodology, in contrary to a panel study. A panel study tends to focus more on the changes in individuals' opinions without discussing the subjects at hand (Okoli & Pawlowski, 2004).
- A Delphi Panel matches the number of experts available, it is designed for relatively small groups. Okoli and Pawlowski (2004) and Worrell, Di Gangi, and Bush (2013) both indicate that ten is the minimum number of participants for a Delphi panel. It is very difficult for smaller group to reach consensus.
- From a practical point of view, it is not necessary for the experts to meet physically. Taking into account the diverse locations of the experts and the limited time they have available, a Delphi Panel is a suitable solution.

- Delphi Panel are flexible in their design, there is a wide variety of tools available.
- Delphi Panels are flexible in their duration, the rounds can be executed in one afternoon or once a week.
- Delphi panels can be executed anonymously easily, which improves contribution of participants.

As stated by Diamond et al. (2014) in their conclusion, there is no commonly accepted definition of achieving consensus with a Delphi Panel. Okoli and Pawlowski (2004) indicate that using Kendall's W (a statistical method for measuring consensus in a group) could be a valuable measurement of the level of consensus. However, this method requires a number of participants which is higher than the eleven participants available for the Flowmanager Delphi Panel. Paragraph 3.3.1 elaborates further on the participating experts. Although it is not possible to prove statistical significance with such a small number of participants, the outcome of this Delphi Panel provides valuable information about the improvements of Flowmanager desired by the customers.

4.2. Methodology of a Delphi Panel

In this section, the methodology is described which is used to execute the Delphi Panel, based on the design by Okoli and Pawlowski (2004). The following procedures need to be completed in the execution of a Delphi Panel:

Selection of experts

Okoli and Pawlowski (2004) provide an extensive method for selecting the appropriate experts and placing them into different panels, based on their skills, possessed knowledge on the subject at hand and the type of organization they are involved in (Academics, Practitioners, Government and NGO's). Next, the authors provide steps for approaching the experts for the first time and eventually inviting them for the Delphi Panel.

Data collection configuration

Two important configuration steps are discussed in this section. First a selection has to be made of the mechanisms which assist in the execution of the Delphi Panel. These mechanisms can vary from specific software to a simple e-mail or even a fax. Second, the design of the questionnaire is selected and put into place.

Administration phases

During the administration phases, the Delphi Panel is executed. Three different phases can be distinguished:

Phase 1: Brainstorming

In this phase two questionnaires need to be filled out by the participants. In the first questionnaire, the experts are asked to provide factors they find relevant for the subject at hand. The second questionnaire has grouped the answers of the first round and asks the experts if their answers have been interpreted correctly.

Phase 2: Narrowing down factors

In the second phase, the experts are asked to make a selection of answers of the lists that previously have been composed. After completion, the researcher selects answers based on the frequency of selection of the items (50% is often used as a minimum). The number of items selected for the next round is limited, depending on the Delphi Panel, subject and total number of possible answers.

Phase 3: Ranking relevant factors

In the third phase, the experts are presented the results of the previous round and they are asked to prioritize the items based on their own preferences. The experts also need to provide comments to justify their rank.

At this stage, Kendall's W (value between 0 and 1) is used to measure consensus. A Kendall's W value of 0.7 or greater indicates that consensus has been reached, the Delphi Panel can be marked as completed. If Kendall's W is lower than 0.7, the questionnaire has to be resent to the experts. For this thesis, Kendall´s W is not a useful indication of consensus achievement, due to the relatively small amount of participants. The results of this Delphi panel cannot be assessed on their significance. Therefore, estimates on the amount of consensus are made based on logic reasoning and analysis of the answer provided by the experts.

If resent, the questionnaire is accompanied by additional information which can aid the experts in the revision of their rankings:

- The average ranks of the items of the panel,
- The expert's own ranking in last round,
- The comments provided by all the experts and
- An indication of current consensus (normally Kendall's W, in our case an estimate) (Okoli & Pawlowski, 2004).

This process repeats itself until one of the following three stopping criteria has been met:

- Kendall's W reaches a value of 0.7 or more,
- The amount of revision rounds exceeds the amount of rounds stated as a maximum at the beginning of the panel, or
- The value of W does not improve substantially between the rounds (Okoli & Pawlowski, 2004).

4.3. Methodology of the Flowmanager Delphi Panel

Applying the method of Okoli and Pawlowski (2004) to the Flowmanager Delphi Panel leads to the following configuration.

4.3.1. Selection of experts

The selection process of the Flowmanager Delphi Panel experts was relatively simple compared to the complex procedure as described by Okoli and Pawlowski (2004). An expert for the Flowmanager Delphi Panel should have experience with the commissioning of workflows in Flowmanager and should have some basic knowledge about semantic checking. With the assistance of several BuildIT employees, a number of experts have been selected. All of the experts have agreed in participating in the Delphi Panel. In total five internal experts (employed at BuildIT) and six external experts (employed at customers of BuildIT) are participating in the Flowmanager Delphi Panel. The daily tasks of the internal experts that are participating is to commission workflows for customers of BuildIT, since some customers do not have the knowledge to perform these tasks. This makes the internal experts appropriate participants to the Delphi panel.

Taking into account the limited amount of experts, all the experts are placed in the same panel. This means that they receive the exact same questions.

4.3.2. Data collection configuration

Spilter

For the collection of data in the Flowmanager Delphi Panel, the Spilter Decision room software[2] is used (www.spilter.nl). Spilter is a company specialized in providing a large variety of group decision making software to small and large companies (Spilter). Due to the connections of Dr. J. van Hillegersberg and the University of Twente, Spilter is available free of charge for research purposes and thus for the execution of the Flowmanager Delphi Panel[3].

Spilter provides a promising environment in which a Delphi Panel can be conducted:

- Multiple rounds can be created.
- A large variety of different questionnaires is available, ranging from open question to prioritization questions.
- Every participant has his own account.
- Participants can be contacted through Spilter's e-mail feature.
- Typical Delphi Panel aspects can be put into practice, such as comparing personal answers with other expert's answers.
- Results are presented in reports which can easily be adapted and implemented.

Overall, it can be concluded that Spilter is easy to use in the managing of a Delphi Panel and its participants.

Design

The Flowmanager Delphi Panel is executed as follows. The Flowmanager Delphi Panel consists of three rounds. In the first round Flowmanager is analyzed on possible additions that can be made in the field of workflow commissioning. In the second round experts need to make a selection of the most useful additions to Flowmanager for workflow commissioning and need to select the situations which require semantic checks. In the third round experts need to prioritize the selections they have made in the second round. The third round can be repeated if no consensus has been reached. The 'Administration phases' section in paragraph 4.3.3 provides a further elaboration on the three phases.

Furthermore, the following design decisions are implemented:

- Due to the limited availability of the experts, the rounds of the Flowmanager Delphi Panel are open for experts for an entire week. Experts are free to participate when it is most convenient for them.
- Due to the limited availability of the experts, the amount of revisions of the third round is limited to three.
- The topics of workflow commissioning and semantic checking are processed separately.
- The stopping criteria as described by Okoli and Pawlowski (2004) in paragraph 3.2 apply to the Flowmanager Delphi Panel.

4.3.3. Administration phases

Phase 1: Brainstorming

After thorough consideration, the decision has been made not to let the experts execute this phase for the Flowmanager Delphi Panel. The Delphi Panel is already very time consuming for the experts even without including this phase. So, in order not to be too demanding, I have carried out this phase with

[2] http://www.spilter.nl
[3] http://ut.spilter.nl

my supervisors at BuildIT. Both supervisors work with Flowmanager daily and have plenty of knowledge on the topics at hand.

Only the subject of workflow commissioning is relevant for the brainstorming phase. Flowmanager needs to be analyzed on its functionalities, based on the different patterns as described by van der Aalst et al. (2003). The literature review on semantic checking has provided a list of six suggestions for implementing semantic checking which is ready for phase 2.

Phase 2: Narrowing down factors

For both the topics of workflow commissioning and semantic checking, the participants are asked to indicate whether they find the provided additions to Flowmanager useful or not. Obviously, first of all experts receive an e-mail with instructions and their login details. The first thing experts see after logging in, is a welcoming message and a repeating of the most important instructions, as shown in Appendix E.

After clicking on Next, the experts are provided with a short introduction into Workflow patterns, after which they are presented ten separate questions. Each question is tied to one of the ten workflow patterns concluded from the previous phase. The question is accompanied by an explanation of the pattern and a simplified example. An example of a question is provided in Appendix F.

After answering these ten questions, the experts move on to the topic of semantic checking. The same structure is used as with workflow patterns: The subject is introduced and followed by six questions, matching the situations as described by Ly et al. (2008). The questions are again accompanied by a situation description and an example. Experts were provided a process on template level and several instances which fall under the template level and were modified individually. The template level required an update, which could have consequences for the modified instances. Experts needed to indicate whether they would make use of semantic checking in the described situations.

Phase 3: Ranking relevant factors

After phase two is concluded and an analysis is performed on the results, phase three is initiated. In this phase, experts need to prioritize the workflow patterns and semantic checks which were seen as most relevant in phase 2 (more than 50%). Both topics have the same type of questions. In order to prevent the presence of abundant information in this section, only the setting of workflow patterns is illustrated. The setting of semantic checking has the same properties.

Again, experts receive an e-mail containing instructions. When logging in, a welcoming message repeats the most relevant instructions. After the experts have read the introduction they can proceed to the first task: the prioritization of additions to the workflow commissioning of Flowmanager and providing a motivation for the made prioritization. In order to aid the experts with the answering of the questions in the first round of this phase, additional information on the subject at hand is provided as a reminder. An example of a prioritization question can be found in Appendix G.

After the completion of the prioritization on workflow patterns, experts are asked to perform a MoSCoW analysis on the workflow patterns. Where the prioritization provides useful information on relative importance of the additional of certain workflow patterns, it does not indicate the individual need for a certain solution (Hatton & Society, 2008). The MoSCoW method is a commonly used method in the field of information systems, indicating the importance of a feature. Experts need to place the patterns in one of the following four groups as indicated by Hatton and Society (2008):

- '**M**ust have': An absolute essential feature.

- '**S**hould have': Features in this group would be nice to have.
- '**C**ould have': Features in this group are interesting, but have no priority.
- '**W**on't have': Features which are not unimportant.

An example of a MoSCoW analysis question is provided in Appendix H. After the completion of the MoSCoW analysis of workflow patterns, the experts move on to semantic checking. For this topic, the exact same type questions need to be answered as for workflow patterns, as stated in the introduction of this paragraph.

After the first round of phase three, one or possibly two additional rounds can be performed, depending on the outcome of the first round (indication on the amount of consensus). Should the second and third round take place, they will merely consist of prioritization, since the MoSCoW analysis has already been completed. In order to aid the experts with the revision of their rankings, they receive additional information as stated by Okoli and Pawlowski (2004) in paragraph 4.2.

4.4. Results of the Flowmanager Delphi Panel

This section presents the results of the conducted Delphi Panel, based on the methodology as provided by Okoli and Pawlowski (2004). The results are supported by various quotes, provided by the experts.

4.4.1. Phase 1: Brainstorming

The analysis as described in phase one of paragraph 4.3.3, delivers the following results for Flowmanager, as presented in table 8. A + indicates that a pattern is available in Flowmanager. A – indicates that a pattern is absent in Flowmanager. Conclusively, ten of the twenty patterns are not available in Flowmanager, these are shown in bold. These patterns are used in the next phases of the Delphi Panel.

	Sequence	**Parallel split**	**Synchronization**	Exclusive choice	Simple merge	**Multi choice**	**Synchronizing merge**	**Multi merge**	**Discriminator**	Arbitrary cycles	Implicit termination	**M.i. w/o synchronization**	**M.i. w/ a priori design time**	**M.i. w/ a priori runtime**	**M.i. w/o a priori runtime**	Deferred choice	Interleaved parallel routing	Milestone	Cancel activity	Cancel case
Flowmanager	+	-	-	+	+	-	-	-	-	+	+	-	-	-	-	+	+	+	+	+

Table 8: Applying the framework by van der Aalst et al. (2003) to Flowmanager.

4.4.2. Phase 2: Narrowing down factors

As described in paragraph 4.3.2, the Delphi Panel is divided in two sections: workflow commissioning and semantic checking. First, the results of the workflow commissioning part are discussed, followed by semantic checking.

Workflow commissioning

The experts were asked to indicate whether the patterns selected in phase 1 would be of added value to the commissioning of workflows in Flowmanager. Figure 28 shows the results provided by the participants. A clear division between two groups of possible additions can be observed. The first five additions in figure 28 are rated as useful additions to Flowmanager by at least 5 of the 11 experts: "*A parallel split can be useful when you need to wait for another person (customer, middleman or internal department), the process will be able to continue instead of being paused."* The latter five additions

are graded as non-relevant additions to Flowmanager, experts see no relevance or use for an implementation in Flowmanager: *"There is no added value for this addition." "There is no situation where this type of additions can be applied."*

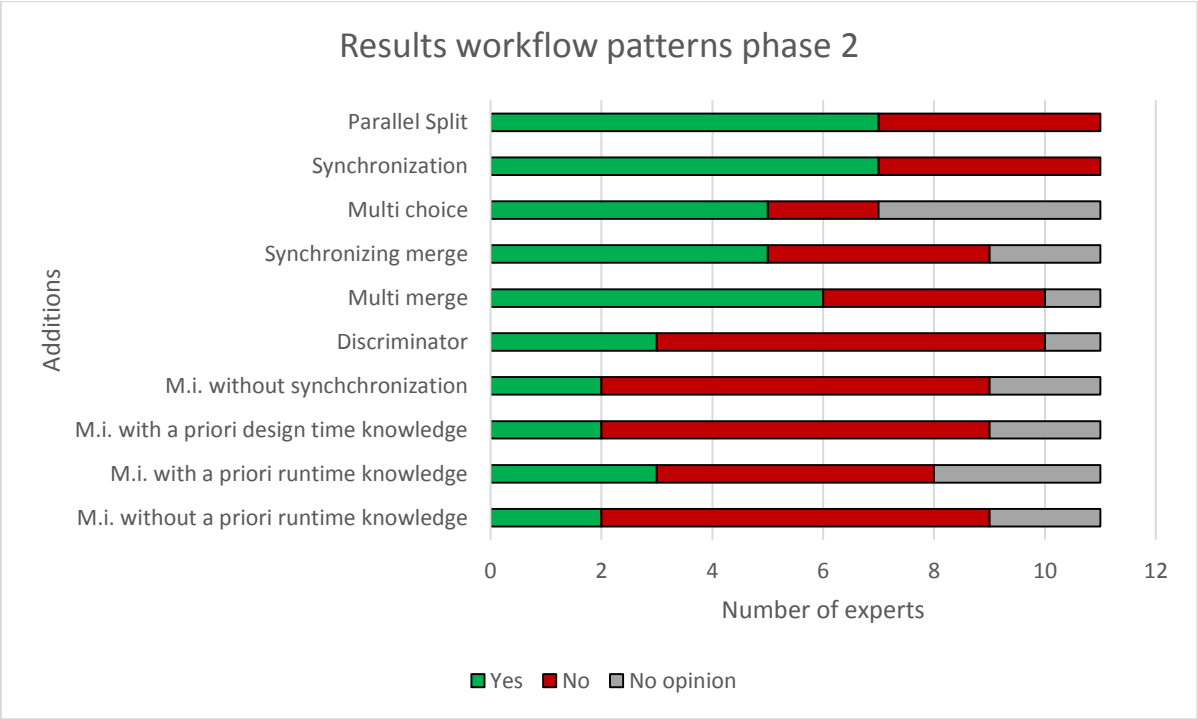

Figure 27: Results of the workflow commissioning questionnaire in phase 2.

For Parallel split, Synchronization and Multi merge a clear case can be made for including them in the next phase of the Delphi panel. A majority of the experts has indicated these three options as useful additions to Flowmanager. Although both Multi choice and Synchronizing merge are selected by just five experts, they need to be included in the remainder of the Delphi panel. The experts which answered with ´No opinion´ indicate in their motivation that a combination of these patterns can provide a useful addition: "*This addition is necessary when Multi choice is introduced." "This option is only relevant when Multi choice is introduced."* This correction of the results of the experts answering 'No opinion' based on their provided motivation, results in a majority acknowledging the potential of Multi choice and Synchronizing merge.

Conclusively, Parallel split, Synchronization, Multi choice, Synchronizing merge and Multi merge are selected for the next phase of the Delphi Panel. Although some experts have indicated that certain patterns are only useful when they are combined (as stated in the previous paragraph), the different patterns are handled separately in the next phases. Reason for this choice in configuration is that the majority of experts have indicated the relevance of patterns individually. However, the combination of two relevant patterns as indicated by a number of experts should be taken into account in the discussion and recommendations to BuildIT.

Semantic checking

For semantic checking, the experts needed to answer six questions which are related to the six situations for potential semantic checking as described by Ly et al. (2008). Background information on the functioning and reasoning of semantic checks can be found in paragraph 3.2 of this thesis. The experts were asked to indicate whether they would use semantic checking in the different situations. Figure 29 shows the results provided by the participants.

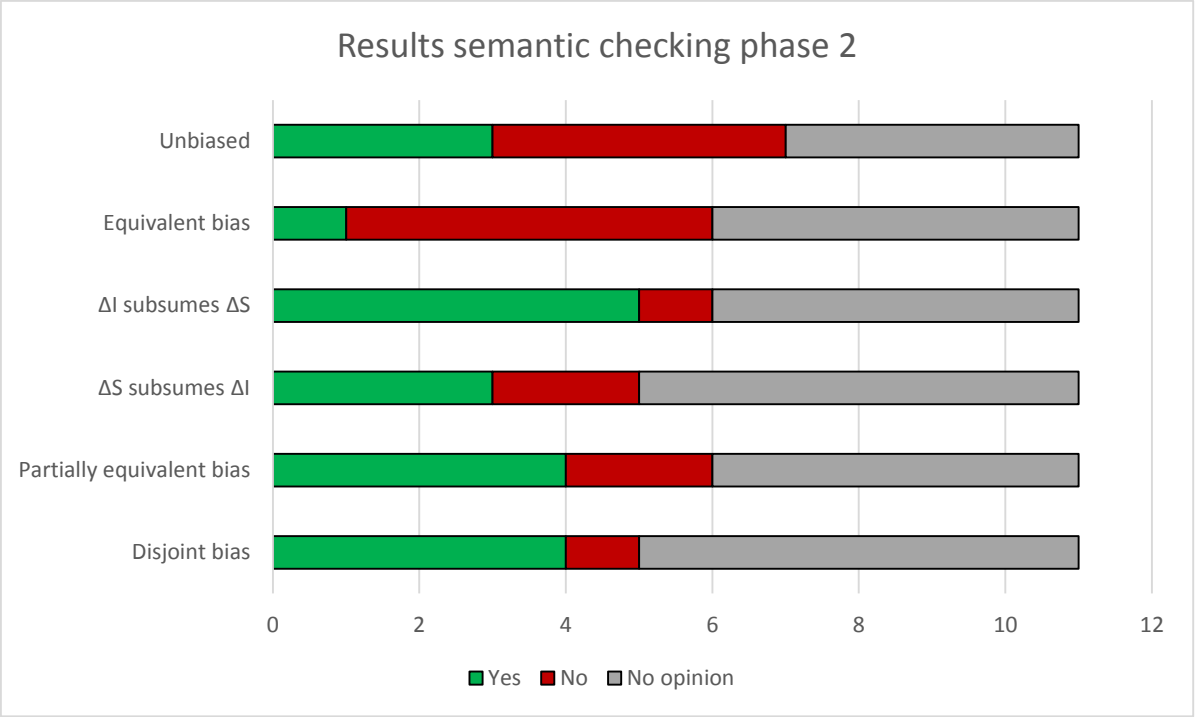

Figure 28: Results of the semantic checking questionnaire in phase 2.

The first observation that really stands out is the large number of experts that have selected the 'No opinion' option for all of the six questions. For every situation, at least five of the eleven experts indicated that they have no opinion. An examination of the motivation for their choice provides results which can be summarized in the following quotes:

- *"I have not yet indulged myself in semantic checking."*
- *"I cannot imagine how this feature could be applied in practice."*
- *"I do not understand the subject/question."*

Although the experts have been selected for their knowledge on the subjects at hand, the subject of semantic checking appears to be too complex for five of the eleven experts. With almost 50% of the respondents not being able to answer the questions due to the aforementioned reasons, the results have become blurred. The results of the experts who do have knowledge about semantic checks cannot be assessed properly.

If we take out the experts which answered 'No opinion' and merely focus on the ones which either provided 'Yes' or 'No', the results as shown in figure 30 emerge. However, these results represent a limited group of experts. For the remainder of the Delphi Panel and this thesis, the limited number of participants in this part of the Delphi panel needs to be taken into account when conclusions are drawn.

Looking at figure 30, a distinction can be made between two groups. The first group consists of situations in which semantic checks appear to be redundant: A majority of the answered 'No' for these situations. Two types of instances are situated in this group, being Unbiased and Equivalent bias. The second group logically consists of situations in which semantic checks are desired, a majority answered 'Yes' for these situations. The remaining four types of instances are placed into this group: ΔI subsumes ΔS, ΔS subsumes ΔI, Partially equivalent bias and the Disjoint bias.

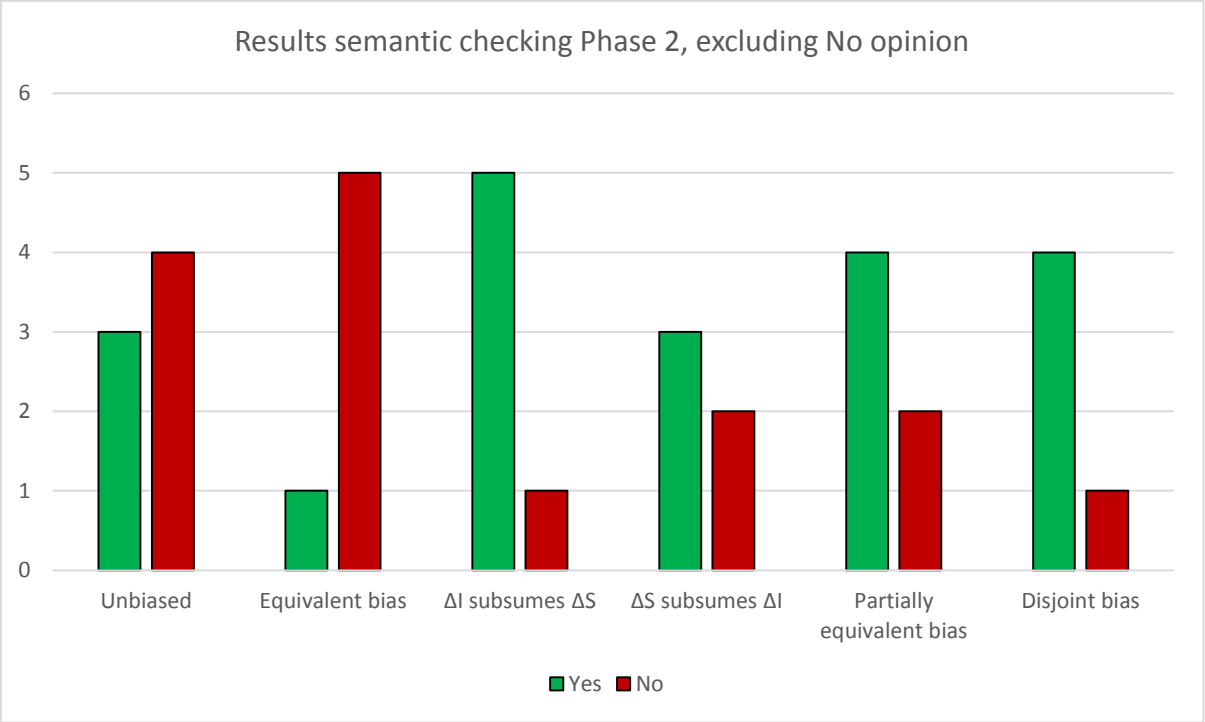

Figure 29: Results of the semantic checking questionnaire in phase 2, excluding no opinion.

These results provide some interesting data for the discussion. In their article Ly et al. (2008) suggest that semantic checks are only useful for instances following the Partially equivalent bias and the Disjoint bias. The participants of the Delphi panel however, indicate that there are two additional situations that should be accompanied by semantic checks, being ΔI subsumes ΔS and ΔS subsumes ΔI. These differences are discussed in chapter 5.

Conclusively, the group containing ΔI subsumes ΔS, ΔS subsumes ΔI, Partially equivalent bias and the Disjoint bias is selected for the next phase of the Delphi Panel. As stated by Okoli and Pawlowski (2004) and Worrell et al. (2013), it is difficult to achieve consensus with a panel with less than 10 participants. However, the prioritization of the four previously mentioned situations is a low effort task for the experts. Therefore, the six participants who were able to answer either a 'Yes' or a 'No' in this round, are asked to participate in the prioritization in the next round for semantic checking. The other participants will skip this part of the panel. After the first round of phase three an evaluation takes place on the usefulness of this prioritization. This evaluation indicates whether the prioritization of situations is continued or cancelled for the succeeding round(s).

4.4.3. Phase 3: Ranking relevant factors

In this phase, the experts were asked to create a prioritization and perform a MoSCoW analysis for both the research topics workflow commissioning and semantic checking. Two rounds of prioritization were executed in this phase. The results of each round are discussed separately.

For the remainder of the Delphi Panel, the number of respondents is decreased from eleven to ten, one of the experts failed to respond to the rounds of phase three.

Round 1

Workflow commissioning

The prioritization of the five possible workflow additions as described in the previous phase has led to the results as presented in figure 31. On the left, the different possible additions are displayed. The

yellow bars represent a graphical ranking of the additions. The additions are ranked as follows: Multi choice is at the top and has the highest average ranking of 2.3 out of 5. Multi merge comes in last with a ranking of 3.5.

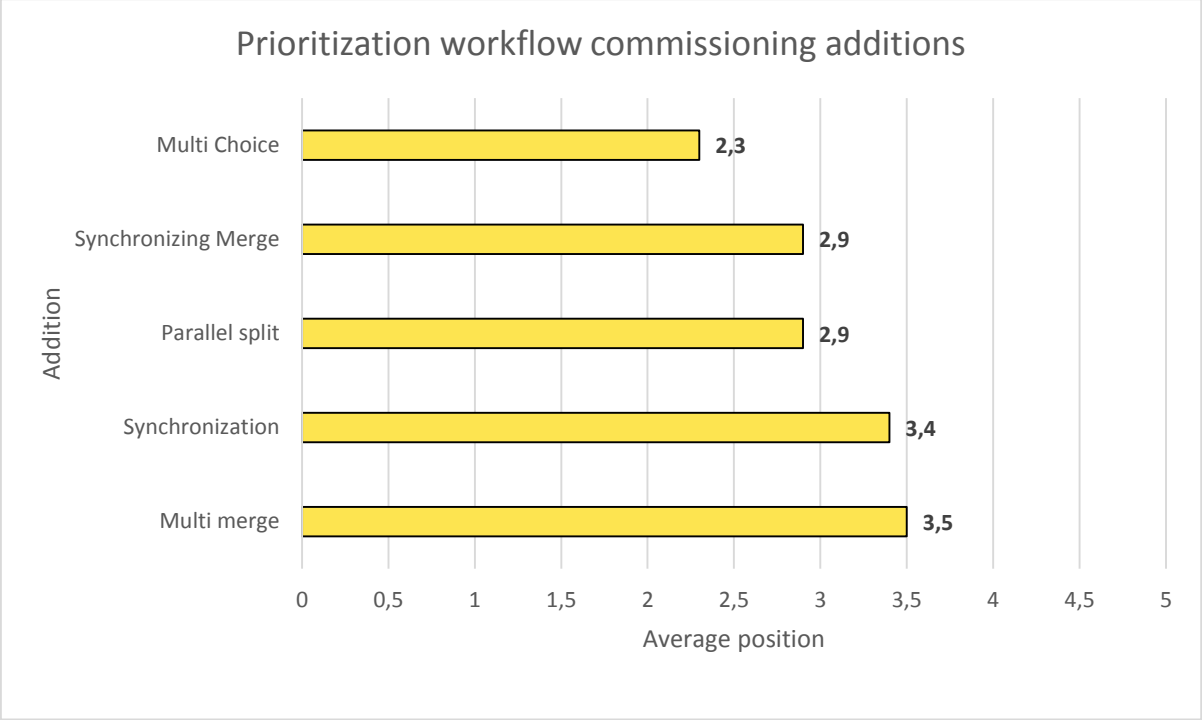

Figure 30: Results prioritization workflow additions (phase three, round one).

The results of the first round do not provide a clear ranking yet. One can clearly tell that Multi choice is the mostly preferred addition: "*I have ranked Multi choice above Parallel split, because it is more flexible." "If possible, I would make most use of Multi choice. We have a lot of tasks that could follow this structure, such as calling and cancelling tasks." "The Multi choice combined with synchronizing merge is more versatile than Parallel split combined with synchronization."*

For the positions two and three, there is a tie between Synchronizing merge and Parallel split. Between position four and five, only a minimal difference exists between Synchronization and Multi merge. These results call for another round of prioritization, no conclusion can be derived from these minimal differences.

The MoSCoW analysis which the experts performed in this round is presented in figure 32. The individual needs for the additions indicated by the experts show resemblances to the relative prioritization: the higher the average prioritization ranking, the more the addition is seen as a 'Must have' or a 'Should have'. Multi choice for example is rated as the most important addition in this round of the prioritization. This addition also received the highest 'Must have' ranking. Another example is Multi merge, ranked last in the prioritization. This addition received no 'Must have' indications. The MoSCoW analysis provides some interesting input for a comparison of its results with the results derived from the final round of workflow addition prioritization. These results are discussed in chapter five.

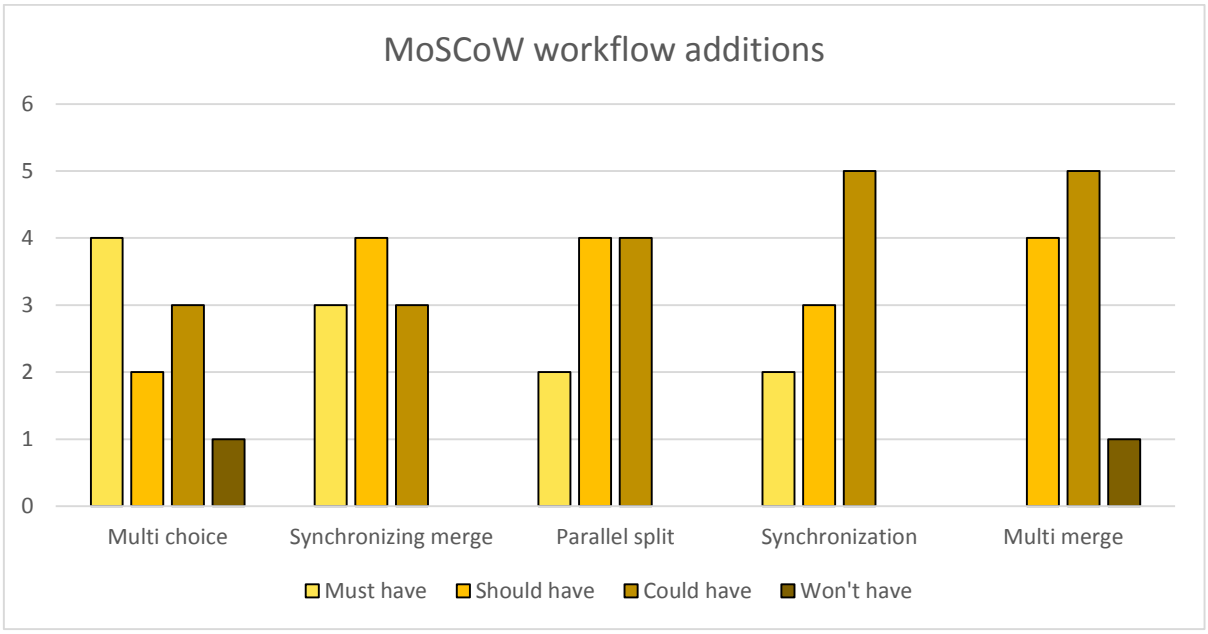

Figure 31: Results prioritization workflow additions (phase three, round one).

Semantic checking

The prioritization of the four situations as described in the previous phase has led to the results as displayed in figure 33. As stated in the previous phase, it is difficult to reach consensus with a small number of experts. Looking at the results in figure 33, this also seems the case for this part of the Flowmanager Delphi Panel.

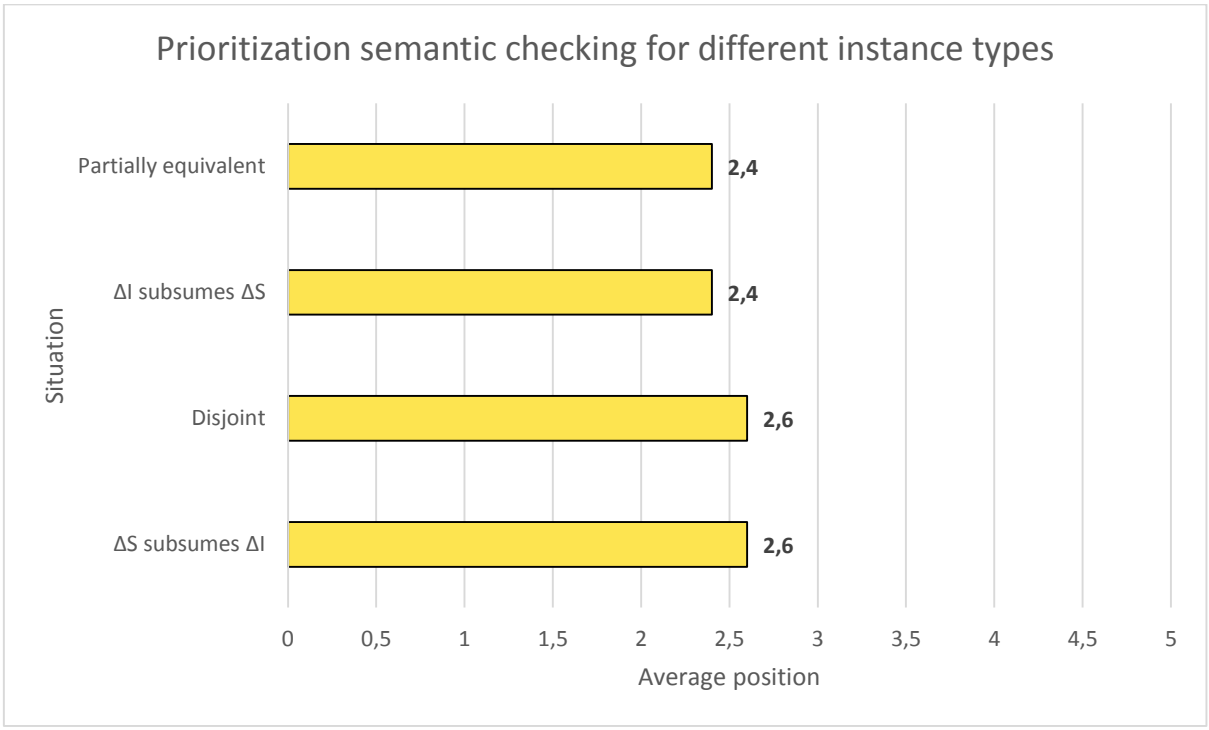

Figure 32: Results prioritization semantic checks (phase three, round one).

The experts that were selected to prioritize the situations, answered in such a way that the average ranking of all the situations are very close to each other. This means that when Expert A gave a high priority to Situation 3 and a low priority to Situation 6, it is nullified by a complete opposite opinion of

Expert B. Whereas the workflow additions prioritization shows some agreement among the experts and also shows some room for improvement in consensus, the semantic checking prioritization shows no agreement with every situation ranking between 2.4 and 2.6. This indicates that consensus among the experts is not achievable for this subject.

Even though the experts have been selected because they have certain knowledge about semantic checks, four out of six experts exemplify that they are not comfortable enough with the subject of semantic checking to make a solid judgment on the prioritization. *"It is difficult for me to make a decision, I do not have much experience yet with semantic checking.", "I don´t know that much about semantic checking."*

We can conclude that no reliable conclusions can be drawn from this part of the Delphi panel, based on the following arguments:

- The limited amount of experts.
- The provided ranking with little perspective for better results.
- The associated motivation which describes on what grounds the prioritization has been executed by the experts.

Therefore, the subject of semantic checking requires no additional prioritization rounds. The next round of the Flowmanager Delphi panel merely consists of a prioritization round for the workflow additions.

Round 2

As stated, the experts only execute prioritization for the workflow additions in this round of the Delphi panel. This prioritization has led to the results as shown in figure 34.

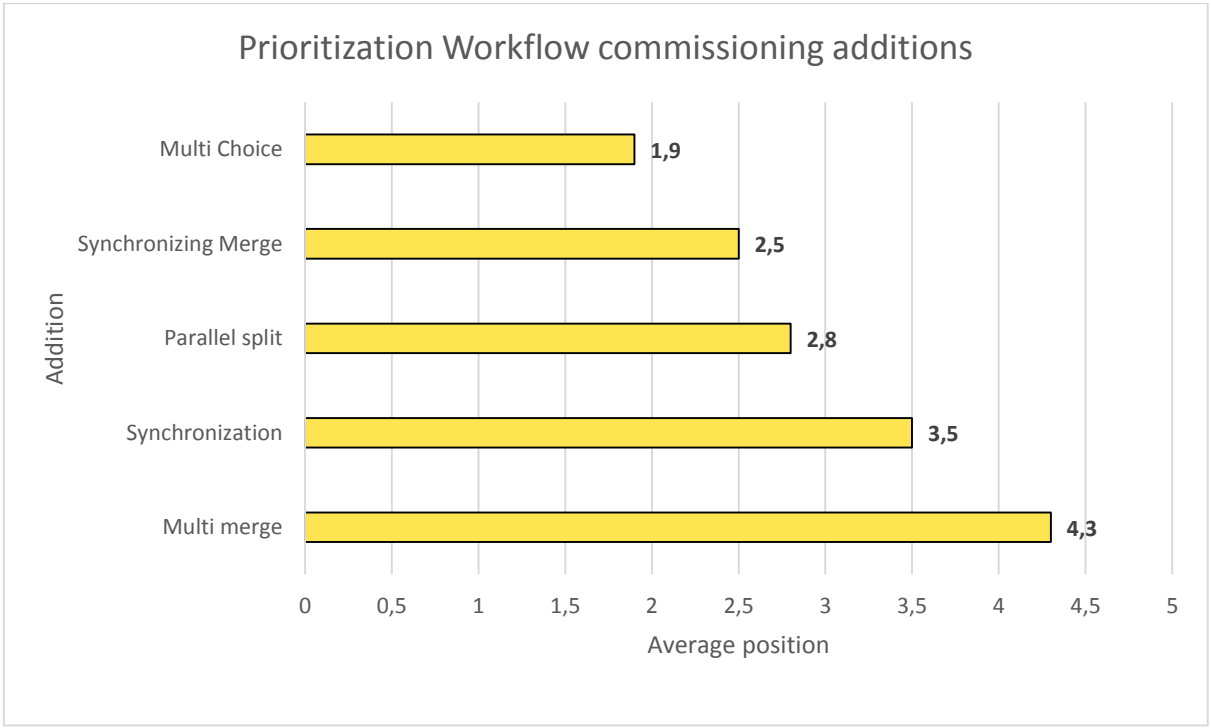

Figure 33: Results prioritization Workflow additions (phase three, round two).

The outcome of this round shows the same ranking as the previous round, with Multi choice as the addition which has received the highest average ranking and Multi merge at the bottom with the

lowest average ranking. However, there is a bigger distinction between the possible additions in this round. Whereas in the first round the scores of the additions ranked two to five were very close to each other, the second round shows larger differences and a clearer ranking. The average scores show a wider spread in the scores. The first round produced scores relatively close to 2.5 (ranging from 2.3 to 3.5). The second round shows scores which are more divergent, ranging from 1.9 to 4.3. This indicates that the experts have reached a higher degree of consensus, by taking into account the average ranking and motivations of other experts from the first round in their own prioritization of the second round.

A good example to explain this principle is the Multi merge addition. In the first round its average score was 3.5. This means that there was no real consensus on this addition among the experts. Some experts have probably rated Multi merge as important, some as average and some as not important, leading to a score relatively close to the middle of 2.5. In the second round, the average score for Multi merge was 4.3. This shows that overall, more experts have given Multi merge the same relatively low prioritization which has led to the last position in the ranking.

Conclusively, the second round of phase three of the Delphi panel has provided us with results which are a useful contribution to the discussion. Several experts have reviewed their prioritization which has led to a clearer ranking of possible additions. No further rounds of the Delphi panel are executed, due to the following two reasons:

- Although we have argued that there is a lot of improvement on consensus between the first and second round, it is unlikely that an extra round provides further improvement. When we look at the motivations that a number of experts provided, it is stated that despite the provided information on the average ranking and the associated motivation, several experts decided not to alter their ranking. Several experts indicated that they were not influenced by the arguments of others: "*I have made the ranking based on the same arguments as the previous rounds"*. *"I have entered the same prioritization, despite the motivation of others."* Since a number of experts already have altered their ranking and others are resolute in maintaining their original ranking, real improvements in consensus are not to be expected.
- Despite the shortened round (exclusion of semantic checking), it proved to be difficult to get the experts involved in this round. In order to gather all the prioritizations the second round of phase three was open for two weeks instead of one week as applied to the previous round and phase. Experts indicated by e-mail that they did not have enough time to complete the Flowmanager Delphi panel. An additional round would have two negative consequences. First, an extended round of two weeks or more needs to be introduced which would delay the process of this thesis. Second, there is a risk of experts not participating properly in the panel due to time constraints. This 'cutting corners' behavior could negatively influence the results which are used in the discussion Day and Bobeva (2005).

In short, an additional round of prioritization is not executed for two reasons. First, no significant improvement is expected, taking into account the limited improvement in consensus between the first two rounds of prioritization. Second, another round would be too time consuming due to the limited availability of the experts and could lead to experts not participating properly.

Overall, we can conclude that the Delphi panel has been executed successfully. Experts were involved in selection and prioritization of workflow patterns in three rounds and in selection and prioritization of semantic checking in two rounds. The rounds have resulted in a clear prioritization of workflow patterns which can be introduced for the topic of workflow commissioning. For semantic checking the

results are less explicit, although input for the discussion has certainly been provided. The next chapter analyzes and discusses the results of the Flowmanager Delphi panel.

5. Discussion

The goal of this chapter is to analyze the results of the Flowmanager Delphi panel (paragraph 5.1) and to discuss the commonalities and differences between the theoretical concepts and the results of the Flowmanager Delphi panel (paragraph 5.2). Consistent with the previous chapters, the two research topics of workflows commissioning and semantic checking are discussed separately.

5.1. Analysis of the Delphi panel results

This paragraph presents the findings which can be derived from the results of the executed Delphi panel as presented in chapter 4.

5.1.1. Workflow commissioning

As stated in the previous chapter, the participants were asked to select and prioritize a number of workflow patterns provided by van der Aalst et al. (2003), which are currently unavailable in Flowmanager. After one selection round and two prioritization rounds, the results as displayed in figure 34 were accomplished. These results are accompanied by the outcome of the MoSCoW analysis, as presented in figure 32.

The prioritization and the MoSCoW analysis have resulted in the same ranking of additional workflow patterns. The patterns which were given a higher prioritization, were also given a higher degree of urgency in the MoSCoW analysis. The combined conclusion of these two methods provides a solid foundation for recommendations on additions to Flowmanager's workflow commissioning.

An interesting observation is that the patterns which are related to each other are next to each other in the ranking. Multi choice is the possible diverging of one activities into several activities, synchronizing merge is a pattern associated with the possible converging of these activities towards a single activity. Also, a Parallel split enables multiple processes to be performed simultaneously, whereas the synchronization patterns synchronizes multiple processes towards one process. The individual or pairwise implementation of these patterns is discussed in paragraph 5.2.

Overall, a trend can be spotted among the several rounds of the Delphi panel: experts are especially interested in additions which focus on the simultaneous execution of workflow activities.

5.1.2. Semantic checking

The participants were asked to indicate whether they would add semantic checks in several situations, as described in chapter 4. The subject of semantic checking has proven to be a difficult subject for the participants, not all participants were able to answer the questions due to aforementioned reasons.

The Delphi panel for this subject lasted one selection round and one prioritization round. As discussed in the previous chapter, the results of the prioritization showed no chance of reaching reliable consensus. However, the results from the selection round provide some interesting discussion material. As shown in figure 30, a majority of the participants have indicated that in four out of six situations semantic checks are required. These results differ with the theory by Ly et al. (2008), which indicates that only the partially equivalent bias and the disjoint bias require additional semantic checks. This difference is one of the points of discussion in paragraph 5.2.

5.2. Comparing theoretical concepts to Delphi panel results

This section discusses the commonalities and difference between the concepts derived from theory and the results of the executed Delphi panel.

5.2.1. Workflow commissioning

As stated in the previous paragraph, a clear prioritization of desired patterns has been derived from the Delphi panel. The MoSCoW analysis of these patterns supports this prioritization, the higher the ranking, the more the patterns is determined as a 'Must have' or a 'Should have'.

As mentioned in chapter four, an interesting point of discussion is the combined development of patterns. Two potential pairs of patterns can be identified among the five patterns which were prioritized: Multi choice combined with Synchronizing merge and Parallel split combined with Synchronization. The development and implementation of single patterns should lead to a shorter delivery time and choosing for a combined development leads to the delivery of a more complete product.

The decisive factor in this choice is the provided prioritization by the participants combined with the results of the MoSCoW analysis. These two factors show that the ratings of the two pairs of patterns are very close to each other, a potential combination of the two sets of patterns is in line with the provided rankings. On the first and second place of the ranking the combination of Multi choice and Synchronizing merge can be found. Also, these two patterns have a very similar score in the MoSCoW analysis, indicating that the importance of their development is close to each other. The same goes for the second pair of patterns, Parallel split and Synchronization. These are positioned on the third and fourth place in the prioritization and also have similar MoSCoW ratings.

Based on the argumentation as provided above, the decision is made to combine the two sets of patterns for recommendations on development and implementation, leading to a prioritization consisting of three items:

1) A combination of Multi choice and Synchronizing merge.
2) A combination of Parallel split and Synchronization.
3) Multi Merge.

Another argument which needs to be taken into account, is the presence of the presented patterns in other WfMS's as indicated in table 5. Two of the five patterns as described above are relatively uncommon in other WfMS and thus provide BuildIT with an opportunity to distinguish Flowmanager from other WfMS. These two patterns are Synchronizing merge and Multi merge. Since Synchronizing merge is already ranked first since the combining of the patterns, there is no need to give this pattern a higher priority.

For Multi merge, one could argue that it should receive a higher priority due to the distinguishing opportunities that come along with this pattern. However, we need to take into account the results of the Delphi panel at this point. The participants have clearly indicated that Multi merge is the least interesting pattern to add to Flowmanager with an average position of 4.3 out of 5.0. This is also supported by the MoSCoW analysis: no participants have indicated Multi merge as a 'Must have' and a majority have indicated Multi merge as either 'Could have' or 'Won't have'. The need for this pattern as indicated by the MoSCoW analysis is much lower than the needs for other patterns. Taking into account this strong common opinion of clients on the use of Multi merge, we decide not to alter the ranking of the patterns as described above.

Conclusively, a case for the combination of two sets of two patterns and one single pattern is made in this paragraph, after which the prioritization resulting from the Delphi panel is evaluated, based on the distinguishing possibilities of each pattern as derived from theory. Several questions arise after this conclusion concerning the next steps that need to be taken in the realization of the implementation of these patterns. At this moment, Flowmanager is only able to commission workflows which are

sequential. The use of parallel and/or multiple activities is a new concept. What are the consequences of the introduction of these patterns? What are the possibilities for the introduction of these patterns in Flowmanager, taking into account that each WfMS has their own specific configuration and language? How should these new patterns be introduced? What are the possibilities of introducing these new patterns in existing workflows? These question are outside the scope of this thesis and are addressed in the paragraphs 6.2 (recommendations) and 6.4 (future research).

5.2.2. Semantic checking

As concluded in chapter four, a difference in need for additional semantic checks can be observed. Theory indicates that two situations require additional checks, where the participants of the Delphi panel indicate that four situations should be accommodated with additional semantic checks. In contrary to the theory by Ly et al. (2008), the participants state that instances falling under *ΔI subsumes ΔS* and *ΔS subsumes ΔI* are also in need of additional semantic checks.

Although the number of participants with expertise in the field of semantic checks is limited, this does not mean that their answers to the questions in the Delphi panel are not useful. In contrary, the knowledge of these experts provides useful insights into the practicality of the theoretical concepts by Ly et al. (2008). Another point which has to be taken into consideration, is that a forward and backward citation analysis on Web of Science did not find other researchers attempting to bring these principles into practice. Therefore, at the start of the Delphi panel, the gap between the theory and practice was unknown. This thesis has confirmed the existence of such a gap by illustrating the difference in opinion on the use of semantic checks in different situations. In paragraph 6.4 suggestions are presented on narrowing down the gap between theory and practice.

Even though the participants understand that theoretically the semantic checks for the ΔI subsumes ΔS and ΔS subsumes ΔI biases are basically superfluous, in practice they would still want these additional checks: "*Although this situation does not seem to need additional checks, you will always want to perform a semantic check when the template is changed and the instance receives additional checks."* Another participant: "*If there is a new version of a template that differs from the changes in the instance, the instances need to be checked whether they possess the correct activities.*"

This shows that not all the principles introduced by Ly et al. (2008) are applicable in this practical context of Flowmanager. An explanation for this observed difference is that the participants (and thus the users commissioning the workflows) want to deliver a workflow process to the end users of high quality with an absolute certainty that no semantic issues will arise. Since the ΔI subsumes ΔS and ΔS subsumes ΔI patterns are relatively complex and have a difference in variables (in contrary to unbiased instances and equivalent bias instances), participants can vouch for additional checks for these instance types under the guise of 'better safe than sorry'.

Taking into account the opinions of the participants on the topic of semantic checking, we can conclude that for this situation it is justifiable to include additional checks for workflow process which are placed in the following situations:

- ΔI subsumes ΔS,
- ΔS subsumes ΔI,
- Partially equivalent bias and
- Disjoint bias.

The gap between theory and practice is also illustrated in the way that workflow processes are presented. Whereas Ly et al. (2008) differentiate between workflow templates and instances, Flowmanager does not contain instances. Every workflow pattern in Flowmanager follows the original

sequence of activities without exceptions. This difference in interpretation of workflow commissioning has consequences for the application of semantic checks in Flowmanager. Since Flowmanager does not work with workflow templates and instances, the principles by Ly et al. (2008) are not directly applicable to Flowmanager. However, in their paper, Ly et al. (2008) assume that the changes at the template level are semantically correct. This assumption does not apply to the situation of Flowmanager. At this moment, there is no semantic checking at all, so a change in the workflow template of Flowmanager is in need of semantic checks.

A principle by Ly et al. (2008) that provides possibilities for Flowmanager, is the use of constraints in determining the semantic checks. Constraints are used to determine which activities in a workflow should receive semantic checks. The constraints concerning workflow processes in Flowmanager are presented in business rules. So, for future research and future implementation of semantic checks in Flowmanager, the business rules provide guidelines into the correct placement of these checks.

Also, a logical observation can be made if we look at the selected four biases by the participants. What stands out, is that all the biases which were selected for additional checking, focus on the addition of one or more activities which are new to the workflow. After that, the constraints of the workflow tell us which activities require semantic checks. So, for Flowmanager we can conclude that should a new activity be introduced to the workflow, it should always be accompanied by semantic checks indicated by the workflow's business rules.

Conclusively, due to the differences in language between Flowmanager and the theory and the current state of semantic checks in Flowmanager, it is difficult to provide clear recommendations for semantic checks in the current state of Flowmanager, which are within the scope of this thesis. For future activities concerning the research and development of semantic checks in Flowmanager, constraints provide guidelines for the execution of semantic checks. Further research on the practical implementation of the theory by Ly et al. (2008) needs to be executed in order to properly assess the added value of the theory to practice.

Before a concrete proposition can be made on the implementation of semantic checks in Flowmanager, more research is required. This research requires a more in-depth technical approach, one which falls out of the scope of this Business Administration thesis. This future research should address questions such as: How can the concepts on semantic checking from the theory be translated so that it can be implemented in Flowmanager? Is there added value to the introduction of templates and instances in the workflow processes of Flowmanager? How can semantic checks be introduced in existing versions of Flowmanager? Based on these questions, recommendations for future research by BuildIT are made in paragraph 6.2.

5.3. Combined implementation

Throughout this thesis, the subjects of workflow commissioning and semantic checking have been discussed separately. In the previous two paragraphs, we have concluded that for workflow commissioning certain workflow patterns should be introduced in order to improve Flowmanager, and that for semantic checking no direct recommendations can be made for the current state of Flowmanager. However, the implementation of the suggested patterns has consequences for the functioning of Flowmanager. This is where the semantic checking principles as described by Ly et al. (2008) come in to play. This paragraph elaborates further on this topic.

The previous chapters have discussed five workflow patterns which should be implemented into Flowmanager: Multi choice, Synchronizing merge, Parallel split, Synchronization and Multi merge. These patterns provide a certain flexibility which is new to Flowmanager: activities in the workflow

process can be executed simultaneously. The two patterns which have received the highest prioritization are Multi choice and Synchronizing merge. The difference between these two patterns and the other three patterns is that Multi choice and Synchronizing merge do not oblige the process to execute all the activities which could be executed parallel. This property leads to instances of the workflow process which differ from each other: the mortgage application of person A needs one credit check, whereas the mortgage application of person B needs two credit checks.

We can perceive several similarities to the template and instances levels which are part of the principles presented by Ly et al. (2008). The template is the basic process and the instances may differ from the original template situation. Instances provide the flexibility which is required for these two patterns. This type of workflow commissioning is new to Flowmanager and its introduction will obviously lead to issues which have not occurred before in Flowmanager. These issues can be caused by changes in the workflow, which can be induced by changes in regulations or a change in the products offered by the owner of the process. These similarities indicate that the methods by Ly et al. (2008) are a useful addition for the introduction of Multi choice and Synchronizing merge.

When a process is commissioned in the form of a template accompanied by instances, a change in the template process could have consequences for the instances which differ from the template. These instances need to be checked if they are semantically correct before the actual changes are implemented. Ly et al. (2008) argue that only two of the six possible instance types (partially equivalent bias and disjoint bias) need to be checked. However, the participants of the Delphi panel have indicated that on top of these two instance types, instances falling under the ΔI subsumes ΔS bias or the ΔI subsumes ΔS bias should also receive a semantic check before proceeding.

In practice, this procedure could result in disorder. It is very well possible that hundreds of instances follow the structure of a single template. Checking all these individual instances (either by hand or automatically) is very time consuming. In order to prevent this situation, it makes sense to label each type of instance when the instance is initiated. Ideally this labeling should be done automatically (Flowmanager checks to what extent the instance differs from the template). It is also an option to perform this task by hand, if this procedure proves to be too complex or too time consuming to implement in Flowmanager. However, such a manual procedure leaves room for human error and is more time consuming in its execution, so an automated system is preferred.

For this paragraph, we can conclude that an introduction of the Multi choice and Synchronizing merge patterns requires a number of modifications in Flowmanager. The inclusion of these patterns should be accompanied by the introduction of the principle of workflow templates and instances, since it is a possibility that instances differ from each other. In order to maintain control of the semantic correctness of these instances, guidelines are provided by the principles of Ly et al. (2008) combined with the answers of the participants of the Delphi panel.

6. Conclusion

This chapter provides a conclusion for this thesis by elaborating on four different subjects. First, the research question is answered by describing the several stages of this thesis and the final outcome. Second, the recommendations to BuildIT which can be derived from this thesis are presented. Third, the limitations of this thesis are discussed. Fourth and final, based on the results of this thesis and its limitations, propositions are made for future academic research.

6.1. Answering the research question

The goal of this thesis has been to advise BuildIT on the improvement of workflow commissioning and semantic checking of Flowmanager, based on input from academic literature and the clients who use Flowmanager. Flowmanager is a Workflow Management System (WfMS) developed by BuildIT for the financial services industry. By enabling Flowmanager, these institutions are able improve the automation of their financial processes by commissioning workflows in Flowmanager.

For the gathering of relevant academic knowledge a literature review has been performed on the topics of business rules, workflow commissioning and semantic checking by applying the Grounded Theory Literature Review Method (Wolfswinkel et al., 2013). This resulted in two literature frameworks: the first framework focuses on workflow commissioning and is based primarily on the paper by van der Aalst et al. (2003). The second framework discusses semantic checks and is founded on the article by Ly et al. (2008).

Based on these frameworks, a Delphi panel has been created and executed with the goal of gathering input on the usability of the theoretical context for Flowmanager. The Delphi panel consisted of multiple rounds divided over three phases, based on the methodology by Okoli and Pawlowski (2004). In the first round, for workflow commissioning a selection was made consisting of patterns by van der Aalst et al. (2003) which are currently not available in Flowmanager.

In the second phase both the topics of workflow commissioning and semantic checking were discussed. For workflow commissioning, the participants assessed the possible contribution that these patterns could have to Flowmanager, leading to the selection of five workflow patterns for further analysis: Multi choice, Synchronizing merge, Parallel split, Synchronization and Multi merge. For semantic checking, participants answered question concerning the relevance of semantic checks in six situations. In their answering, they indicated that four situations were in need of semantic checks: ΔI subsumes ΔS, ΔS subsumes ΔI, Partially equivalent bias and Disjoin bias.

In the third phase, participants made rankings for both of the selections from the previous phase. For workflow commissioning this has led to a clear prioritization of the selected patterns, which is also displayed in figure 34:

1) Multi choice,
2) Synchronizing merge,
3) Parallel split,
4) Synchronization and
5) Multi merge.

For semantic checking, the answers of the participants in the third round of the Delphi panel were not reliable enough to draw conclusions similar to the results of the workflow commissioning part of this thesis. No distinctions could be made in the average ranking of the four biases.

A comparison of the theoretical concepts with the results of the Delphi panel has led to the following answer to the research question.

For *workflow commissioning*, the following prioritization for the implementation of new patterns should be used:

1) A combination of Multi choice and Synchronizing merge.
2) A combination of Parallel split and Synchronization.
3) Multi merge.

Two sets of two patterns should be introduced together, based on their relevance to each other, their ranking in the prioritization as shown in figure 34 and their ranking in the MoSCoW analysis as shown in figure 32.

For *semantic checking*, we can conclude that due to the gap between theory and the specific practice of this thesis, and the technical aspects which remain underexposed in this thesis, no concrete stand-alone recommendations can be made for the direct implementation of semantic checking in Flowmanager. On this topic, this thesis provides direction for future research on semantic checking by concluding that the business rules of Flowmanager prove to be important factors in the further development of semantic checking.

Although it is not possible to provide stand-alone recommendations for semantic checking in Flowmanager, semantic checking does play an important role in the introduction of Multi choice and Synchronizing merge in Flowmanager. These two patterns introduce flexibility that is new to Flowmanager. Instances can differ in the paths they follow within the process, they are able to entail different activities of the process. These differences can have consequences for the semantic correctness of the instances which differ from the main process, should the main process be altered. In order to prevent these issues, the implementation of Multi choice and Synchronizing merge in Flowmanager should be accompanied by the introduction of the principles of workflow templates and instances by Ly et al. (2008). These principles focus on the application of semantic checking in situations similar to the ones of Multi choice and Synchronizing merge. Finally, the answers of the participants of the Delphi panel have shown us for which of these instances semantic checks are required and for which situations semantic checks are irrelevant.

6.2. Recommendations to BuildIT

This part of the thesis has been marked as confidential.

6.3. Limitations

This thesis is subject to a number of limitations which are described in this paragraph.

- First, the focus of this thesis has been to provide BuildIT with an advice on how to improve Flowmanager. The Delphi panel has been designed and its participants have been selected specifically to Flowmanager. Therefore, the end results of this thesis are only applicable to Flowmanager and do not contribute directly to the theory, although several suggestions for future research can be done based on this thesis.
- Second, although the results of the Delphi panel provide interesting and useful findings for the context of Flowmanager, no conclusions based on statistical significance can be drawn due to the limited number of the participants. However, this limitation was known beforehand and no results based on statistical analysis were expected.
- Third, due to the gap between the used literature and the properties of Flowmanager, the semantic checking suggestions by the literature are not directly applicable to the current situation of Flowmanager. Suggestions for future research on this topic are made in this chapter for both BuildIT and academic research.

- Finally, this thesis has been written in the context of a Business Administration master program. Although there certainly has been some deepening in the technical aspects of business rules, workflow commissioning and semantic checking, the absence of real in depth knowledge on these topics has limited the degree of direct and concrete implementation advice delivered to BuildIT.

6.4. Future academic research

In this thesis, theories derived from the literature on workflow commissioning and semantic checks have been applied in the analysis of Flowmanager. This paragraphs provides two recommendations for future research.

This thesis has shown that for both workflow commissioning and semantic checks it appears to be difficult to translate theoretical concepts to useful contributions for Flowmanager. Several issues have arisen in the interpretation and implementation of the theoretical concepts. An interesting research topic is to investigate whether this is a common problem for WfMS's similar to Flowmanager. Research needs to be done to find out whether there is literature (or even a literature review) available on the process of implementing these theoretical concepts into practice. An analysis of this literature on both the topics of workflow commissioning and semantic checking could lead to the development of a roadmap for the implementation of these two theoretical concepts in Flowmanager.

In this thesis it has been shown that Flowmanager is not familiar to working with templates and instances as illustrated by Ly et al. (2008). It can be an interesting research topic to see to what extent other WfMS's include these principles in their operations. By doing so, we can properly assess the practical relevance of the methodology by Ly et al. (2008). Also, we can possibly derive useful methods on implementing these principles from other WfMS's. These methods can be applied to the implementation of templates, instances and the associated semantic check logic in Flowmanager.

7. Bibliography

Babbie, E. (2007). Unobtrusive Research. In E. Babby (Ed.), *The practice of Social Research* (pp. 325-388). Belmont, USA: Thomson Wadsworth.

Charfi, A., & Mezini, M. (2004). *Hybrid web service composition: business processes meet business rules*. Paper presented at the Proceedings of the 2nd international conference on Service oriented computing, New York, NY, USA.

Corbin, J., & Strauss, A. (1990). Grounded Theory Research: Procedures, Canons and Evaluative Criteria. *Qualitative Sociology, 13*(1), 19.

Day, J., & Bobeva, M. (2005). A Generic Toolkit for the Successful Management of Delphi Studies. *The Electronic Journal of Business Research Methodology, 3*(2), 103-116.

Diamond, I. R., Grant, R. C., Feldman, B. M., Pencharz, P. B., Ling, S. C., Moore, A. M., & Wales, P. W. (2014). Defining consensus: A systematic review recommends methodologic criteria for reporting of Delphi studies. *Journal of Clinical Epidemiology, 67*(4), 401-409. doi: 10.1016/j.jclinepi.2013.12.002

Georgakopoulos, D., Hornick, M., & Sheth, A. (1995). AN OVERVIEW OF WORKFLOW MANAGEMENT - FROM PROCESS MODELING TO WORKFLOW AUTOMATION INFRASTRUCTURE. *Distributed and Parallel Databases, 3*(2), 119-153. doi: 10.1007/bf01277643

Group, T. B. R. (2000). Defining Business Rules ~What Are They Really? , 77.

Haddar, N. Z., Makni, L., & Ben Abdallah, H. (2014). Literature review of reuse in business process modeling. *Software and Systems Modeling, 13*(3), 975-989. doi: 10.1007/s10270-012-0286-4

Hatton, S., & Society, I. C. (2008). *Choosing the "Right" prioritisation method*. Los Alamitos: Ieee Computer Soc.

Iacob, M. E., & Jonkers, H. (2009). A model-driven perspective on the rule-based specification and analysis of service-based applications. *Enterprise Information Systems, 3*(3), 279-298. doi: 10.1080/17517570903042762

Lanz, A., Weber, B., & Reichert, M. (2014). Time patterns for process-aware information systems. *Requirements Engineering, 19*(2), 113-141. doi: 10.1007/s00766-012-0162-3

Ludascher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., . . . Zhao, Y. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation-Practice & Experience, 18*(10), 1039-1065. doi: 10.1002/cpe.994

Ly, L. T., Rinderle, S., & Dadam, P. (2008). Integration and verification of semantic constraints in adaptive process management systems. *Data & Knowledge Engineering, 64*(1), 3-23. doi: 10.1016/j.datak.2007.06.007

Okoli, C., & Pawlowski, S. D. (2004). The Delphi method as a research tool: an example, design considerations and applications. *Information & Management, 42*(1), 15-29. doi: 10.1016/j.im.2003.11.002

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for Information Systems Research. *Journal of Management Information Systems, 24*(3), 45-77. doi: 10.2753/mis0742-1222240302

Reichert, M., & Dadam, P. (1998). ADEPT(flex) - Supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems, 10*(2), 93-129. doi: 10.1023/a:1008604709862

Rinderle, S. (2004). *Schema Evolution in Process Management Systems.* (Ph.D. Thesis), University of Ulm.

Rinderle, S., Reichert, M., & Dadam, P. (2004a). Correctness criteria for dynamic changes in workflow systems - a survey. *Data & Knowledge Engineering, 50*(1), 9-34. doi: 10.1016/j.datak.2004.01.002

Rinderle, S., Reichert, M., & Dadam, P. (2004b). Disjoint and overlapping process changes: Challenges, solutions, applications. In R. Meersman, Z. Tari, W. VanderAalst, C. Bussler, A. Gal, V. Cahill, S. Vinoski, W. Vogels, T. Gatarci, & K. Sycara (Eds.), *On the Move to Meaningful Internet Systems 2004: Coopis, Doa, and Odbase, Pt 1, Proceedings* (Vol. 3290, pp. 101-120). Berlin: Springer-Verlag Berlin.

Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems, 33*(1), 64-95. doi: 10.1016/j.is.2007.07.001

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., & Edmond, D. (2005). Workflow resource patterns: Identification, representation and tool support. In O. Pastor & J. F. E. Chunha (Eds.), *Advanced Information Systems Engineering, Proceedings* (Vol. 3520, pp. 216-232).

Salimifard, K., & Wright, M. (2001). Petri net-based modelling of workflow systems: An overview. *European Journal of Operational Research, 134*(3), 664-676. doi: 10.1016/s0377-2217(00)00292-7

Schwarz, R. B., & Russo, M. C. (2004). How to Quickly Find ARticles in the Top IS Journals. *Communications of the ACM, 47*(2), 4.

Spilter. Over Ons. Retrieved 30-01, 2015, from http://www.spilter.nl/organisatie

Stohr, E. A., & Zhao, J. L. (2001). Workflow automation: Overview and research issues. *Information Systems Frontiers, 3*(3), 281-296. doi: 10.1023/a:1011457324641

Thomson, & Reuters. (2014, 2014-11-17). Web of Science [v.5.15] - All Databases Citation Report. Retrieved 11, 2014, from http://apps.webofknowledge.com/CitationReport.do?product=UA&search_mode=CitationReport&SID=X2ugQSKrWmDoA7fL73f&page=1&cr_pqid=3&viewType=summary

van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases, 14*(1), 5-51. doi: 10.1023/a:1022883727209

Verbeek, H. M. W., Basten, T., & van der Aalst, W. M. P. (2001). Diagnosing workflow processes using Woflan. *Computer Journal, 44*(4), 246-279. doi: 10.1093/comjnl/44.4.246

Viriyasitavat, W., Xu, L. D., & Martin, A. (2012). SWSpec: The Requirements Specification Language in Service Workflow Environments. *Ieee Transactions on Industrial Informatics, 8*(3), 631-638. doi: 10.1109/tii.2011.2182519

Wolfswinkel, J. F., Furtmueller, E., & Wilderom, C. P. M. (2013). Using grounded theory as a method for rigorously reviewing literature. *European Journal of Information Systems, 22*, 11.

Worrell, J. L., Di Gangi, P. M., & Bush, A. A. (2013). Exploring the use of the Delphi method in accounting information systems research. *International Journal of Accounting Information Systems, 14*(3), 193-208. doi: 10.1016/j.accinf.2012.03.003

Xu, L. D., Viriyasitavat, W., Ruchikachorn, P., & Martin, A. (2012). Using Propositional Logic for Requirements Verification of Service Workflow. *Ieee Transactions on Industrial Informatics, 8*(3), 639-646. doi: 10.1109/tii.2012.2187908

zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and business rules: A representational analysis. *Information Systems, 35*(4), 379-390. doi: 10.1016/j.is.2009.02.006

8. Appendices

This part of the thesis has been marked as confidential.