Proposition of an agile knowledge-based process model

V. Llamas*,. T. Coudert*. L. Geneste*.**
J.C. Romero-Bejarano. A. de Valroger****

**Laboratoire Génie de Production / ENIT – University of Toulouse, 65016 Tarbes Cedex, France.*
(e-mail: {vllamas, thierry.coudert, laurent.geneste}@enit.fr)
***Axsens-bte. 31100 Toulouse, France (e-mail: {juan.romero, aymeric.devalroger}@axsens.com)*

Abstract: Modern organizations generally define and use standardized models of their processes. They manage their activities using such standards. In these processes, the generalization and reuse of knowledge is facilitated by the standardization. But it is sometimes difficult to react to unexpected events due to over-constrained standards. Companies need to become agile to survive to continuous changes in their environments. There is a requirement of agility for the processes in order to ensure constant responsiveness and flexibility. This necessity of agility can be achieved through a knowledge-based system. Therefore, this article proposes a knowledge-based agile process model in which agility is driven by the reuse of knowledge and experiences. For this purpose, agility operators are defined as formalized pieces of knowledge. A model of an agile process in which these operators are used is presented. The basis of a methodology describing incremental versions of the model is also presented. This "versioning" allows to formalize experiences and to capitalize them for future reuse. Finally, an application of the method to the problem solving domain is presented. It is shown how the standard 9 Steps process becomes more agile by deploying the proposed methodology.

Keywords: Agility, Business Process Management, Knowledge Management, Experience Feedback.

1. INTRODUCTION

In recent decades, companies and organizations have evolved to survive in the turbulent and unstable global market place. In this context, the control and the adaptation of their processes have been efficient key drivers.

Process management and modelling techniques have evolved along with organizations by proposing more flexible solutions to overcome continuous changes. Thus, process management has progressed from Business Process Reengineering (BPR) to Business Process Management (BPM) in the last twenty years, Jeston and Johan (2014). In general terms, the BPM approach supports and controls business processes through their analysis, modelling, simulation, documentation and execution, aiming at increasing the benefits of the organization through the improvement of its effectiveness, Van der Aalst et al. (2003).

Despite the efforts done to continuously improve business processes in today's organizations, some lacks of *agility* arise, von Rosing et al. (2014). This article intends to describe an *agile approach* to improve business processes. In order to illustrate and to introduce this *agile approach*, three types of generic processes are shown in Fig. 1. On the left, a standard fixed enterprise process (structured), and on the right a totally non-structured process are represented. The third case, on the middle, corresponds to an *agile process*.

The left situation represents a *structured process*. A pre-defined sequence of activities is formalized, standardizing the process for its systematic use. This allows both to guide the

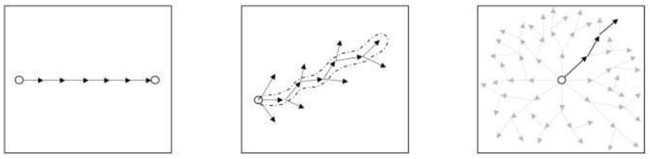


Fig. 1. Generic processes structuration

realization of the activities without ambiguities, and to formalize and reuse knowledge to help the decision makers all along the realization of the process. The scenarios are generally well known and formalized, consequently the knowledge that has to be reused is also well formalized (rules, constraints, formulas, models, standards, etc.). However, in such organizations, it is very difficult to change and to react to unexpected events or disturbances. Any difference from the standard process can be seen as a problem. The available and standardized knowledge can be obsolete when non formalized situations (i.e. new activities performed out of the standard) need to be overcome.

At the opposite, the situation on the right in Fig. 1 describes a *non-structured process*. In this case, several alternatives are available, providing a high level of flexibility that allows readjustments all along the process. Furthermore, when facing disturbances, this kind of process is able to be reconfigured and many changes can be brought to reach the objectives. New activities or new sub-processes can be added in real time in order to overcome the problems. Nevertheless, this high level of flexibility involves a low level of formalization. Such processes are widely open and it is quite difficult to define standards for their systematic reuse. Without a standardized process, the formalization and the reuse of knowledge is difficult to achieve because from one process execution to another one, the context and the process itself have both been changed. Each decision making is performed without any kind of help in a quite blind and reactive manner.

Therefore, both non-structured and structured processes present advantages and drawbacks. Thus, a third approach is proposed in this paper: an agile process. An agile process is a combination of both extreme situations, resulting in a flexible approach based on the continuous reuse of knowledge. The main drawbacks described previously are taken into account in order to define an agile process that:

- is sufficiently structured in order to guarantee objectives satisfaction and process efficiency but not over constrained by standards,
- can be reconfigured and adapted to unexpected situations,
- is based on experience feedback principles (i.e. the process is driven by knowledge reuse and permits to learn new knowledge and experiences during its execution).

In order to apply agile thinking to business processes and to define in detail the requirements such a process has to fulfil, a state of the art on agility concepts, core values and application domains is presented in the next section. It leads to the definition of the problematic addressed in this paper. Then a proposition for the definition of an agile process is given in section 2.3. In section 3, some elements of an agile process structure and its modelling are presented. In section 4 an example of an agile process applied to problem solving is presented. Finally, the conclusion and the perspectives of this work are presented in the last section.

2. STATE OF THE ART

The concept of agility has been studied in different application domains: business agility, enterprise agility, agile organization, agile workforce, IT agility, agile manufacturing, agile supply chains and agile software development, Kettunen (2009).

However, the most discussed subjects of agility in the literature are agile software development methods and agile manufacturing. There is no general consensus on a definition of agility, Izza et al. (2008). The concepts of Agile Software Development, Agile Manufacturing and other domains of agility, followed by an analysis of their main characteristics and how they contribute to this research work are presented in the next sections.

2.1 Agility in software development

Agile software development methods emerged to provide improvements regarding traditional ones. Traditional methods are characterized by pre-defined processes, activities planned in advance and regular documentation, Tarhan (2014). In the late 1990s, several traditional methods failed because of their inflexibility to adapt themselves to a strong demand driven by constant technological evolution, Lindstrom and Jeffries (2004).

In 2001, some agile software developers worked together to share practices. Then, the Agile Alliance was created introducing formally agility through the Agile Manifesto. The Manifesto outlines values and principles common to all agile methods, Lindstrom and Jeffries (2004): individuals and interactions should be valued over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan, Fowler and Highsmith (2001). The general principles these methods introduce are the flexibility and adaptability face to changes in requirements through the project. It means that, thanks to agile practices, the developer can easily modify the code to respond to changes of the requirements without major losses for the project, McCauley (2001). Two of the most spread methods are Extreme Programing and SCRUM, Schwaber and Beedle (2002), Lindstrom and Jeffries (2004).

The second main domain where agility has been developed is manufacturing. The principles are presented in the following section.

2.2 Agility in manufacturing and other domains

Since 1980, manufacturing companies are facing unprecedented levels of globalization, socio-political changes and market instability. Several studies were led in order to provide clarification on the causes of this new conditions in business, Sharifi and Zhang (1999), including a study by a group of scholars of the Iacocca Institute of Lehigh University in 1991 that introduced for the first time the concept of Agile Manufacturing.

There is no unified definition of Agile Manufacturing nor of its core concepts. Sharifi and Zhang (1999) define the agility as "*the ability to cope with unexpected changes, to survive unprecedented threats of business environment, and to take advantage of changes as opportunities*". In Yusuf et al. (1999), the authors define the agility as "*the successful exploration of competitive bases (speed, flexibility, innovation, proactivity, quality and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environment*".

Moreover, the concept of agility has been used in other domains. Besides the two above mentioned domains, research was extended to Agile Enterprise and Agile Supply Chain areas. In the agile enterprise domain, Dove (1994) defines agile dimensions, agile attributes, agile enterprise elements and change domains to introduce a structure of enterprise agility. In agile supply chain domain, Christopher (2000) introduces agility as the key to increase responsiveness through a hybrid approach in the supply chain. Considering the above cited concepts and definitions of agility, a study of its features is proposed below.

2.3 Characteristics of agility

In order to define the drivers that make a process agile, the agile characteristics defined in the literature were taken into consideration, in particular from the above described domains of Agile Software Development Methods and Manufacturing to reuse and adapt them to Agile Processes. Below, some of these articles with their addressed drivers are cited.

In Qumer and Henderson-Sellers (2008), the authors propose a set of features derived from his definition of agility, they are Flexibility, Speed, Leanness, Learning and Responsiveness. Yusuf et al. (1999) describe the core concepts of agile manufacturing: Core competence management, Virtual enterprise, capability for reconfiguration and knowledge driven enterprise.

In Sharifi and Zhang (1999), the authors propose an approach to achieve agility in organizations defining agility capabilities, agility drivers and agility providers. Amongst the agile capabilities, they describe responsiveness, competency, flexibility and quickness.

Following this short panorama on agility in many domains, in the next section, our own definition of agility is proposed.

2.4 Our definition of agility

Based on the relevant characteristics described in the last sections, the following requirements for an agile process are proposed:

Capability for reconfiguration: Ability to easily and significantly change activities of an agile process to answer to new purposes, constraints or events,

Collaboration: Association with team members and with other enterprises or individuals in order to solve a problem or make a decision,

Concurrent Engineering: Integrated organization path for agile processes in which the activities overlap and all the departments collaborate from the beginning of the process,

Core Competences Management: Knowledge of the available set of skills, its continuous improvement and its affectation to the adequate work position,

Innovativeness: Continuous engagement to search and experiment new ideas,

Knowledge Driven Process: Ability to reuse knowledge and experiences through the process,

Proaction: Actions taken to predict and adapt to change before it occurs,

Responsiveness: Ability to identify changes (expected and unexpected), respond fast, reactively or proactively, and recover from them, Sharifi and Zhang (1999),

Robustness: Ability to tolerate all transitions caused by change without having to take corrective actions, Conboy and Fitzgerald (2004),

Short activities: Cutting or division of long tasks to increase flexibility.

Considering these requirements and the definitions of agility encountered in the literature, adapting them and incorporating key concepts, a definition of an agile process is proposed.

Definition: An agile process is composed of an indeterminate number of sub processes and activities, limited by constraints (budget, resources, time, stakeholders' expectations and regulations) aiming to the achievement of a global objective. It has the ability to define and modify efficiently, through the process, the sub processes and activities of which it is composed. It takes into account, dynamically, the conditions of the environment, last results and past experiences. It enables continuously knowledge formalisation and reuse.

This article focuses on the *knowledge intensive reuse principle* as a main driver for agility. An agile process has to be sufficiently guided by past experiences and knowledge to be efficient. In order to develop such an agile process, a model based on the definition of different operators of agility is necessary. The next section is dedicated to the description of this agile process model and how it can be driven by knowledge, learning and reusing well formalized elements of experiences and lessons learned.

3. PROPOSITION OF AN AGILE PROCESS MODEL

In this section, an agile process model with its structure and components is presented. First, a global view of the agile process will help to clarify its main aspects. Then, more details will be provided, mainly on agility operators and the agile process versioning.

3.1 Agile process structure

The proposed agile process is mainly composed of sub processes, activities and decision points predefined before its launch. During its execution, modifications can be inserted in real time when unexpected events occur. An illustration of an agile process is proposed in Fig. 2.

Different types of activities are represented. Those included in the predefined first version of the process are foreseen activities and foreseen alternative activities. On the other hand, the activities created and planned after an event occurs are represented in new versions of the process (newly planned activities). Every time a decision has to be made, the decision makers can be helped and guided by the available knowledge and by the previous experiences capitalized into knowledge and experience bases. It is important to notice that decision points are also activities which are able to modify the agile process structure.

Before the process is launched, planned decision points (i.e. nominal decision points) are planned all along the process. When a decision point is reached within the agile process, decision makers will evaluate what to do next by analysing information collected during the previous activities and / or new aspects to consider. At each decision point, inputs are: process global and partial objectives; external information (e.g. changes in the market that affect the process); internal information; and indicators such as Key Performance Indicators (KPI) related to the process. In order to improve the agility, the knowledge and experience bases have to be consulted to provide insights on the decision to make. Similar previous situations and the available knowledge permit to aid decision makers.

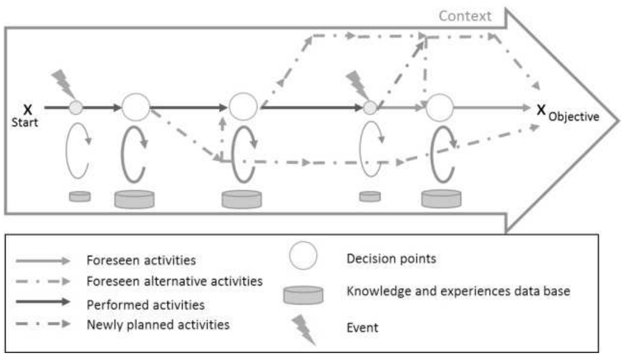


Fig. 2. Illustration of an agile process

Decision points can also be unplanned (event based decision points) when an unexpected event occurs, the decision makers will evaluate whether it is an impacting event that should be treated immediately, or it can be reported to the next planned decision point. During event-based decision points, activities or sub-processes can be modified to overcome the unexpected situation.

All along the agile process, but especially at decision makings (both nominal and event-based), the knowledge and experience bases will be consulted, searching for similar previous situations. This repository is accessible in both directions, not only to search past situations but also to store new situations and the way they were treated. The experience base will allow to carry out an experience feedback process which will help the future decision makings.

3.2 Agile process versioning

An agile process model is composed of incremental versions. The first version of the process model (V_0) includes planned activities, sub-processes and nominal decision making points. This initial version is built a *priori* taking into account objectives, context (budget, resources, time; stakeholders' expectations, and regulations), and experience feedbacks from previous processes executions. During the execution of the agile process, at each decision making point, the way forward is decided. Every time a decision involves a change to the previous version, a new version (V_{n+1}) is created.

Each new version will include a specific notation mentioning the result of decisions. This notation builds a trace of all the decisions made during the process additionally with the rational that led to that decision. This versioning facilitates the formalization of experiences, their capitalization and their future reuse.

In order to build an agile process, some agility operators have been identified. They permit to define the initial version of the agile process and to build new versions after each decision making or unexpected events. These operators are presented in the next section.

3.3 Agility operators

Every decision point is characterized by agile operators that will define next steps in the process as well as possible alternatives. There are two types of agile operators: Logical and Action. Logical operators are used to indicate that a choice needs to be done between multiple options, they are represented through logical gates (AND, OR, and XOR). An action operator describes an arrangement that needs to be done next.

The notations used to formalize the agile operators are represented in Table 1. Logical and action operators' possible options have been identified for an agile process and further explained in Table 2.

Table 1. Notation for agile processes

D_i^N	Nominal decision point i
D_j^E	Decision point j based on the occurrence of the event E
D_k	Any kind of decision point k
A_n	Activity n
SP_m	Sub-process m
SP_m*	Exploratory sub-process m
SA_p	Set of activities p
SSP_q	Set of sub-processes q

A decision D_k is specified as follow: D_k: OP{[options]}. OP corresponds to the agile operator (one or more). [options] corresponds to the different possibilities or choices that can be executed in the decision. The logical and action operators are represented in Tables 2 and 3.

Table 2. Logical operators

Multiple selection: AND {set of operators} The operation defines the multiple selection of a combination of action and/or logical operators
Exclusive selection: XOR{set of operators} The operator defines the exclusive choice between a combination of action and/or logical operators
Non-exclusive selection: OR{set of operators} The operator defines the non-exclusive choice between a combination of action and/or logical operators i.e. one or more possible choices

Table 3. Action operators

Sequencing: SEQ (set of objects) The operator allows to change the sequence of a set of objects (set of activities SA_p or a set of sub-processes SSP_q)

Insertion: INS (object) The operator generates the insertion of an object (activity A_n, sub-process SP_m or a decision D^N)
Paralleling: PAR (set of objects) The operator permits to achieve a set of activities SA_p or a set of sub-processes SSP_q in parallel
Activation: ACT (object) The operator permits to activate an activity A_n or a sub-process SP_m
Deactivation: DEA (object) The operator permits to deactivate an activity A_n or a sub-process SP_m
Fractionation: FRAC (A_n) The operator permits to fractionate the activity A_n
Definition: DEF{SP*} The operator defines the activities and decisions composing an exploratory sub process SP_m*

The use of these agility operators is illustrated on an example of problem solving process.

4. EXAMPLE OF AGILE PROCESS: APPLICATION TO PROBLEM SOLVING

In this section, an example of agile process is presented involving the main concepts introduced in the section 4. This example illustrates an agile problem solving process model. The problem solving method named "9 Steps" is used for this illustration. The same activities performed in 9 Steps have been used to build the agile problem solving process.

The predefined actions to perform (sequentially) in the 9 Steps method are:

A1: Start immediate containment

A2: Build the team

A3: Define problem

SP1*: Identify root causes

A4: Complete and optimize containment actions

A5: Define and select permanent corrective actions

A6: Implement permanent corrective actions and check effectiveness

A7: Standardize and transfer the knowledge

A8: Reward the team and close the process

To apply agile concepts into this example, the activity "Identify root causes" has been predefined as an exploratory sub process (SP*). In any problem solving situation, the identification of the root causes is an exploratory activity due to the high level of uncertainty.

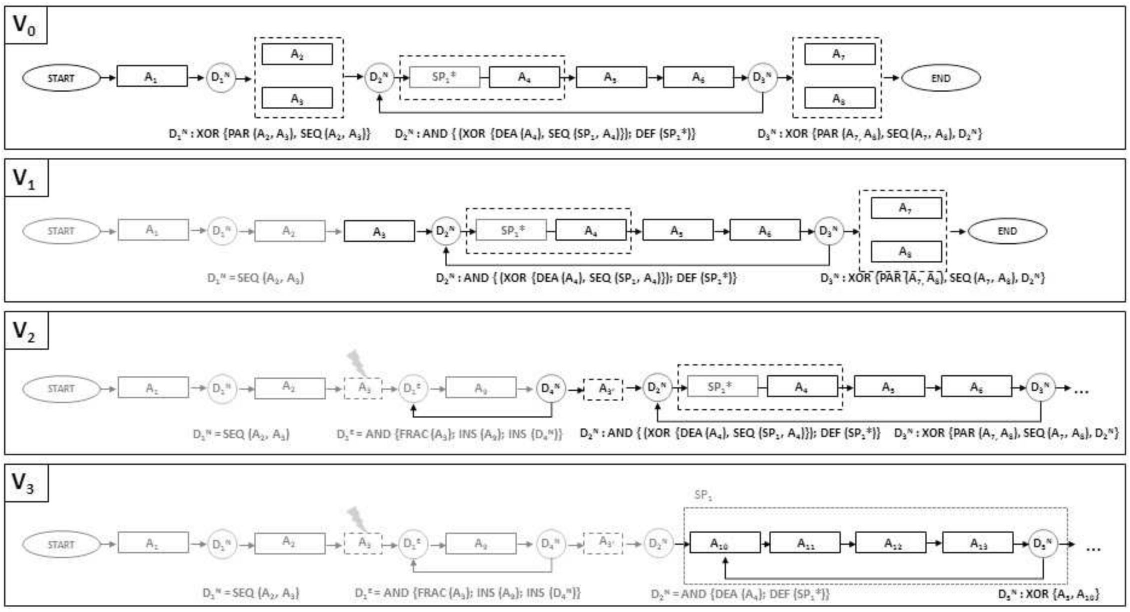
This example is illustrated in Fig. 3, it includes the first four versions of the process. The initial version (V_0) predefines the nominal decision points and the activities (according to the above cited list) of the process.

During run time, the first decision making point (D_1^N) is defined as follows:

D_1^N: XOR{PAR(A_2,A_3); SEQ(A_2, A_3)}

This notation means that a choice needs to be done between performing the activities A_2 and A_3 in parallel, or changing the order of the activities A_2 and A_3.

A new version needs to be defined to include the decision made in D_1^N, thus the second version of the process, V_1, is created. In this new version, the decision made in D_1^N is indicated as: D_1^N = SEQ (A_2, A_3), which means that the activities 'Build the team' and 'Define problem' will be performed in that order (A_2 and then A_3).



V_0
START
A_1
D_1^N
A_2
A_3
D_2^N
SP_1*
A_4
A_5
A_6
D_3^N
A_7
A_8
END
D_1^N : XOR {PAR (A_2, A_3), SEQ (A_2, A_3)}
D_2^N : AND { (XOR {DEA (A_4), SEQ (SP_1, A_4)}); DEF (SP_1*)}
D_3^N : XOR {PAR (A_7, A_8), SEQ (A_7, A_8), D_2^N}
V_1
D_1^N = SEQ (A_2, A_3)
D_2^N : AND { (XOR {DEA (A_4), SEQ (SP_1, A_4)}); DEF (SP_1*)}
D_3^N : XOR {PAR (A_7, A_8), SEQ (A_7, A_8), D_2^N}
V_2
D_1^E
A_9
D_4^N
$A_{3'}$
D_1^N = SEQ (A_2, A_3)
D_1^E = AND {FRAC (A_3); INS (A_9); INS (D_4^N)}
D_2^N : AND { (XOR {DEA (A_4), SEQ (SP_1, A_4)}); DEF (SP_1*)}
D_3^N : XOR {PAR (A_7, A_8), SEQ (A_7, A_8), D_2^N}
V_3
SP_1
A_{10}
A_{11}
A_{12}
A_{13}
D_5^N
D_1^N = SEQ (A_2, A_3)
D_1^E = AND {FRAC (A_3); INS (A_9); INS (D_4^N)}
D_2^N = AND {DEA (A_4); DEF (SP_1*)}
D_5^N : XOR {A_5, A_{10}}

Fig. 3. Example of an agile process: application to the problem solving domain

After the execution of A_1 and A_2, A_3 is performed but an unexpected event E occurs as shown in version V_2 of Fig. 3. The unexpected event is that the immediate containment action (A_1) was not conclusive and it needs to be completed immediately to avoid further damages. At that point, an event-based decision D_1^E is instantiated to define what to do next. The decision is: i) to fractionate the activity A_3 to treat this event and finish it later, ii) to insert a new activity A_9 (Complete containment actions), and iii) to add a new nominal decision point at the end of A_9 to ensure it has been correctly performed.

The last situation illustrated in Fig. 3 refers to the treatment of the above mentioned exploratory sub-process SP_1* (Identify root causes). When root causes need to be identified, a set of actions are defined based on this specific problem, its nature and the course of the problem solving process until that moment.

Thus, the version V_3 illustrates the decision made in D_2^N: i) to deactivate the activity A_4 (because the containment actions were already completed and optimized with the activity A_9); AND ii) to define the sub process SP_1* by four activities and one decision point to ensure the efficacy of the actions. The activities are: A_{10} (Collect data), A_{11} (Search for causes), A_{12} (Class and prioritize causes) and A_{13} (Identify and validate root causes).

5. CONCLUSION

In this article, an introduction to the general concept of agility had been proposed and applied to business processes in order to express the problematic: to define efficient knowledge-based agile processes for industry.

From the definitions found in the literature in different domains, our definition of an agile process and its requirements has been presented. First, a model for an agile process has been defined. Such a model is composed of a set of sub processes, activities and decision making points which are described though agility operators. Each decision which enables agility fosters the systematic and continuous reuse of knowledge and experiences. These operators, facilitating the formalisation of experiential knowledge, its reuse and integration into the process, constitute a key driver to make agile decisions. In order to formalize experiences and to permit their future reuse, a versioning mechanism has been proposed. Every time an agile decision which modifies the agile process is taken, a new version of the process is provided. Thus, it is possible to follow the evolutions, to formalize an experience and to capitalize it for a later reuse.

More work needs to be done on the agility operators discussed in this paper. First, agile operators need to be improved to ensure continuous agility all along the process. Second, based on the versioning method, an experience feedback process has to be defined in order to be able to identify similar previous experiences and to reuse them efficiently. In order to perform that, each agile decision which leads to a new version of the process has to be characterized (context, problem, explanations of the decision) in order to be reused. Then, ad hoc models have to be defined in order to capitalize and reuse such experiences, which could be facilitated through the use of Case Based Reasoning.

REFERENCES

Christopher, M. (2000). The agile supply chain: competing in volatile markets. Industrial Marketing Management 29, no. 1.

Conboy, K. and Fitzgerald, B. (2004). Toward a conceptual framework of agile methods. In Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, Newport Beach, CA, pp. 37-44

Dove, R. (1994). Tools for Analyzing and Constructing Agility. Agility forum. Bethlehem, PA.

Fowler, M. and Highsmith, J. (2001). The Agile Manifesto. Software Development.

Izza, S., Imache R., Vincent, L., and Lounis, Y. (2008). An approach for the evaluation of the agility in the context of enterprise interoperability. Edited by K. Mertins, R. Ruggaber, K. Popplewell, and X. F. Xu. New York: Springer.

Jeston, J. and Johan, N. (2014). Business Process Management. Routledge.

Kettunen, P. (2009). Adopting key lessons from agile manufacturing to agile software product development—a comparative study. *Technovation* 29, no. 6–7.

Lindstrom, L., and Jeffries, R. (2004). Extremme programing and agile software development methodologies. Information Systems Management 21:3, 41-52.

McCauley, R. (2001). Agile development methods poised to upset status quo." SIGCSE Bulletin 33 14–15.

Qumer, A., and Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Information and Software Technology 50, no. 4, 280–95.

Schwaber, K. and Beedle, M. (2002). Agile software development with Scrum. Prentice Hall.

Sharifi, H, and Zhang, Z. (1999). A methodology for achieving agility in manufacturing organisations: an introduction. International Journal of Production Economics 62, no. 1–2, 7–22.

Tarhan, A. and Seda Gunes, Y. (2014). Systematic analyses and comparison of development performance and product quality of incremental process and agile process." Information and Software Technology, Performance in Software Development, 56, no. 5, 477–94.

van der Aalst, W. M. P., ter Hofstede, A. H. M., and Weske, M. (2003). Business process management: a survey. In Business Process Management, edited by Wil M. P. van der Aalst and Mathias Weske, 1–12. Lecture Notes in Computer Science 2678. Springer Berlin Heidelberg.

von Rosing, M., von Scheel J., and Qumer Gill, A. (2014). Applying agile principles to BPM. In von Rosing, M., Scheer, A., and von Scheel, H., *The complete business process handbook,* 553 - 577. Morgan Kaufmann - El Sevier, Waltham, USA.

Yusuf, Y. Y., Sarhadi, M. and Gunasekaran, A. (1999). Agile manufacturing: the drivers, concepts and attributes. International Journal of Production Economics 62, no. 1–2, 33–43.