Accepted for the 5th International Conference on Enterprise Information Systems (ICEIS 2003),
Angers, France, 23–26 April 2003

SYSTEMS DEVELOPMENT METHOD RATIONALE
A Conceptual Framework for Analysis

Pär J. Ågerfalk, Kai Wistrand
Dept. of Informatics (ESI), Örebro University, SE-701 82 Örebro, Sweden
Email: pak@esi.oru.se, kwd@esi.oru.se

Keywords: Method rationale, method goals, method values, method fragments, standard methods, Method Engineering, method integration, Method Configuration

Abstract: The rationale of their creators inherently influences information systems development methods. Rationale, which is always based on the creator's values and assumptions about the problem domain, motivates, implicit or explicit, the different modelling activities and primitives prescribed by a method. The method, and hence its inherited rationale, directs method users' attention toward certain kinds of phenomena and away from others. Today we see a trend towards standardizing systems development in terms of standard modelling languages and standard development processes. When using an existing (standard) method, developers are forced to rely on the rationale of that particular method. Sometimes, however, there are reasons to enhance the standard method to reflect aspects of the world held as important by the method users – but not emphasized by the method creator. Hence, there is a need to integrate the rationale of the method users with that of the existing method. In this paper, we investigate what method rationale is and how it can be modelled and analysed. The paper employs a method engineering approach in that it proposes method support for analysing, describing and integrating method rationale – an emerging essential task for method engineers in a world of standardization.

1 INTRODUCTION

The rationale of their creators inherently influence information systems development (ISD) methods (Cronholm & Ågerfalk, 1999; Goldkuhl, Lind, & Seigerroth, 1998). Rationale, which is always based on the creator's values and assumptions about the problem domain, motivates, implicit or explicit, the different modelling activities and primitives prescribed by a method. The method, and consequently its inherited rationale direct method users' attention towards certain kinds of phenomena and away from others (Ågerfalk & Åhlgren, 1999). Today we see a trend towards standardization of system development in terms of standardized development processes and modelling languages, such as the Rational Unified Process (RUP) (Kruchten, 1999) and the Unified Modeling Language (UML) (Booch, Rumbaugh, & Jacobson, 1999). When using an existing method, developers are forced to rely on the rationale of the creator of that particular method.

Additionally, there is usually a need to configure standard methods to comply with the requirements imposed by various situated project-specific factors (Karlsson, Ågerfalk, & Hjalmarsson, 2001). In such cases there is a need to integrate the rationale of two or more methods.

The aim of this research is to understand what method rationale is and investigate how it can be modelled and analysed. The resulting framework for analysing method rationale can be used, for example, to facilitate the task of integrating the rationale of different methods, to verify method congruence, and to match methods with specific project requirements (cf. Ågerfalk & Åhlgren, 1999). Communicating methods through method rationale can assist method users in making informed decisions regarding the process of method configuration and also explicate otherwise tacit knowledge concerning organisation specific method configurations (Rossi, Tolvanen, Ramesh, Lyytinen, & Kaipala, 2000; Stolterman, 1992). These issues are addressed and elaborated in a project in co-operation with two industrial partners where the main focus is configuration and tailoring of commercially available ISD methods (see Ågerfalk *et al.*, 2003). The configured processes will be deployed and evaluated in the two existing software development organizations. The project employs a method engineering approach in that it proposes method support for analysing, describing and integrating the rationale of different methods – an emerging essential task for method engineers in a world of standardization.

The organization of the paper is as follows. First, we present and discuss the concepts of method rationale as applied within systems development. Then we present a framework for modelling and analysing method rationale based on the introduced concepts along with two examples of how the framework can be used.

2 THE CONCEPT OF METHOD RATIONALE

Basically, a method is a way of doing things – a process. That is, a method is a proposed pattern of activities, with the purpose of achieving some effect or goal. A method can also be viewed as a normative prescription for actions. Furthermore, a method, and the communication of its results, always implies the use of some concepts that can be expressed in some notation (a modelling language). Hence, the concept of method can be understood as a combination of concepts, notations and a process (Goldkuhl *et al.*, 1998). Acknowledging these three aspects of methods, Brinkkemper *et al.* (1999) propose the concept of *method fragment* to speak of a systems development method or any coherent part thereof. Method fragments are classified according to the three dimensions: *perspective*, *abstraction level*, and *layer of granularity*.

The perspective dimension considers the *product* perspective and the *process* perspective. A product fragment is a deliverable, such as a document or a system module. A process fragment represents the activities to be carried out to produce the products.

The abstraction dimension consists of the *conceptual level* and the *technical level*. Method fragments on the conceptual level are descriptions of systems development methods or parts thereof. Technical method fragments are executable tools implementing methods (i.e., CASE-tools).

The granularity dimension concerns what layer of granularity (or aggregation) a particular method fragment resides on. There are five possible layers: *method*, *stage*, *model*, *diagram*, and *concept*. A complete method, for example OMT (Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991), resides on the method layer. A fragment at the stage layer addresses a segment of the development lifecycle, for example 'Detailed design' of OMT. A fragment on the model layer addresses a particular view of the system, such as a data model or user interface model. A fragment at the diagram layer concerns a specific view of a model fragment, for example, a Class Diagram showing parts of a Class Model. Fragments on the concept layer are the atomic building blocks of the method, its process and notation. An example would be a modelling primitive such as 'object'.

A method's activities together with sequence restrictions among them form the method's process. If a method has been externalized and stated explicitly in, for example, a user's guide, that statement consists of several different components. It might consist of prescriptions regarding activities with corresponding modelling concepts and their notation. It might consist of statements addressing proposed effects and qualities regarding the expected results which can be seen as arguments and thereby rationale for the method. (Ågerfalk & Åhlgren, 1999) Two essential parts of such arguments are the values and goals constituting the motivation for prescribed method fragments. To be successful, method users have to understand and internalize (Berger & Luckmann, 1967) the rationale of the method creator and thus rely on the creator's judgements and normative statements. Internalization is a function not only of the existing method but also of the user's pre-knowledge, experience, values, rationale, tacit knowledge, *et cetera*.

This *rationality dimension* of methods has so far been left out of the method fragment concept. In order to respect this dimension as well, we turn to Ågerfalk & Åhlgren (1999) who argue that the rationale of a method equals the relation between method activities and goals – an approach to method rationale that can be extended to also incorporate product fragments as well as the relation between goals and values.

Every method fragment is (or at least should be) included in a method for some reason. A product fragment is, for example, supposed to contribute to the overall development effort by realizing some of the method goals. For example, a Class Model is suggested by OMT to facilitate conceptual understanding of the system under construction. This way, any method fragment can be related to one or more method goals, even though these are not always well articulated in method descriptions. A goal can thus be understood as a verifiable state of the world achieved by means of following method prescriptions; defined in Oxford English Dictionary

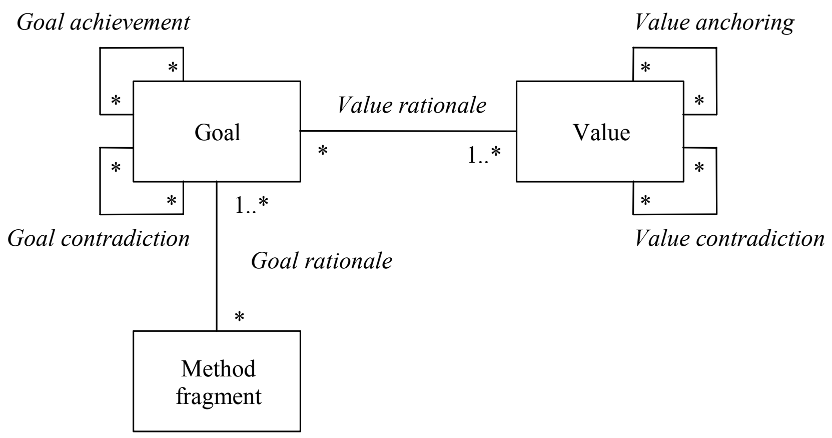

Figure 1: Method rationale as constituted by goals, values and their relationships.

as ‘an end or result towards which behaviour is consciously or unconsciously directed’.

Goals are manifestations of the method creator’s value base – all goals are anchored in values. Aligning with emotive theory (e.g., Ayer, 2001), a value can be understood as an ethical judgement that is not an assertion or report, but is an expression of feeling or attitude. That is, values are imperatives that cannot be judged as true or false.

To summarize this discussion, we suggest two different kinds of sub-rationale forming method rationale: (1) method prescriptions anchored in goals, referred to as *goal rationale*, and (2) goals anchored in values, consequently referred to as *value rationale*.

Furthermore, goals can be related to each other in goal hierarchies. For example, when a goal is viewed as a means to achieve another (higher) goal. Similarly, values can be anchored in other values. We refer to these two properties of rationale as *goal achievement* and *value anchoring*, respectively.

In addition to goal achievement, there is a possibility that goals contradict rather than complement each other – hence there is an additional *goal contradiction* relation defined over the set of goals. Similarly there is a *value contradiction* relation defined over the set of values.

Figure 1 depicts these concepts using the UML Class Diagram notation.

As depicted in Figure 1, every method fragment is related to at least one goal, and each goal is related to at least one value. This means that any method fragment is supposed to directly or indirectly contribute to the overall goal of using the method. It also means that every goal is a measurable counterpart to at least one value. Analytically, this implies that the reason for using a particular method fragment can be based on (a) its contribution to other higher-level goals, and (b) its realization of (parts of) the method’s underlying philosophy as expressed by identified values. Goals not belonging to either category (called *intrinsic goals*) are goals for their own sake and are questionable in a rational development process.

Conversely, expressed values not operationalized into goals and corresponding method fragments are symptoms of either inadequate method descriptions or unsubstantiated claims. In the latter case, further method development would probably be appropriate.

3 MODELLING RATIONALE

Using the concepts introduced in the previous section, method rationale can be modelled as follows.

Let F be the set of method fragments and G the set of goals, then the relation $R_G \subseteq F \times G$ is the goal rationale.

Similarly, let V be the set of values, then $R_V \subseteq G \times V$ is the value rationale.

Accordingly, $GAR \subseteq G \times G$ and $GCR \subseteq G \times G$ are the goal achievement relation and the goal contradiction relation, respectively.

Finally, $VAR \subseteq V \times V$ and $VCR \subseteq V \times V$ is the value anchoring relation and the value contradiction relations.

To illustrate the ideas, let us consider an example. The RUP is claimed to be use case driven in the meaning that use cases are the foundation for the entire development process (Kruchten, 1999). RUP also claims to be based on software developments best practises. These are believed to

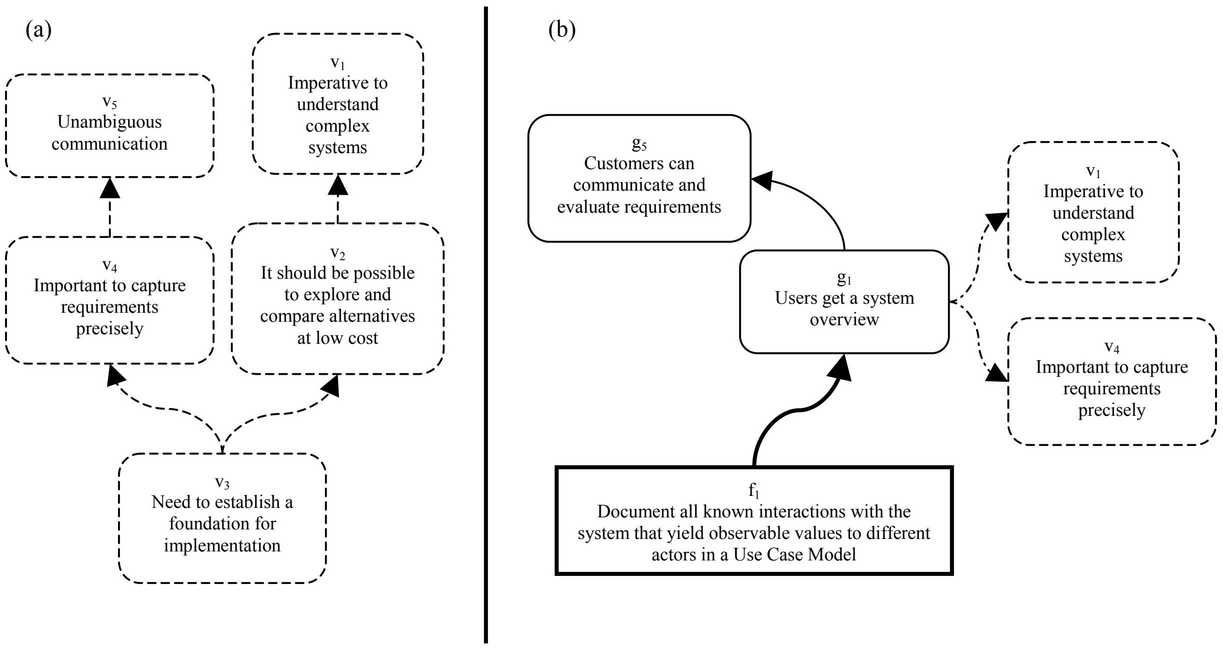
(a)
v_5 Unambiguous communication
v_1 Imperative to understand complex systems
v_4 Important to capture requirements precisely
v_2 It should be possible to explore and compare alternatives at low cost
v_3 Need to establish a foundation for implementation
(b)
g_5 Customers can communicate and evaluate requirements
g_1 Users get a system overview
v_1 Imperative to understand complex systems
v_4 Important to capture requirements precisely
f_1 Document all known interactions with the system that yield observable values to different actors in a Use Case Model

Figure 2: (a) Visual representation of values and value anchoring, and (b) view of the complete method rationale model focusing on one goal (g_1), traversing the model one relation away from g_1 (i.e., its adjacent nodes).

capture knowledge regarding how software development should be practised and accordingly constitute a value base for RUP. Therefore, the Use Case Model seems to be an appropriate starting point for analysis. For the purpose of this paper – to ascertain the viability of the approach – the following example is restricted to only encompass the values that can be found in one particular best practise, called Model Visually. As a consequence, some goals that can be related also to values found in other best practises are omitted. Likewise, only one method fragment is considered: a conceptual product fragment at the model layer of granularity – the Use Case Model.

Given these conditions it is possible to form the following tentative sets of values (V), goals (G) and method fragments (F):

V = {v_1:It is imperative to understand complex systems, v_2:It should be possible to explore and compare design alternatives at low cost, v_3:There is a need to establish a foundation for implementation, v_4:It is important to capture requirements precisely, v_5:The communication concerning decisions should be unambiguous (by using the UML)}.

G = {g_1: Potential users can get a system overview, g_2: Designers can get a system overview, g_3: People outside the project but within the organization, can get an insight into what has been done, g_4: Future developers of the next version of the system are able to understand how the existing version works, g_5: The customer can communicate whether the system fulfils the requirements or not, g_6: The software architect is able to identify architecturally significant functionality}.

F = {f_1: A Use Case Model documents all known interactions with a system that yield observable values to different actors}.

Now we can form the known goal rationale (R_G) and value rationale (R_V) as:

R_G = {(f_1, g_1), (f_1, g_2), (f_1, g_3), (f_1, g_4), (f_1, g_5), (f_1, g_6)} and R_V = {(g_1, v_1), (g_1, v_4), (g_2, v_1), (g_2, v_2), (g_2, v_3), (g_2, v_4), (g_3, v_5), (g_4, v_1), (g_4, v_2), (g_4, v_3), (g_4, v_4), (g_4, v_5), (g_5, v_4), (g_5, v_5), (g_6, v_1), (g_6, v_2), (g_6, v_3)}.

In this model the pair (f_1, g_1) of the goal rationale represents that: Documenting all known interactions with the system that yield observable values to different actors in a Use Case Model (f_1) is performed in order give potential users a system overview (g_1). Furthermore, the pair (g_1, v_1) of the value rationale represents that: Potential users should get a system overview (g_1) since it is imperative to understand complex systems (v_1). Similarly (g_1, v_4) reads: Potential users should get a system overview (g_1) since it is important to capture requirements precisely (v_4), *et cetera*.

In addition to rationale, we can form the goal achievement relation (GAR) and the value anchoring relation (VAR) as: GAR = {(g_1, g_5)} and VAR = {(v_2, v_1), (v_3, v_1), (v_4, v_5), (v_3, v_2), (v_3, v_4)}.

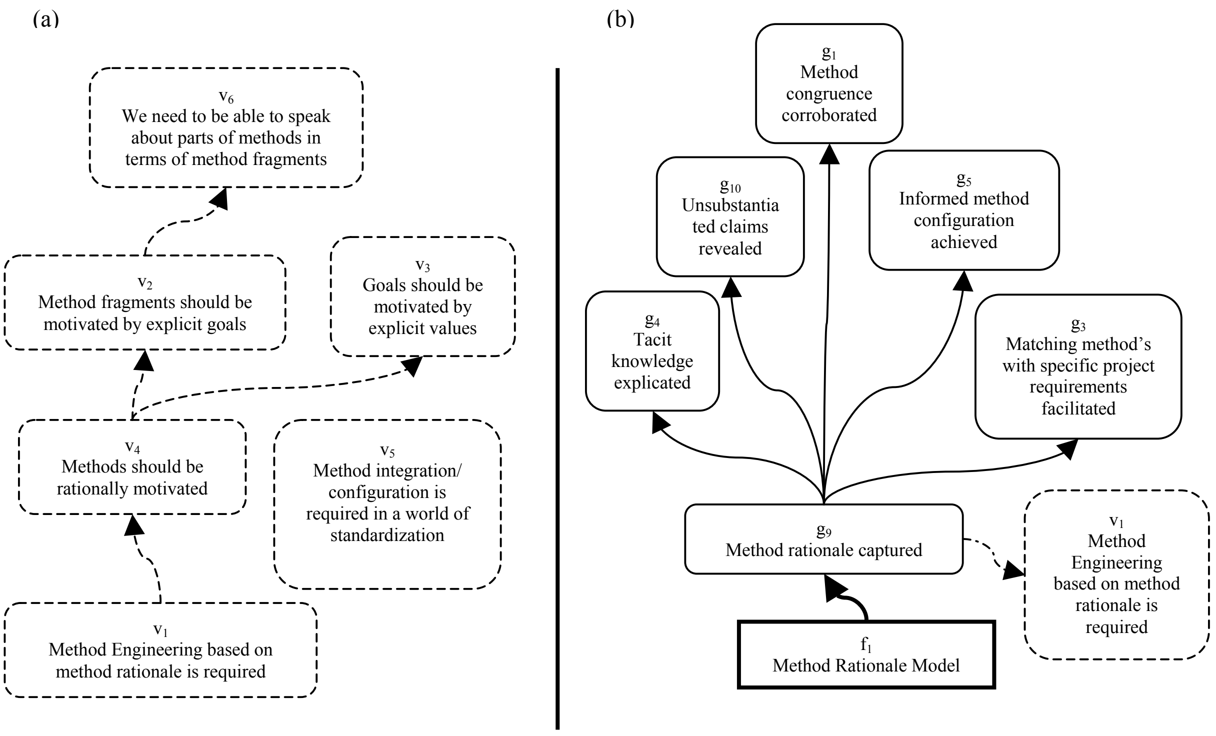
(a)
v_6 We need to be able to speak about parts of methods in terms of method fragments
v_2 Method fragments should be motivated by explicit goals
v_3 Goals should be motivated by explicit values
v_4 Methods should be rationally motivated
v_5 Method integration/ configuration is required in a world of standardization
v_1 Method Engineering based on method rationale is required
(b)
g_1 Method congruence corroborated
g_{10} Unsubstantia ted claims revealed
g_5 Informed method configuration achieved
g_4 Tacit knowledge explicated
g_3 Matching method's with specific project requirements facilitated
g_9 Method rationale captured
v_1 Method Engineering based on method rationale is required
f_1 Method Rationale Model

Figure 3: (a) Visual representation of values and value anchoring, and (b) view of the complete method rationale model focusing on one goal (g_9).

The pair (g_1, g_5) of the goal achievement relation should be understood as: When the potential users get a system overview, the customer can communicate whether the system fulfils the requirements or not. The pair (v_2, v_1) of the value anchoring relation reads: Since it should be possible to explore and compare design alternatives at low cost, it is imperative to understand complex systems. Another example, (v_4, v_5), states that: Since it is important to capture requirements precisely, the communication concerning decisions should be unambiguous (by using the UML).

Since we have not yet identified any goal contradictions or value conflicts we can simply form the goal contradiction relation (GCR) and the value conflict relation (VCR) as: $\mathrm{GCR} = \mathrm{VCR} = \varnothing$.

Figure 2 shows an example of a tentative visual representation of the model. Figure 2a shows V and VAR as a directed graph with the most prominent value (the base value) as a root. Of course, such a graph need not be rooted; it may in fact be disconnected (as is the case with G and GAR in the example). Figure 2b shows a view of the complete model focusing on one particular goal and its direct relations to other goals, values and method fragments. These kinds of graphs may be useful when constructing and validating the underlying model. Exactly what visual representations to use when modelling method rationale is one important issue to investigate further.

4 THE RATIONALE OF THE PROPOSED APPROACH

In order to ascertain whether we practice what we preach or not, this section is devoted to an analysis of the proposed modelling approach itself.

Based on the discussion so above, we can form the following model of the proposed approach (partly depicted in Figure 3):

V = {v_1: Method Engineering based on method rationale is required; v_2: Method fragments should be motivated by explicit goals; v_3: Goals should be motivated by explicit values; v_4: Methods should be rationally motivated; v_5: Method integration/ configuration is required in a world of standardization; v_6: We need to be able to speak about parts of methods (method fragments)}. G = {g_1: Method congruence corroborated; g_2: Method integration/configuration facilitated; g_3: Matching method's with specific project requirements facilitated; g_4: Tacit knowledge explicated; g_5:

Informed method configuration; g_6: Method fragments identified; g_7: Method goals identified; g_8: Method values identified; g_9: Method rationale captured; g_{10}: Unsubstantiated claims revealed; g_{11}: Need for method development identified}. $F = \{f_1$: Method Rationale Model}. $R_G = \{(f_1, g_1), (f_1, g_2), (f_1, g_3), (f_1, g_4), (f_1, g_5), (f_1, g_6), (f_1, g_7), (f_1, g_8), (f_1, g_9), (f_1, g_{10}), (f_1, g_{11})\}$. $R_V = \{(g_1, v_4), (g_2, v_1), (g_2, v_5), (g_3, v_5), (g_4, v_4), (g_4, v_1), (g_5, v_1), (g_5, v_1), (g_6, v_6), (g_7, v_2), (g_8, v_3), (g_9, v_1), (g_{10}, v_4), (g_{11}, v_4)\}$. $GAR = \{(g_7, g_9), (g_6, g_9), (g_8, g_9), (g_9, g_4), (g_9, g_{10}), (g_9, g_1), (g_9, g_5), (g_9, g_3), (g_{10}, g_{11}), (g_3, g_2)\}$. $VAR = \{(v_1, v_4), (v_4, v_2), (v_4, v_3), (v_2, v_6)\}$. $GCR = VCR = \varnothing$.

From this example, it is clear that the proposed approach can be used to reconstruct its own rationale, thus substantiating the claims above.

5 CONCLUSION

In this paper we have discussed the concept of method rationale in relation to information systems development. We have argued that method rationale is based on *values* and *goals*, which are related by *value rationale*. Furthermore, goals are related to method prescriptions by *goal rationale*. In addition, goal achievement relations describe hierarchies amongst goals. Similarly, values are related to each other by a value anchoring relation.

Using two examples, a small fraction of the RUP and the suggested approach itself, we have shown how to model these components to form a framework useful for analysing method rationale. The purpose of this work is to serve as a foundation for future research on method configuration – an emerging important task for method engineers. Derived models can easily be stored in a relational database, which also can be used for congruency checks in an integration situation.

A problem with the proposed framework is its rigorous approach. In most practical cases, it is probably impossible, or at least intractable to track all suggested primitives. Consequently, a future task is to develop a process to tailor rationale analysis frameworks at appropriate levels of granularity. However, such ‘simplified’ frameworks must still be anchored in the framework presented in this paper.

REFERENCES

Ågerfalk, P. J., & Åhlgren, K. (1999). Modelling the rationale of methods. In M. Khosrowpour (Ed.), *Managing Information Technology Resources in Organizations in the Next Millennium* (pp. 184–190). Hershey, PA: IDEA Group Publishing.

Ågerfalk, P. J., Wistrand, K., Karlsson, F., Börjesson, G., Elmberg, M., & Möller, K. (2003). Flexible Processes and Method Configuration: Outline of a Joint Industry-Academia Research Project, *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003)*.

Ayer, A. J. (2001). *Language, truth and logic* (New ed.). London: Penguin.

Berger, P. L., & Luckmann, T. (1967). *The social construction of reality: a treatise in the sociology of knowledge*. New York: Anchor books.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language user guide*. Harlow, UK: Addison-Wesley.

Brinkkemper, S., Saeki, M., & Harmsen, F. (1999). Meta-modelling based assembly techniques for situational method engineering. *Information Systems, 24*(3), 209–228.

Cronholm, S., & Ågerfalk, P. J. (1999). On the Concept of Method in Information Systems Development. In T. Käkölä (Ed.), *Proceedings of the 22nd Information Systems Research Seminar in Scandinavia (IRIS 22)* (Vol. 1, pp. 229–236).

Goldkuhl, G., Lind, M., & Seigerroth, U. (1998). Method integration: The need for a learning perspective. *IEE Proceedings - Software, 145*(4), 113–118.

Karlsson, F., Ågerfalk, P. J., & Hjalmarsson, A. (2001). Process Configuration with Development Tracks and Generic Project Types, *Proceedings of the 6th CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD’01)*.

Kruchten, P. (1999). *The rational unified process: an introduction*. Reading, MA: Addison-Wesley.

Rossi, M., Tolvanen, J.-P., Ramesh, B., Lyytinen, K., & Kaipala, J. (2000). Method Rationale in Method Engineering, *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33)*: IEEE Computer Society Press.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs: Prentice Hall.

Stolterman, E. (1992). How system designers think about design and methods: Some Reflections Based on an Interview Study. *Scandinavian Journal of Information Systems, 4*, 137–150.