Patterns in Business and Information Systems Engineering

DOI 10.1007/s12599-009-0073-0

The Authors

Prof. Dr. Robert Winter
Universität St. Gallen
Institut für Wirtschaftsinformatik
Müller-Friedberg-Strasse 8
9000 St. Gallen
Switzerland
robert.winter@unisg.ch

Jan vom Brocke,
Peter Fettke,
Peter Loos,
Stefan Junginger,
Christoph Moser,
Wolfgang Keller,
Florian Matthes,
Alexander Ernst

This article is also available in German in print and via http://www.wirtschaftsinformatik.de Winter R (2009) Patterns in der Wirtschaftsinformatik. WIRTSCHAFTSINFORMATIK. doi: 10.1007/11576-009-0195-5.

In software engineering, patterns are indispensable both from a research- and from a practice-based perspective. In Business and Information Systems Engineering (BISE), however, it is not entirely clear where and how to position patterns in the system of the established result artifact types of construct, (reference) model, method, and (information system) instance as well as in the design and (re-)use process of artifacts. Although patterns bear certain similarities to reference models, they seem to evolve in other ways and are also used differently.

This discussion deals with the question of why patterns have found comparatively little attention/application in BISE research in spite of being very successful in software engineering. Besides a general description of their experience, the contributors were asked to possibly comment on the following questions:

- Why are patterns successful in software engineering?
- Why has hardly anyone transferred the "pattern mechanism" to BISE so far?
- Is the "real" transfer of the "pattern mechanism" to BISE the key to an unused potential for the design of business models, processes, etc.?
- Or, can reference models be understood as the patterns of the business architecture? What then differentiates reference models from patterns?
- Should also patterns exist for the result creation process in addition to patterns describing the structure of the resulting artifact?
- Are there any successful examples of patterns in BISE?

As the perspectives of researchers and practitioners may well be different, I invited people representing these different views for contributions. The following authors accepted my invitation to this discussion (in alphabetical order):

- Prof. Dr. Jan vom Brocke, Hilti Chair of Business Process Management, Institute of Information Systems, University of Liechtenstein,
- PD Dr. Peter Fettke and Prof. Dr. Peter Loos, Institute of Information Systems at the German Research Center for Artificial Intelligence (DFKI), Saarbrücken,
- Dr. Stefan Junginger, member of the Board of Managers, and Christoph Moser, Product Manager, BOC AG, Vienna
- Wolfgang Keller, author and consultant, object architects, Gräfelfing,
- Prof. Dr. Florian Matthes and Alexander Ernst, Chair for Informatics 19 (sebis), Faculty for Informatics, Technical University of Munich.

Prof. Matthes and Mr. Ernst report on their own experience in the identification and description of patterns in BISE and provide an overview of the basic pattern characteristics.

Prof. vom Brocke recommends applying the experience from the pattern community or from reference modeling to a design theory which is interdisciplinary and designed for the reuse of artifacts.

Dr. Junginger and Mr. Moser as well as Mr. Keller analyze the properties of patterns and the cultural differences between the pattern community and BISE from the perspective of the pattern-using practice. They show in what aspects BISE might learn from the pattern community.

Dr. Fettke and Prof. Loos finally propose – based on an analysis of developments in the discussion of reference models as well as the pattern debate – to stop distinguishing between reference models and patterns from a terminological view. They propose five characteristics based on which patterns and reference models can be understood as different instances of a similar basic concept.

The contributions make clear that on the one hand significant similarities are seen between the artifact types pattern and reference model in software engineering and BISE. On the other hand, especially patterns cannot only be reduced to the respective artifacts. Instead, the success of patterns is marked by a vibrant ecosystem of the pattern community. Writer workshops, review culture, and lived reuse are key elements of this phenomenon, which can not or only in beginnings be found in terms of reference models.

If you would like to comment on this topic or another article of the journal Business & Information Systems Engineering (BISE), please send your contribution (max. 2 pages) to the editor-in-chief, Prof. Hans Ulrich Buhl, University of Augsburg, Hans-Ulrich.Buhl@wiwi.uni-augsburg.de.

Prof. Dr. Robert Winter
Institute of Information Management
University of St. Gallen

Capturing, Structuring, and Passing Knowledge for the Design of Complex Systems as Patterns

Over the past fifteen years, patterns in software engineering have evolved worldwide as a successful tool to capture knowledge for the design of complex systems, to

structure and distribute it (Buschmann et al. 2007). Patterns document solutions to recurring problems in a given context which have proved successful in practice. The solution is described by structures, relationships, interactions, and principles rather than by detailed algorithms or procedures.

So-called anti-patterns document bad solutions frequently occurring in practice for given design tasks. They arise, for example, due to a lack of experience, skill, and foresight or as a result of poor communication and teamwork.

Patterns everywhere?

Gradually, the scope of patterns has been expanded: design patterns (Gamma et al. 1995) assist in the design of individual object-oriented software components, architectural patterns (Buschmann et al. 1997) assist with the composition of software components to applications, architectural patterns for enterprise applications (Fowler 2002) finally capture design principles for networking applications to IT environments.

Recent work uses patterns of organization to describe and design social processes and structures in the field of IT. The knowledge captured as a pattern refers not only to the respective relevant artifacts (documents, diagrams, models, technical systems), but also to roles, social structures, activities, and processes.

Coplien and Harrison (2004), for example, focus on organizational and behavioral patterns in agile software development, Manns and Rising (2004) examine patterns conducive for change management. DeMarco et al. (2008) describe a number of constructive and destructive behavioral patterns in projects and teams and provide systematic instructions to avoid conflicts and increase team productivity (especially in IT projects).

Based on similar objectives, since 2007 Buckl et al. have developed a catalog of patterns with experienced partners for the management of enterprise architectures as a public wiki (sebis EAM Pattern Catalog, Ernst 2008). It does not include enterprise architecture patterns, but patterns which assist in creating a company-specific approach to managing the enterprise architecture based on the individual situation of a particular company.

The patterns catalog distinguishes 55 different possible design tasks (concerns) in the EAM and identifies best practice patterns for these tasks (patterns methodology). These describe roles and organizational structures (e. g. committees, project teams) and their responsibilities and interactions. They are clarified by reference to each of the chart and document types used by the respective players. These constitute target group specific views on the enterprise architecture and are collected and described in the catalog as viewpoint patterns. For each perspective the catalog documents the required architectural information (information model patterns) by reference, which are UML class diagram fragments similar to the classic design patterns.

An example of an EAM-anti-pattern which is documented in the catalog is the lack of focus on just few company-specific core design tasks in the establishment of EAM. As a result, an architectural model is designed which is too rich in detail. Furthermore, it is not possible to collect and to maintain the required pieces of architectural information with sufficient quality. Ultimately, such projects fail early because the expected benefits of modeling cannot be demonstrated due to the poor quality of the models.

Characteristics of pattern-based approaches

As a rough outline, the characteristics of pattern-based approaches can be summarized as follows:

Patterns are

- *constructive and compositional:* A pattern provides a defined contribution to resolving a manageable aspect of an overall problem. A system design typically uses several patterns.
- *situational:* Driving forces and basic conditions for the application of patterns are discussed, positive and negative consequences of the application of patterns are demonstrated.
- *formulated in natural language:* The name of the pattern is especially important which is supposed to enrich the vocabulary of users. Examples, illustrations, diagrams, cross-references, and citations are promoted. A formalization by meta-models and formal languages does not occur.
- *informal:* The selection of an appropriate pattern and the implementation of a pattern in the draft requires human intelligence and creativity; only in rare cases can it be attributed to a simple "customizing" through configuration, option selection, instantiation, and parameterization of a generic template.
- *practice-oriented:* The patterns are obtained through a careful study of the practice (pattern mining). The practical relevance should be documented with (three or more) concrete and verifiable application examples. Patterns should be formulated in an understandable way for novices and in an interesting way for experts, making them attractive for training. They should also be assessed by practitioners.
- *incrementally usable:* The application of a pattern is often described as optimizing the transformation of an existing system (draft). This allows an evolutionary approach in dynamic environments and in case of incomplete information (fix problem and observe effects).
- *networked:* Some patterns are only applicable in the context of other patterns; there are compound patterns and mutually exclusive patterns, etc. These relationships are thoroughly discussed in pattern books and pattern catalogs and explained with diagrams and cross-references.
- *conducive to discourse:* The mentioned pattern characteristics facilitate cooperation between pattern-author and pattern-users about artifacts of low complexity and high coherence. The typical pattern life cycle begins with the discovery, followed by documentation, peer review, publication, and public feedback, each including incremental improvement of the documentation. Over the years, the pattern community has developed very effective and constructive methods of cooperation. These include wikis, authoring workshops, author coaching (shepherding), and pattern-conferences with specific guidelines for review and the demand for publishing pattern descriptions in an openly accessible form.

One obvious application field for patterns is information management. Here, they are in competition with established BISE methods, in particular from reference modeling and business engineering, and thus exciting research questions evolve on what a cross-fertilization may look like, for example:

- Do reference model catalogs constitute an appropriate source for pattern mining? (see Fettke 2009)
- Can the reuse of design results intended by reference models also be achieved with pattern-based design methods?
- Which benefit may be achieved by models and meta-models in pattern-based design?
- What can initiatives for the open exchange and cross-company development of models (open models) in BISE learn from the pattern community?
- What corresponds to anti-patterns in information management?

Prof. Dr. Florian Matthes
Alexander Ernst
Chair for Informatics 19 (sebis)
Faculty for Informatics,
Technical University of Munich

References

Buckl S, Ernst A, Lankes J, Schneider K, Schweda C (2007) A pattern based approach for constructing enterprise architecture management information models. In: 8. Internationale Tagung Wirtschaftsinformatik, Karlsruhe

Buschmann F, Henney K, Schmidt D (2007) Pattern oriented software architecture. Vol. 5: On patterns and pattern languages. Wiley

Buschmann F, Meunier R, Rohnert H, Sommmerlad P (1997) A system of patterns: Pattern-oriented software architecture. Wiley

Coplien J, Harrison N (2004) Organizational patterns of agile software development. Prentice Hall PTR

DeMarco T, Hruschka P, Lister T, Robertson S, Robertson J, McMenamin S (2008) Adrenaline junkies and template zombies: Understanding patterns of project behavior. Dorset House

Ernst A (2008) Enterprise architecture management patterns. In: Pattern languages of programs conference (PLoP08), Nashville

Fettke P (2009) Reference model catalog. http://rmk.iwi.uni-sb.de/index.php. Accessed 2009-07-28

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns. Elements of reusable object-oriented software. Addison-Wesley

Manns M, Rising L (2004) Fearless change: Patterns for introducing new ideas. Addison-Wesley

Interdisciplinary Design Theory

As a contribution to this discussion, I would suggest that we let go of single term discussions and rather think towards an interdisciplinary design theory. Is it really the case that "patterns" have hardly been used in BISE? Aren't they perhaps just called differently? Do also specific requirements for designs in BISE exist, requiring different approaches? An interdisciplinary design theory would offer the opportunity to better understand the process of constructing artifacts and to let the results benefit a variety of disciplines. In the following we would like to specify this further.

First, it makes sense to go back to the intent of patterns in order to detach the discussion from single aspects of this approach in software engineering: it is well-known that patterns originate from town and country planning (Alexander et al. 1977). The intention was to describe proven solutions in such a way that they can be reused in multiple contexts. Already here templates for describing patterns were established, such as the designation of the name, the problem, the context, and the solution (Alexander 1979, p. 247). This approach has been applied to software engineering and specified using various types of patterns (Coad 1992; Gamma et al. 1996; Fowler 1996).

In BISE, the transfer of the pattern idea cannot be observed in the same way. However, can we conclude that there are no patterns in this field? Consider, for example, models such as ITIL (IT Infrastructure Library) or SCORM (Supply Chain Operation Reference-Model). These models are explicitly developed with the intention of reusing proven solutions and certainly find worldwide attention. Also, many "smaller" models can be mentioned here, such as accounts and documentary structures (Scheer 1994) which definitely act as "patterns" in the sense of reuse – although they were not explicitly designed as such. Even questions of the construction of "patterns" have a long tradition in BISE, as research on reference modeling shows (Becker et al. 2007). There, specific methods are developed to particularly reuse information models (specifically: process and data models) in different contexts (vom Brocke 2007).

But not only models, as the result of constructions, are reused in BISE. Also knowledge about the process of design is the subject of consideration. Here, the field of method engineering is to be mentioned (Brinkkemper 1996). Especially in situational method engineering issues of reusing core method elements are examined as well as their adaptation in different application contexts (Bucher and Winter 2009). In the same way as reference models, also "reference methods" could be regarded as patterns in which – in this case – rules to achieve specific goals are examined and described in terms of activities, outcomes, and roles.

All in all, it becomes clear how close the different approaches are. The core aspect is the reuse of artifacts in the design process. Ultimately, it could be concluded that almost every artifact that is designed as part of a design-oriented research process (Winter 2008) could well be understood as a pattern. After all, we are dealing with generic artifacts which are explicitly intended to be reused beyond just the particular case and thus in different contexts. Moreover, this would expand the discussion to terminology design which deals with very similar issues, as is shown by the extensive research on ontologies, for example.

Of course it remains to discuss whether or not we should speak of "patterns" in all of these cases. However, it appears of greater importance to better understand the underlying phenomenon of reuse in order to exploit synergies between the contributions of different disciplines. At least three design aspects would have to be considered:

- *Methodical aspects*: Which rules should reuse follow? While patterns can typically be applied to a specific case by analogy (Fowler 1996, p. 8), in reference modeling the focus had long been on the configuration of models (Becker et al. 2007). Meanwhile, there is a comprehensive set of modular methods which also provides for instantiation, aggregation, and specialization (vom Brocke 2007). In addition, there are questions concerning the specification of the situational context (Bucher and Winter 2009). What attributes, but also what kind of vocabulary should be used to describe the scope of an artifact? This is a central problem which can also be observed for the reuse of components and services.
- *Organizational aspects*: What are the institutional exchange relationships of the actors involved? Here, e. g. transaction cost theory (Coase 1937) provides evidence. Often, a market-based, or even open source based, exchange of artifacts is assumed. The majority of patterns in BISE, however, are used within the company, e. g. in the form of so-called blueprints of larger com-

panies or consulting firms. The reason may be that a relatively high "specificity" or "strategic importance" is attributed to these artifacts (vom Brocke and Buddendick 2004). Many initiatives fail because such basic organizational aspects of reuse are ignored.

- *Technical aspects*: What tools can support the process of reuse? Still, construction techniques for the reuse of most of the modeling and case tools are supported only rudimentarily. A more comprehensive tool support would be necessary, preferably in a vertically integrated way at all levels of the development process in order to make reuse practicable. In addition, collaboration features are necessary to enable an evolutionary development of artifacts (vom Brocke 2004). Again, a multitude of general questions arise.

The examples show what issues are generally connected with the reuse of artifacts in software engineering and BISE. The aim of an interdisciplinary design theory is to cease analyzing these for single cases (and again for different concepts), but instead to examine them for design processes in general.

Besides the discussion on the aspect of reuse as an example there are other important issues, such as the sustainability of design results. It therefore seems promising to discuss these issues in close cooperation with other disciplines. Software Engineering and BISE could make a start regarding this objective. Other disciplines, such as architecture, engineering sciences, or also visual arts can surely provide additional exciting ideas.

Prof. Dr. Jan vom Brocke
Hilti Chair for Business
Process Management
Institute of Information Systems
University of Liechtenstein

References

Alexander C (1979) The timeless way of building. Oxford University Press, New York

Alexander C, Ishikawa S, Silverstein M (1977) A pattern language: Towns, buildings, construction. Oxford University Press, New York

Becker J, Delfmann P, Knackstedt R (2007) Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models. In: Becker J, Delfmann P (eds) Reference modeling. Efficient information systems design through reuse of information models. Physica, Heidelberg, pp 23–49

Brinkkemper S (1996) Method engineering: Engineering of information systems development methods and tools. Information and Software Technology 38(4):275–280

Bucher T, Winter R (2009) A taxonomy of business process management approaches. In: vom Brocke J, Rosemann M (eds) Handbook on business process management. Springer, Heidelberg (In print)

Coad P (1992) Object oriented patterns. Communications of the ACM 35(9):125–159

Coase RH (1937) The nature of the firm. Economica 4(11):386–405

Fowler M (1996) Analysis patterns: Reusable object models. Addison-Wesley Professional

Gamma E, Helm R, Johnson R (1995) Design patterns. Elements of reusable object-oriented software. Addison-Wesley Longman, Amsterdam

Scheer AW (1994) Business process engineering: Reference models for industrial enterprises. Springer, New York

vom Brocke J (2004) Internetgestützte Referenzmodellierung. State-of-the-Art und Entwicklungsperspektiven. WIRTSCHAFTSINFORMATIK 46(5):390–404

vom Brocke J (2007) Design principles for reference modelling. Reusing information models by means of aggregation, specialisation, instantiation, and analogy. In: Fettke P, Loos P (eds) Reference modelling for business systems analysis. Idea Group Publishing, Hershey, pp 47–75

vom Brocke J, Buddendick C (2004) Organisationsformen in der Referenzmodellierung, Forschungsbedarf und Gestaltungsempfehlungen auf Basis der Transaktionskostentheorie. WIRTSCHAFTSINFORMATIK 46(5):341–352

Patterns are more than just Solutions to a Problem in a Context

With design patterns, many readers of BISE associate what they can find in the book "Design Patterns: Elements of Reusable Object-Oriented Software" (Gamma et al. 1995): Solutions to technical problems of object-oriented programming which are preferably applied in a particular context. The fact that patterns originally have been written by the (building) architect Christopher Alexander on a larger scale and have been introduced to the community of object-oriented software developers around 1987 by Kent Beck and Ward Cunningham should be mentioned here for the sake of completeness. If you are interested in the history, you will find a good overview e. g. in the following sources: Portland Pattern Repository n.d.; Coplien 1996.

Meanwhile, patterns in many areas of software development and also in other disciplines exist. The question was why not use patterns in BISE on a larger scale. However, one may first wonder about the benefits of using patterns.

Writing patterns yourself

Superficially, one would assume that the benefit comes from the fact that it is easier for a software designer (or designer in another area where patterns exist) to build good solutions if it is possible to reuse the solutions of other, more experienced designers. This (and nothing more) constitutes the benefit most of the users see in using patterns. However, there are a lot more benefits resulting from the use of patterns. These are obtained especially when writing patterns. Different pattern forms and the guidelines of experienced pattern authors make you go into the question why a solution is good. The conflicting powers that shape a solution are explicitly made apparent by so-called forces. If you work on this, you will learn to clearly justify design decisions and to first analyze the environment of a future solution before you arrive at the famous solution for which unfortunately no problem can be identified.

Productive reviews

Another benefit arises as a result of pattern conferences where patterns of different authors are examined and discussed after several rounds of feedback in a so-called writers' workshop. If you are interested in the HOW, we again refer to Coplien (1996). Apart from the fact that authors receive mostly good suggestions for improving design results during the process before the workshop, the so-called shepherding, they also get to know a very productive review process that is based on the fact that any feedback provider is forced by rules and rituals to only state observations that are useful for the pattern author.

Thus, we can sum up the interim findings as follows: The study of patterns – preferably by actively writing them – is of high value for practitioners in multiple ways: Good solutions are found faster and better justified. Reviews become extremely productive.

Patterns and academia

We can only speculate on why patterns in BISE in particular and in the academia

in general have not prevailed on a large scale. However, it is fact that completely different objectives are pursued when writing a large amount of patterns or a dissertation.

Patterns are not under the obligation to be original. On the contrary, the principle of "3 independent known uses" applies – something is only a pattern if it can be proven that a solution has been deployed in at least three different situations – and in a productive context at best, such as in the economy.

Dissertations based on the fact that the concept has already been described three times cannot exist. Research culture thrives on originality and new solutions – regardless of whether there is a corresponding commercial problem or not. Whether this makes economic sense is not to be discussed here. It is definitely a culture that is deeply rooted in the genetic code of the academic establishment.

This means: Patterns do not really help a research assistant on continuing his/her way to the thesis. You can then write patterns as a hobby. A collection of patterns will, however, only be used in rare cases in a dissertation.

Pseudo patterns

Another component of being an academic is that we should publish. As shown, you get very helpful feedback through the review process of patterns, and the culture of immediately refusing "self-invented solutions" does not exist. Moreover, the majority of pattern conferences produce a quotable publication, which is – due to the positive and helpful pattern community – often more easily accessible than conventional journals where only approval or rejection exists, but rarely elaborate review processes and workshops with experts are offered. Therefore, pattern conferences seduce many academics to submit things that, try as they might, do not have "3 independent known uses" – because they describe their latest inventions in the form of patterns.

Principle of non-originality

It can thus be stated: Patterns are not the best way for "real academics" to arrive at especially respected publications. Both communities have different objectives: The pattern community intends to document known knowledge so that it is perfectly usable and can be optimally combined. Here, it is exactly the non-originality which constitutes the value of the community. Research aims at generating new knowledge and new concepts. Here, in case of doubt originality is more important than immediate practical utilizability. Both cultures can coexist without problems, learn from one another, and stimulate each other. Border crossings will often be productive.

Wolfgang Keller
objectarchitects

References

Coplien JO (1996) Software patterns. SIGS Management Briefings

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: Elements of reusable object-oriented software. Addison-Wesley Longman, Amsterdam

Portland Pattern Repository (nd) History of patterns. http://c2.com/cgi-bin/wiki?HistoryOfPatterns. Accessed 2009-07-11

Patterns in Business and Information Systems Engineering

In software engineering, patterns, which originally have been derived from architecture, have received high interest for more than 15 years. Before discussing the reasons why the pattern approach have hardly found application to BISE-typical questions so far, we give would like to comment on how patterns are devolved (see for example http://hillside.net/patterns/writing/writingpatterns.htm): "Good" patterns are tried and tested and are accepted as such by the community. They thus require practical know-how and experience, in many cases appropriate knowledge of the line of business and a community in which they are discussed and refined.

Business consultants are predestined for the development and evaluation of "BISE Patterns". Formulated in an exaggerated way, it is their business model to have experienced as many circumstances in different companies as possible and then to be able to assess which solution fits the situation in question. Consultants therefore constantly apply patterns, but they are not formulated as such. Indeed, frequently consultants and consulting firms have no interest in making their expertise available to third parties. The same applies to the companies differentiating themselves by their business models and/or the business models underlying their products and business processes in the market.

Why is this different in software engineering? In this area, a different spirit seems to prevail: There are enough experts who like to talk about their experience and are also willing to publish them. To begin with, software patterns are usually quite distant from the differentiating elements of the companies. Another reason certainly is that the development of software systems is difficult enough; patterns alone do not enable the development of such systems. In contrast, many BISE questions focus on the design of a concept, to which patterns may quite well contribute large parts of the solution.

The central technique of BISE is modeling. Consequently, in BISE new modeling methods, often with a focus on modeling languages, are constantly developed, proposed, and evaluated. This is fully justified, but users are looking for a solution to their problems. For this purpose, in many cases patterns formulated as a solution would be of greater help than methods for the preparation of solutions. One approach in BISE to describe solutions in a reusable way are reference models. However, many practical projects show that, for example, reference business process models make limited contributions. No later than at the execution level, the differences between reference and actual solutions become too large. Anyone acquainted with the business processes of several companies in an industry will be able to confirm this problem. Moreover, the used modeling languages usually do not include inherent concepts to adapt the reference models to their own needs. This is the job of complementary methods. In examining these methods, many similarities to the description of patterns are found. Here, we consider the description of patterns as an approach to better yield the desired results. The great success of ITIL (IT Infrastructure Library) shows that this is possible – even if the described "best practices" (unfortunately) are not specified as patterns.

We see little difference between the problem solving process and "normal" business processes. In the latter, although not referred to as modeling, the business objects dealt with in e. g. financial accounting, controlling, and also in the core busi-

ness processes indicate many properties of models. Thus, if patterns can be formulated for business processes, the same can be done for the problem solving process, i. e. for the process from a modeling perspective: who creates the models, updates, releases them and how they are used. A demand for such patterns certainly exists. Successful examples can be found again in software engineering with the description of process models and their tailoring concepts as patterns.

For BISE patterns it should be noted, however, that the domain is a great deal larger than in software engineering. Patterns can relate to different objects under consideration, such as business models, business processes, organizational structures, services, etc. Additionally, the industry should be considered as a further dimension. Many practitioners – and their cooperation is essential in the formulation of patterns – will only be authorized to cooperate by their company if they are dealing with areas that have no potential for differentiation. The public administration and parts of the health sector demonstrate clearly, however, that this can also work in entire sectors. ITIL, which initially originated in the public administration sector and which refers to the IT area increasingly becoming a commodity, is also a good example. As regards differentiation potential, it should be noted that this in future will increasingly shift to the expert level by means of concepts such as SOA (service-oriented architectures) and SaaS (software as a service). Thus a corresponding need and great potential for patterns for the identification and use of services will emerge – especially because the problem of the semantic description of services is still unresolved.

In summary, the authors would welcome a development in which patterns played a greater role in BISE in future; but we are skeptical whether the above mentioned reasons do not oppose this in parts.

Dr. Stefan Junginger
Christoph Moser
BOC AG, Vienna

Morning Star, Evening Star, and Venus – On the Use of the Words "Reference Model" and "Pattern"

While in the past people believed that the morning and evening star are two different stars, we now know that the planet Venus appears as morning star in the eastern sky in the morning and as an evening star in the western sky in the evening. We conceive the situation with reference models and patterns in a similar way: We believe that reference models and patterns represent two forms of the same central idea. Therefore, we propose not to distinguish terminologically between reference models and patterns anymore.

Presumably, our proposal will meet with opposition: For example, the review of relevant literature shows that so far there has been hardly any scientific exchange between the reference model community and the pattern community. Nevertheless, we recommend to no longer make a distinction between both approaches in future.

We acknowledge that the situation was different in the early 1990s: Scheer developed the Y-CIM model as a classical reference model, Gamma laid the foundation of the pattern community with his dissertation published in 1992 which was put on a broad basis with "Design Patterns" by Gamma et al. in 1995.

The initial ideas of a reference model or a pattern are clearly different:

- Characteristic "level of abstraction": Classical reference models describe functional aspects of an information system. In contrast, traditional patterns describe implementation-technical aspects of a software system.
- Characteristic "domain reference": Classical reference models describe business application domains such as industry, trade, and services. Traditional patterns have no relation to business domains.
- Characteristic "granularity": Classical reference models are coarse-grained, they describe an entire company. Patterns usually describe only a small aspect of a software system.
- Characteristic "claim": Classical reference models claim to scientifically formulate knowledge about an application domain. Classical patterns do not raise this claim. Instead they explicitly point out that they are useful within software development practice (rule of thumb: "A pattern must be used at least three times in a real project to be considered as such.").
- Characteristic "language": Classical reference models use event-driven process chains, entity-relationship models, or function trees as a modeling language. Classical patterns are based on object-oriented language solutions.

In recent years, however, the original ideas have changed radically:

- Characteristic "level of abstraction": Reference models are also used for technical aspects (Boles 2002), many patterns now also cover functional aspects (Fowler 1997).
- Characteristic "domain reference": Along with the shift of the degree of abstraction, reference models are losing part of their domain reference and patterns sometimes obtain a domain reference.
- Characteristic "granularity": In reference modeling there is now an abundance of work also trying to describe fine-grained models (Remme 1997). At the same time coarse-grained patterns, for example in the form of "pattern languages", are described (Evitts 2000).
- Characteristic "claim": Often, the explicit claim of scientific knowledge is no longer related to reference models; instead, "only" the reuse is emphasized (Fettke and vom Brocke 2008). At the same time, we find patterns that lay claim on scientificality (Sinz 1998; Tichy 1997).
- Characteristic "language": Meanwhile, there are also object-oriented languages that are used in the formulation of reference models (Schlagheck 2000). Similarly, patterns are not limited to object-oriented concepts anymore (Ambler 1998).

In addition, there are developments that are observed in both communities: While both classical reference models and classical patterns represent artifacts for the representation of knowledge about parts of reality, now also systematic methods to perform similar tasks are discussed in both communities – referred to as reference process models (Vering 2002) or process and workflow patterns (van der Aalst et al. 2003).

The above mentioned examples illustrate that reference models and patterns can be distinguished clearly in a classical understanding. But: Based on which characteristics can a line be drawn in a

meaningful way between the two concepts *today*?

Since we consider discrimination between both concepts based on the above mentioned features as too arbitrary, we propose to drop the terminological differentiation of both concepts and highlight only the central common features of the original ideas: In the development of information systems, actions for the planning, development, implementation, operation, and application of information systems are analyzed. Proposals for the execution of such acts are discussed in particular by *specific instruments*. The word "instrument" is used here in a very broad sense: It denotes "not only methods in the narrow sense of systematic procedures for the performance of similar tasks, but also models as artifacts to represent knowledge about reality, techniques as task- or problem-related combinations of methods and models as well as tools as computerized implementations of techniques" (Zelewski 2009). The common idea of reference models and patterns, the "Venus" so to speak, is to identify *ideal-typical* instruments and to document them publicly. The knowledge of instruments produced in this way can be both scientifically analyzed and taught as well as used in practice.

Our proposal for the standardization of terminology is that the words "reference model" and "pattern" are used to describe ideal-typical instruments for the development of information systems. The mentioned characteristics can be referred to for further linguistic differentiation. It is therefore in the eye of the beholder to what extent the words "reference model" or "pattern" are used for the description of ideal-typical instruments.

PD Dr. Peter Fettke
Prof. Dr. Peter Loos
Institute for Information Systems
German Research Center for
Artificial Intelligence (DFKI)

References

Ambler SW (1998) Process patterns. Building large-scale systems using object technology. Cambridge University Press, Cambridge

Boles D (2002) Integration von Konzepten und Technologien des Electronic Commerce in digitale Bibliotheken. Dissertation, Universität Oldenburg

Evitts P (2000) A UML pattern language. MTP, Indianapolis

Fettke P, vom Brocke J (2008) Referenzmodell. In: Kurbel K, Becker J, Gronau N, Sinz EJ, Suhl L (eds) Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/, Oldenbourg, Munich

Fowler M (1997) Analysis patterns: Reusable object models. Addison-Wesley, Menlo Park

Remme M (1997) Konstruktion von Geschäftsprozessen – Ein modellgestützter Ansatz durch Montage generischer Prozeßpartikel. Gabler, Wiesbaden

Schlagheck B (2000) Objektorientierte Referenzmodelle für das Prozess- und Projektcontrolling. Grundlagen – Konstruktion – Anwendungsmöglichkeiten. DUV, Wiesbaden

Sinz EJ (1998) Geschäftsprozessmodellierung mit Patterns. In: Becker J, Eversheim W, Luczak H, Mertens P (eds) Referenzmodellierung '98. Anwendungsfelder in Theorie und Praxis, RWTH Aachen, pp 3–1 to 3–10

Tichy WF (1997) A catalogue of general-purpose software design patterns. In: Technology of object-oriented languages and systems (TOOLS). Santa Barbara (CD-ROM)

van der Aalst WMP, van Dogen BF, Herbst J, Maruster L, Schimm G, Weijters AJMM (2003) Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering 47:237–267

Vering O (2002) Methodische Softwareauswahl im Handel – Ein Referenz-Vorgehensmodell zur Auswahl standardisierter Warenwirtschaftssysteme. Logos, Berlin

Zelewski S (2009) Wissenschaftstheorie. In: Kurbel K, Becker J, Gronau N, Sinz EJ, Suhl L (eds) Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/, Oldenbourg, Munich