INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITÉ DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

PATTERN LANGUAGES: AN APPROACH TO MANAGE ARCHETYPAL ENGINEERING KNOWLEDGE

Jörg Feldhusen[1], Frederik Bungert[1]
[1] Chair and Institute for Engineering Design IKT, RWTH Aachen University

ABSTRACT
Pattern languages provide a framework to manage archetypal knowledge and make it easily accessible. The concept was originally invented in the 70s by Christopher Alexander as an approach to capture design experience in civil engineering as well as architectural design and to communicate the according solutions with customers. Currently, the approach is commonly perceived in the software community. Pattern languages have been applied to many fields within computer science. They have gained a strong influence on those fields, especially software engineering and programming as well as human-computer interaction (HCI) design. In engineering design as well as Product Lifecycle Management (PLM), pattern languages have yet not been applied.
Engineering design and product lifecycle management are very similar to civil engineering, architecture design and computer science concerning the creational aspects of these areas and their interdisciplinary collaboration as well as customer awareness. This gives a strong evidence for the expedience of pattern languages to be applied likewise to engineering design and PLM and provide a similar benefit.
A huge amount of knowledge has been gained in the context of engineering design and PLM. This knowledge is only loosely connected and lacks of integration so far. The aim of applying pattern languages to engineering design and PLM is to develop a cohesive theory, which integrates current archetypal knowledge and makes it continuously applicable during the whole product lifecycle. This paper introduces the concept of pattern languages and describes the application to engineering design and PLM.

Keywords: Pattern Language, Knowledge Management, Product Lifecycle Management (PLM), Engineering Design

1 INTRODUCTION
The management of engineering knowledge is a crucial element of Product Lifecycle Management (PLM) and one of the key enablers for economical success of manufacturing companies. Experiences from several decades of industrial production show that products and product creation processes are getting more and more complex. To overcome the increasing complexity of current products and processes, engineers need to have appropriate skills and knowledge. Thus, tools and methods to provide engineers with knowledge are getting more important to ensure the competitiveness and sustainability of a company.
Archetypal knowledge denotes all knowledge whose validity is independent from time and specific use cases. It constitutes a precondition to acquire and apply specific knowledge which has a certain period of validity and/or is specialised regarding to the referred objects (special knowledge) or a certain group of persons (expert knowledge). Thus, the availability of archetypal knowledge can be seen as a key success factor in knowledge management. Accordingly, approaches to maintain and provide archetypal engineering knowledge are required. Pattern languages where originally invented to manage archetypal architectural knowledge. They have been commonly applied in the software community for the same purpose. Like described in the following, the pattern framework is likewise expedient to be adopted in the area of PLM and engineering design.
Chapter 2 gives a short overview about current approaches to provide knowledge, which are expedient to be applied in the area of PLM. Chapter 3 describes in general the concept of pattern languages and proposes background patterns as a concept to amend patterns by background knowledge comprising

theoretical evidence corroborating the statements of the pattern and thus turning a pattern language into a holistic theory. Further, a layer concept will be delineated to combine pattern languages with individual knowledge to make them more expedient for industrial application. Chapter 6 exposes aspects concerning the connection of several sublanguages required to constitute a PLM-pattern language. In chapter 4 the application of pattern languages to PLM and the sub-areas of PLM like engineering design will be discussed, before some examples will be given in chapter 5. Chapter 7 finally summarizes the main contributions of this paper and concludes with an outlook on further research.

2 CURRENT APPROACHES

The pattern approach can be seen as a knowledge management approach. Within the field of engineering design, approaches with formal knowledge representation, like Ontologies [1], have been commonly researched. Regarding the model by Nonaka/Takeuch [2], those approaches aim at combining knowledge, i.e. generating new knowledge by automatically processing existing knowledge. However, a pattern language is a semi-formal approach with higher semantic power, which is located nearby “informal” within the spectrum between “formal” and “informal”. This spectrum incorporates a so called semantic gap [3]. A semantic gap denotes the discrepancy between the information content of two representations referring the same issue. Ontologies and other combinatorial approaches furthermore differ from pattern languages in their way of processing a query. Whilst those approaches require queries initiated by the user, a pattern language directly supports the problem solution process.

In accordance with Nonaka/Takeuch [2], a pattern language primarily supports the internalisation of knowledge. Further approaches of that kind are Content Management Systems (CMS) [5], like for example engineering guidance systems [6], and books. Both support the internalisation of knowledge. However, CMS-based approaches on the one hand are designed to provide special knowledge instead of archetypal knowledge. Books on the other hand are hardly appropriate to represent interrelations within knowledge.

A pattern language can also be seen as a theory. One characteristic of a theory is a constituting structure. Engineering design theories, for example Pahl/Beitz [4], are structured according to the temporal order of the design process. However, some engineering patterns, like lightweight construction for example (chapter 5), cannot be integrated in terms of temporal criteria. Hence, current engineering design theories do not comprehensively cover the field of engineering design. For similar reasons, a holistic PLM theory does not exist so far. Most concepts and methods in PLM cannot be related according to their temporal sequence.

3 PATTERN LANGUAGES

3.1 The Pattern Approach

Some current approaches described above are capable to support the retrieval of knowledge according to a certain context utilizing structures or indices. But these approaches focus on specialized ephemeral knowledge. Common knowledge, which every single engineer needs to have at his disposal to originate state-of-the-art results, can still be primarily found only in books. Pattern languages are an approach to provide common knowledge in an easily accessible way. Beyond this, pattern languages aim at some further properties, which are not objectives of other knowledge management approaches (chapter 2). Pattern languages were originally invented by Christopher Alexander [8] [9] [10] in the 70s. They were first applied in civil and architectural design to capture design experience and to communicate the according solutions with customers. During the last decade the approach has been bruit about in the software community. Currently, pattern languages have been applied to many fields within computer science. They have gained a strong influence on those fields, especially software engineering and programming [11] [12] [13] [14] as well as human-computer interaction (HCI) design [15] [16] [17]. Even in pedagogy pattern languages have been applied [18].

In engineering design as well as Product Lifecycle Management (PLM) pattern languages have yet not been applied. The domains described above, in which pattern languages have proven to be expedient, are similar to PLM concerning their design orientation, interdisciplinary collaboration and customer awareness. Thus, pattern languages are supposed to provide a similar benefit, if they are applied within PLM.

3.2 Syntax and Semantics

The central idea behind pattern languages is that, due to complexity, humans have evolved archetypal concepts, which solve recurrent problems. These concepts are called patterns. Within a pattern language all patterns have a uniform format and structure. The structure of the original patterns by Alexander is constituted by a set of content types [8], which can be found with only minor amendments in other patterns languages:

- The unique and descriptive *name* of the pattern.
- A *ranking* of its quality (omitted in some other pattern languages).
- A *picture* illustrating an example of the pattern.
- The *context*, in which the pattern can be applied. The context consists of the patterns which are supported by the actual pattern and thus comprise an according supporting pattern reference (see below).
- A short *statement* delineating the recurring problem referred by the pattern.
- A detailed description of the *problem* comprising a comment on the conflicting forces, which cause the problem. In case of PLM most of these forces are of technical (e.g. an engine should be powerful and ecological), economical (e.g. a product has to be cheaply producible and competitive) or organizational nature (e.g. short time-to-market and lean value chain). The solution must describe how the forces can be balanced or resolved.
- Situations in which the described problem occurs together with a description of the solution are delineated in the e*xamples* section. The Examples should refer to widely known application and explain how those applications solve the problem. Examples are important on the one hand to lead novice readers towards the general solution of the pattern via a comprehensible set of instantiations of this solution. On the other hand they give professional readers empirical verifiable evidence of the validity of the pattern.
- An archetypal *solution* of the problem illustrated by a diagram.
- References to *supporting patterns*. These references point to other patterns, which "unfold" the solution of the current pattern, for example the pattern "Road Crossing" is supported by the pattern "Raised Wall".

Patterns do not stand for themselves. Like described above, they comprise references to supporting patterns. This results in a hierarchical structure whose elements range from a most general top-level pattern over larger-scale patterns to elementary ones (Figure 1). Each elementary pattern can be reached via one or even more ways from the top level pattern. Thus, the set of all patterns belonging to the same language constitutes a semilattice (a partially ordered set closed under either supremum or infimum) with the supporting pattern references as ordering relation.

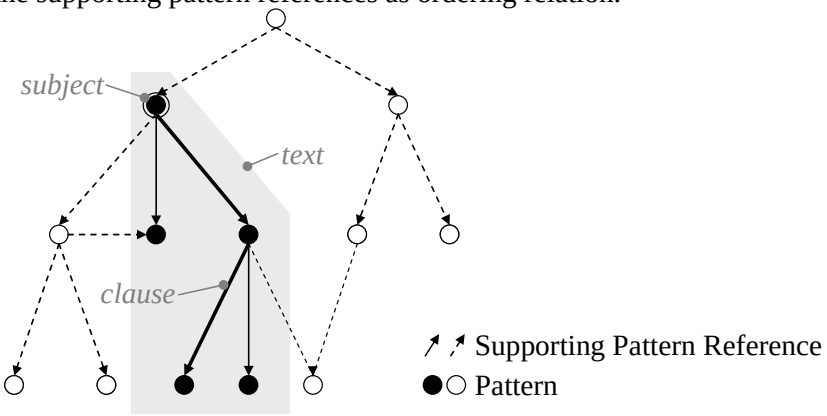


Figure 1. The structure of a pattern language.

This concept can be compared to a spoken language. A spoken language comprises words and grammatical rules, which describe how the words can be related to each other resulting in a clause. In case of patterns the set of supporting pattern references can be seen as the grammar of the pattern language. The clauses generated by this grammar are sequences of patterns describing constitutive solutions. If a problem corresponding to a certain pattern occurs, a subset of supporting patterns which is subject to his attention is chosen by the user (architect, software developer, designer …). The other supporting patterns are discarded. Each chosen pattern in turn entails a further subset of chosen

supporting patterns. As a result, a sublattice evolves with the original problem as the join. The paths down from the joint constitute a set of clauses with the original problem as the subject. This set can be seen as a text, which is written in the pattern language, expressing the user's ideas of solving the problem (cp. Figure 1).

3.3 Background Patterns

The structure of pattern languages primarily aims at making knowledge easily applicable. It comprises only empirical, weak evidence given by the examples for the validity of each pattern. Nevertheless, the context and supporting pattern references establish a network which assembles pieces of knowledge to a coherent whole.

Background patterns are an approach proposed by the authors to amend each pattern by background knowledge, comprising theoretical evidence corroborating the statements of the pattern. By this background knowledge a pattern language is turned into a holistic theory covering the application area of the pattern language. A background pattern is assigned to a single PLM pattern and has no further links to other patterns. It comprises the scientific theory underlying the PLM pattern. Similar to the structure of conventional patterns which has proven to be of value the background patterns are to be structured consistently. Therefore, two alternative pairs of content types seem to be expedient: *These* with *evidence* and *actuality* with *explanation* respectively. Each background pattern may consist of an arbitrary amount of such pairs depending on the statements to be corroborated. At the end of each background pattern, the referenced literature should be listed.

Background patterns make the pattern concept just as expedient for engineering education. Conventional education media like books, scripts or lecture slides present the subject matter in a purely sequential way. References between content elements which constitute each other would improve the comprehensibility. Such references are provided by the context and supporting pattern references of a pattern language. The pattern language provides an overview of the subject matter, the content of teaching is comprised by the background patterns. Thus, the pattern concept amended by background patterns constitutes an approach to give a subject matter an intuitive structure and increases the pedagogic efficiency as well as effectiveness. Furthermore, the context and supporting pattern references support the lecturer in proving whether the subject matter is covered completely.

3.4 Philosophical Considerations

Behind the idea of pattern languages there is a certain philosophical theory, which delineates the intention of this approach: because patterns are supposed to refer to recurring problems and archetypal solutions, a pattern language describes a timeless way of the application domain (cp. [9]). Thus, pattern languages persist over decades and are applicable regardless of ephemeral realization aspects. Many outstanding archetype concepts, which are intended to be described by patterns, imply a certain hardly describable quality like an inner beauty. Alexander names it the quality without a name [9]. This quality cannot be reduced to a single dimension [15] but is to be captured in a pattern. A living pattern language is considered to be the gate to be passed to reach the quality without a name [9]. The usage of a pattern language constitutes the timeless way mentioned above, leading through the gate.

4 TOWARDS A PATTERN LANGUAGE FOR ENGINEERING DESIGN AND PLM

4.1 The Expedience of the Pattern Approach concerning Engineering Design and PLM

In architecture, the pattern approach has not been received well, because Alexander's concept enabled the inhabitants (customers) to influence the design process and take some influence away from the professionals [15]. Obviously, architects were not inspired to apply pattern languages. In contrast to that, PLM and engineering design professionals are encouraged by the management to cooperate with customers. Thus, there is some potential for pattern languages to be bruit about in these areas. The example of software development, where pattern languages are already very popular, endorses this assumption.

Currently, a bunch of methods has been developed in the context of PLM [22] [23] [24]. These methods on the one hand and the methods and theories of the sub-areas within the lifecycle on the other hand are only loosely connected and show a lack of integration so far. The aim to be reached by

the application of pattern languages to PLM is to develop a cohesive PLM-theory, which relates current PLM-methods to each other, integrates them with the methods and theories of the sub-areas and makes the evolving comprehensive set of methods continuously applicable during the whole product lifecycle as an interdisciplinary lingua franca.
A common motivation for the application of pattern languages throughout the various disciplines, which is also expedient for PLM, is to capture archetypal knowledge and initiate a generative problem solution process based on this knowledge. A major objective of Alexander's work is to share the knowledge incorporated by the pattern language with the inhabitants (customers) and facilitate them to participate in the design process [9]. For similar reasons pattern languages are applied to HCI design [6]. Users on the one hand do not understand the technical jargon of software engineers and software engineers on the other hand have only a vague cognition of the application domain. Pattern languages are applied to establish communication between both parties. In general, the language-like concept of pattern languages described in section 3.2 makes this approach capable to become a common lingo, which facilitates collaborative work between experts from different disciplines as well as laymen like customers.

4.2 The Structure of a PLM Pattern Language

Figure 2 depicts a framework for a PLM pattern language comprising the sub-areas of PLM. One of these sub-areas is constituted by engineering design. Therefore, a pattern language for PLM will comprehend a pattern language for engineering design. Dependencies between different areas in Figure 2 are represented by overlapping circles. For example, strategies determine the operational methods to be applied. Operational methods in turn determine how PLM-supporting systems have to be arranged and configured. The approaches delineated in chapter 6 can be applied to integrate overlapping sub-areas so that a PLM pattern language evolves.

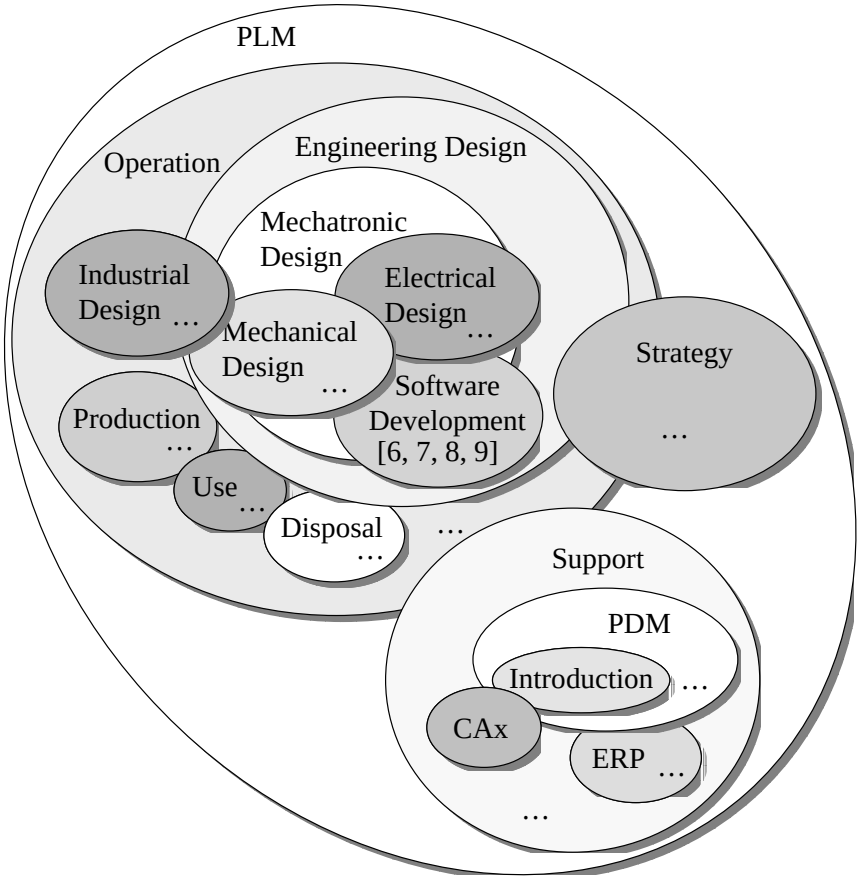


Figure 2. A language framework for PLM.

To develop a complete pattern language for PLM, the grammar of the language has to be developed as well as the structure of a pattern and their content. The grammar of Alexander's language structures pattern according to the spatial structure of the addressed design objects, i.e. patterns addressing large-

scale objects refer to patterns addressing comprised small-scale objects. PLM is more complex than architecture, because it integrates different design objects with various levels of details, different lifecycle states and various engineering disciplines. Unlike the objects of architecture (towns, buildings, rooms and furniture) the design objects of PLM are not homogenous. Instead, PLM patterns refer to abstract concepts. The spatial structure is therefore not an expedient concept to structure a PLM pattern language. A PLM pattern language might be structured according to the level of abstraction of a pattern. This is compliant with the concept of the supporting pattern reference because commonly more concrete concepts are applied to realise abstract concepts. A second structuring criterion of a PLM pattern language might be the orientation towards the product lifecycle. Each process step within this lifecycle is followed by consecutive steps to elaborate the design object. Thus, the lifecycle state, as a structuring criterion, is also compliant with the concept of supporting pattern references. If a sublanguage focuses continuously on a certain design object, the spatial structure of this object might be amended as a third structuring criterion. A structuring criterion regarding the involved engineering disciplines is not expedient, because one aim of the PLM pattern language is to create a common interdisciplinary lingua franca. The inner structure of a pattern proposed by Alexander ([3], cp. section 3.2) seems to be largely appropriate to be transferred to PLM patterns.
The framework depicted in Figure 2 gives an overview of the content covered by the PLM pattern language. It divides the area of PLM into three major segments: superordinate strategic approaches, operational approaches to realize value-added chains according to the strategies and approaches to support the effective and efficient progress of value chains. As an integrative approach it comprises not only PLM-strategies and supporting tools, but also content from various sub-areas within industrial design, engineering design, production, use and disposal. The integration of this content makes the development of PLM pattern languages a challenging, but very rewarding mission. Amongst others, the development of sub-languages describing the following topics seems to be of particular interest:

- PLM-strategies and their dependencies on the operational level,
- integration of customers into the design process,
- collaboration between industrial design and engineering design,
- mechatronic design (collaboration between mechanical design, electrical design and software development),
- integrated representation of current approaches in mechanical engineering,
- collaboration between engineering design and production planning,
- arrangement and configuration of PLM-supporting systems according to operational approaches,
- interconnection of PDM- and ERP-Systems and introduction of PDM-systems.

5 EXAMPLE: THE LIGHTWEIGHT CONSTRUCTION PATTERN

Lightweight construction is a sub-discipline of engineering design which in turn can be seen as a discipline within PLM. It has become increasingly important to build a wide range of competitive products for various applications. The lightweight construction pattern is delineated in Table 1 to illustrate the application of the pattern concept to engineering design.
Like the other patterns of the PLM pattern language, the structure of the lightweight construction pattern follows the structure proposed by Alexander [8] and described in section 3.2:

- *name*,
- *ranking* (omitted here),
- *picture*,
- *context* (indicated by capitals), *statement* +++
- *problem*,
- *examples*,
- **therefore:** *solution* (represented textually and graphically) +++
- *supporting patterns*(indicated by capitals).

Like the pattern language of Alexander, the PLM pattern language comprises no explicit labelling. Instead, a consistent layout helps to identify the content types. This Alexandrian style makes the pattern language more readable.

Table 1: The lightweight construction pattern amended by a background-pattern.

LIGHTWEIGHT CONSTRUCTION[1]

… the aim to design lightweight products is common in many branches of industry. In many cases lightweight construction is required to ensure the proper functioning of a product – HIGH DYNAMIC, BUOYANT FORCE, SPAN LENGTH. It is also applied to save indirect costs incurring during usage – ECO-DESIGN. A side-effect which occurs in some cases is the reduction of direct costs caused by material and manufacturing – DESIGN TO COST.

+++

A major requirement towards a certain product is a low weight. Further requirements have to be fulfilled likewise. Above all, the product has to serve its purpose. This requires several kinds and quantities of material which constitute the product and provide the desired properties and functionality. However, the more material a product comprises the more heavy-weight it will be.

A car for example should accelerate fast and consume little fuel. Additionally, it has to provide high safety to the passengers in case of a crash. Its body should be stiff to increase the comfort and straight running ability. Because in general the architecture[2] of cars is predefined the delineated conflict can only be solved by changing the base materials of the car. Therefore, often high-strength steel or aluminium are applied to build cars.

Similar to cars, motorbikes should accelerate fast, consume little fuel and be stiff. Thus they are likewise built of high-strength steel and aluminium. For motorbikes a further solution exists: The engine can be a load-bearing component of the frame. By integrating additional functions into the engine it gets heavier, but at the same time the frame gets more lightweight. As a result, the whole motorbike gets more lightweight.

The purpose of an aircraft is constituted by its ability to fly. Furthermore, it should consume little fuel and at the same time be rugged and durable. Because an aircraft offers higher saving potentials regarding fuel consumption than a car, nowadays it mainly consists of more expensive composite materials.

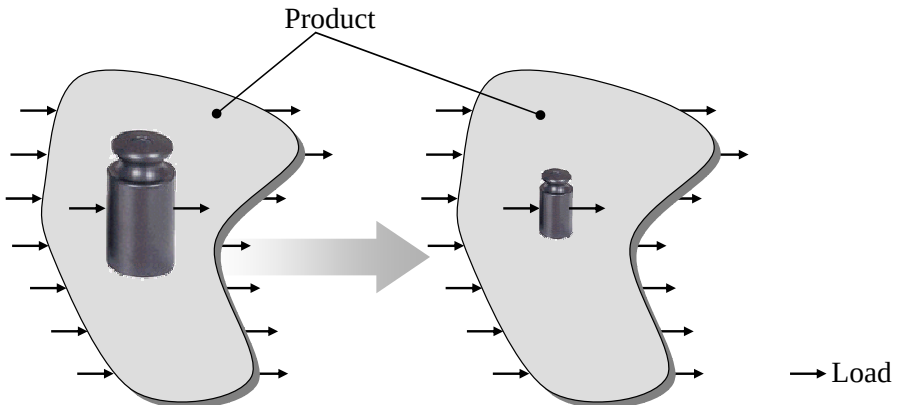
The moving parts of a machine tool have to be lightweight in order to provide high dynamics. Hence, the topology of these parts will be optimised. This leaves material where it is required and removes it from places where it is dispensable.

A bridge is an example which has to be rugged and lightweight at the same time to realise a high span length. Thus, the topological structure of the bridge will be optimised and several kinds of lightweight solid and composite materials will be applied.

Aluminium can be applied in case of steel to save material costs because the specific volume of aluminium is higher. Hence, less bracing is required [1].

Therefore:

Increase the specific load[3] of the product by increasing the specific load of distinct areas of the product[4]. This will reduce the weight of the product, but preserves its other properties.



+++

The specific load of a certain area of the product can be reduced by adjusting the specific properties[5] of this area to the load inducted by the product and optimising its specific properties. Therefore, materials can be replaced by more lightweight materials with similar properties (besides weight) – LIGHTWEIGHT MATERIALS, the topological structure can be optimised according to the flow of forces – TOPOLOGY, the specific properties of different material can be combined – HYBRID STRUCTURES – or functions can be integrated – INTEGRAL CONSTRUCTION – which may outweigh some components but lighten other ones…

Background:

[1] *Lightweight construction* is a sub-area of engineering design which aims at minimising the weight of a product or some of its parts [1]. The principles of lightweight construction are commonly applied amongst others in aircraft engineering, astronautics, automotive engineering, machine tools engineering and even shipbuilding.

2 The *product architecture* denotes a correlation between the function structure and the physical structure of a product [2]. It denotes the function(s) which a certain product component fulfils and in turn, the product component(s) which fulfil a certain function.

[3] According to Pahl & Beitz [3] and Koller [4] a product is loaded by several kinds and quantities of energy, force, material as well as data. In turn, the product loads its environment by these quantities. The load is transferred from input to output by the segments of the product. Therefore, each segment transfers some load. Analogous to the specific volume (quotient of volume and mass), the *specific load* is constituted by a combination of the load of a component or segment and its reciprocal mass. A component has a higher specific load than another one, if it can process the same load but has less mass.

[4] The principle of reducing the specific load is the base principle which is common to the measures applied in lightweight construction.

[5] The s*pecific properties* denote the properties of a component or segment combined with its reciprocal mass. Therefore, the specific properties can be seen as the maximum specific load accomplished by the component.

[1] Wiedemann J. *Leichtbau. Elemente und Konstruktion.* 2006 (Springer, Berlin, 3. Aufl.).

[2] Göpfert J. *Modulare Produktentwicklung. Zur gemeinsamen Gestaltung von Technik und organisation*. 1998 (Dt. Univ. Verl., Wiesbaden).

[3] Pahl G. and Beitz W. *Engineering Design - A Systematic Approach*. 1999 (Springer, Berlin).

[4] Koller R. Konstruktionslehre für den Maschinenbau: Grundlagen zur Neu- und Weiterentwicklung technischer Produkte. 1998 (Springer, Berlin).

6 HOW CAN PATTERN LANGUAGES BE RELATED TO EACH OTHER?

PLM is a strategic approach to manage the entire lifecycle of a product and its components to ensure sustainable benefit. It can be seen as a meta-theory integrating theories and methods from discrete sub-areas within the product lifecycle as depicted in Figure 2. Thus, aspects concerning the interrelation

between several pattern languages are constitutive for a pattern language which integrates the engineering knowledge along the product lifecycle and additionally incorporates the strategic PLM-knowledge. The authors have developed concepts described in this chapter to relate different pattern languages to each other resulting in a more comprehensive superordinate language.

6.1 Include-Relation

A pattern language constitutes a semilattice (section 3.2).Each closed subset of a semi lattice in turn constitutes another semilattice. Thus, a pattern language 2 (Figure 3) can be included by a more comprehensive pattern language 1. This means, that language 2 is a subset of language 1. Each pattern and supporting pattern reference within language 2 belongs to language 1 as well. Even the relative complement of language 2 in language 1 is a pattern language (language 1-2). Language 1-2 is a discrete pattern language which can be interpreted as a pattern language supported by language 2. The context of language 2 is constituted by the path leading from the topmost pattern of language 1-2 to language 2.

The include-relation is particularly expedient to build an integrated PLM-theory, because each connection of pattern languages via an include relation in turn results in a pattern language. Additional effort to convert a certain construct to a pattern language is not required.

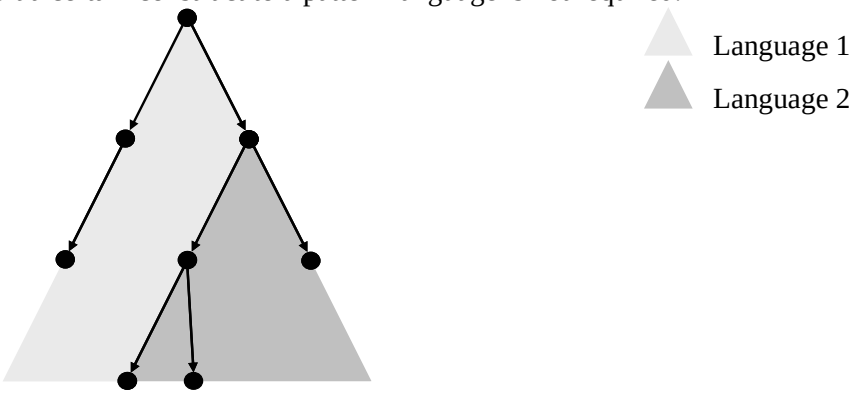


Figure 3. Language 1 includes Language 2.

Example: A sandwich element is a beam with a lightweight core element encapsulated by coatings which are rigid in strain [19]. Sandwich elements are lightweight but rigid and rugged. Lightweight construction focuses on the design of lightweight products. In most cases this conflicts with other properties which are required in order to fulfil the purpose of the product. In case of sandwich elements they have to be additionally rigid and rugged.

Thus, sandwich design can be seen as a sub-area of lightweight construction. A comprehensive pattern language which describes lightweight construction will therefore include a pattern language for sandwich design.

6.2 Intersection-Relation

Two pattern languages are related to each other by an intersection, if they have a shared included language. Figure 4, depicts an intersection of language 1 and language 2. They intersect each other, because Language 3 is included by both, language 1 and language 2.

Although the entire construct established by an intersection-relation is not a pattern language, this construct is constituted by pattern languages which are strongly connected by a further pattern language. Thus, the intersection of pattern languages is a relation which conjoins pattern languages to a cohesive construct and can therefore be considered as an expedient relation to build an integrated theory.

Example: Both, ecological design and functional design are a sub-area of engineering design. The intention behind ecological design is the creation environment-friendly and sustainable products, whilst functional design primarily aims at creating products which fulfil their purpose according to the customer's requirements. In case of ecological design the principles of lightweight construction are applied to reduce indirect costs [19], for example to reduce the fuel consumption of a car. Functional design involves lightweight construction in order to ensure the proper function of a product. The application of lightweight construction to the design of aircrafts as an example is a precondition to make them fly [19].

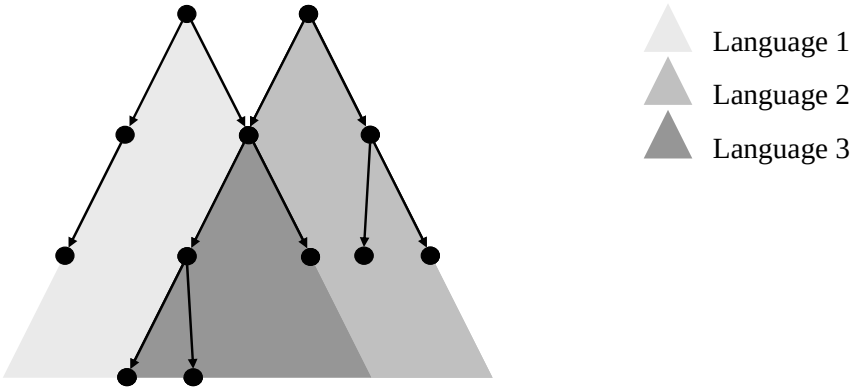


Figure 4. Language 3 constitutes an intersection of language 1 and language 2.

In ecological design as well as in functional design the applied principles of lightweight design do not vary. Therefore, the same pattern language for lightweight design can be applied in ecological design as well as in functional design. Pattern languages for ecological design and pattern languages for functional design therefore intersect each other and the corresponding intersection is constituted by a pattern language for lightweight construction. Furthermore, these three pattern languages are included by a pattern language for engineering design.

6.3 View-Relation

Several patterns languages are related to each other by a view-relation, if they describe different views of the same topic. These views may be for example constituted by certain areas of expertise which are involved in the creation process or may describe different aspects concerning the created object. The view relation is just a semantic concept. It does not establish syntactical links. Hence, pattern languages which are related by a view-relation are not connected to each other by any formal concept. To establish an integrated theory in case of view-relations, further concepts are required.

In many cases views are ranked and can be distinguished into primary and secondary views. A primary view is constituted by aspects which determine the quality of the created object regarding specific requirements. Secondary views contribute to the creation process by supporting the primary view. Accordingly, like depicted in Figure 5 the topmost pattern of the pattern language which describes the primary view (primary language, language 1) may be connected to the pattern language which describes the secondary view (secondary language, language 2) by a cross link. A cross link denotes a supporting pattern reference between different pattern languages. This results in a comprehensive pattern language comprising the primary and the secondary language as well. Moreover, the secondary language will be included by the comprehensive language. To make that construct transparent to the user, the supporting pattern reference constituting the cross link should be denoted by a standardised phrase which indicates the secondary language. Likewise, the context description of the topmost pattern of the secondary language should be amended by a standardised denotation of the primary language.

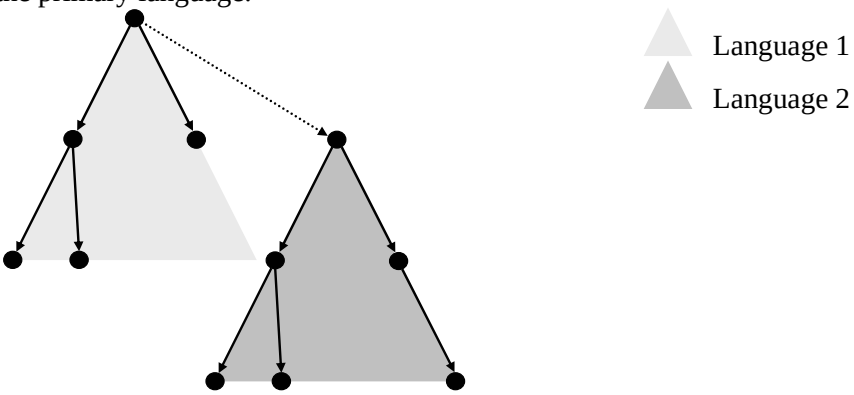


Figure 5: Pattern languages describing ranked views can be integrated by an additional supporting pattern reference.

Some views may be coequal and cannot be subdivided into primary and secondary views. Therefore, the accordant pattern languages cannot be integrated by a cross-link like described above. Coequal views indicate a conflict. If none of these views is primary, no view supports another view. Thus, each view has to be taken into account separately whilst competing with other views. The more a certain view will influence the creation of an object, the more the influence of the other views will be restricted.
A conflict between several views can be adopted to integrate the pattern languages which describe the conflicting views. Therefore, these pattern languages may be amended by an additional pattern, called view-pattern (Figure 6). A view pattern comprises a description of the conflict between the views and integrates the associated pattern languages. The resulting construct is again a pattern language.

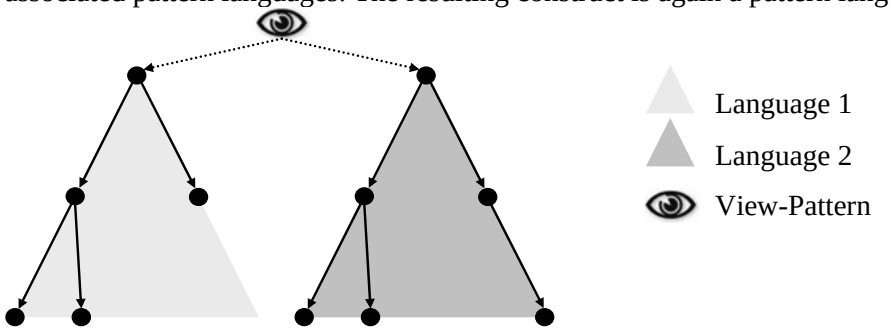


Figure 6. A view-pattern integrates pattern languages which describe coequal views.

Example 1: Similar to human computer interface design (HCI) [15] the design of consumer products considering the interaction with users may be described by a pattern language. This pattern language manly focuses on dedicated interfaces. Optical and haptic aspects (shape and surface) as well as sound and odour are covered by a pattern language about industrial design, which will be described below.
Another pattern language may describe aspects concerning the realisation of product interfaces similar to the pattern language about HCI realisation described in [15]. Both, the pattern language about interface design and the pattern language about interface realisation represent a different view on the same object (the interface). The pattern language about interface design describes the creation of the interface from the customer's perspective. The pattern language about interface realisation describes it from the engineer's perspective.
The quality of a user interface depends on its usability from the customer's view. Thus, the realisation of an interface is dependent on the design of the interface. This means, that the pattern language about interface design is primary, whilst the pattern language about interface realisation is secondary.
Example 2: Engineering design and industrial design constitute different views on the product creation process. A product fulfils not only technical functions, but also communicational functions [20] [21]. Communicational functions effect associations of the user, delineate the character of the product and indicate how to use the product.
Both, technical and communicational functions are a key factor for the economical success of a product. In most cases none of them supports the other one. In contrast, often technical and communicational functions are competing. This is corroborated by conflicts between engineers and industrial designers, which emerge frequently in companies. Therefore, pattern languages for industrial and engineering design may be integrated by a view-pattern which addresses these conflicts.

7 SUMMARY AND OUTLOOK

The pattern approach has been commonly applied software development and computer science. In this paper the application of pattern languages to PLM and engineering design has been discussed. As stated in section 4.1, there is strong evidence that pattern language can be expedient for PLM and engineering design. Approaches to relate different pattern languages to each other in order to connect them to an embracing construct have been proposed in chapter 6. Accordingly, by connecting several sub-languages which cover distinct sub-areas of PLM a comprehensive PLM-pattern language can be created like delineated in chapter 4. An example pattern to account for the practicability of this concept has been given in chapter 5.

Besides the lightweight construction pattern further patterns have already been developed. Nevertheless, a bunch of patterns still has to be amended to constitute a comprehensive PLM pattern language. These ongoing efforts will be published at *www.plmpatterns.org*.

REFERENCES

[1] Staab, S., Studer, R., *Handbook on Ontologies.* 2004 (Springer, Heidelberg).
[2] Nonaka I. and Takeuchi H., *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* 1995 (Oxford University Press, New York).
[3] Ehring, M., *Ontology Alignment: Bridging the Semantic Web.* 2006 (Springer, New York).
[4] Pahl G., Beitz W., Feldhusen J., Grothe K.H., *Engineering Design.* 2007, 3rd Edition (Springer, London).
[5] Gupta V.K., Govindarajan S. and Johnson T., Overview of Content Management Approaches and Strategies. In *Electronic Markets,* Volume 11, Issue 4, December 2001, pp 281-287.
[6] Beitz W. Konstruktionsleitsystem als Integrationshilfe. *VDI-Berichte 812: Rechnerunterstützte Produktentwicklung*, 1990 (VDI-Verlag, Düsseldorf).
[7] Heeren F. Der direkte Weg zum Wissen in den Köpfen. *Wissensmanagement*, 4/01, 2001.
[8] Alexander C., Ishikawa S and Silverstein M. *A Pattern Language*. 1977 (Oxford University Press, New York).
[9] Alexander C. *The Timeless Way of Building*. 1979 (Oxford University Press, New York).
[10] Alexander C., Silverstein M., Angel S., Ishikawa S. and Abrams D. *The Oregon Experiment*. 1975 (Oxford University Press, New York).
[11] Gamma E., Helm R., Johnson R. and Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1995 (Addison-Wesley, Reading, MA).
[12] Coplien J.O. and Schmidt C.C. (eds.) *Pattern Languages of Program Design*. 1995 (Addison-Wesley, Reading, Massachusetts).
[13] Vlissides J.M., Coplien J.O. and Kerth N.L. (eds.) *Pattern Languages of Program Design 2*. 1996 (Addison-Wesley, Reading, Massachusetts).
[14] Buschmann F., Meunier R., Rohnert H., Sommerlad P. and Stal M. *A System of Patterns: Pattern-Oriented Software Architecture*. 1996 (John Wiley & Sons, West Sussex).
[15] Borchers J. *A Pattern Approach to Interaction Design*. 2001 (Wiley, New York).
[16] Bayle E., Bellamy R., Casaday G., Erickson T., Fincher S., Grinter B., Gross B., Lehder D., Marmolin H., Moore B., Potts C., Skousen G. and Thomas J. Putting all Together: Towards a Pattern Language for Interaction Design. *SIGCHI-Bulletin*, 30(1):17-23, 1998.
[17] Tidwell J. Interaction Design Patterns. *PLoP'98 Conference on Pattern Languages of Programming*, 1998 (Illinois).
[18] http://www.pedagogicalpatterns.org [2006/12/05].
[19] Kempf A. *Entwicklung einer mechanischen Verbindungstechnik für Sandwichwerkstoffe*. 2004 (Dissertation, RWTH Aachen).
[20] Brezing, A., N. *Planung innovativer Produkte unter Nutzung von Design- und Ingenieursdienstleistungen*. 2006 (Schriftenreihe Produktentwicklung und Konstruktionsmethodik, 1, Dissertation, RWTH Aachen).
[21] Steffen, D. *Design als Produktsprache*. 2000, (Verlag form Theorie, Frankfurt).
[22] Saaksvuori A. and Immonen A, *Product Lifecycle Management*. 2004 (Springer, Berlin).
[23] Grieves M *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. 2006 (McGraw-Hill, New York).
[24] Stark J *Product Lifecycle Management: 21st Century Paradigm for Product Realisation*. 2005 (Springer, London).

Contact: Jörg Feldhusen, Frederik Bungert
RWTH Aachen University
Chair and Institute for Engineering Design IKT
Steinbachstraße 54 B
52074, Aachen
Germany
Phone: 49 241 8027341
Fax: 49 241 8022286
feldhusen@ikt.rwth-aachen.de, bungert@ikt.rwth-aachen.de
www.ikt.rwth-aachen.de.