INTERNATIONAL DESIGN CONFERENCE - DESIGN 2004
Dubrovnik, May 18 - 21, 2004.

DESIGN 2004

# AUTO-GENERATION OF DYNAMIC CROSS-LINKS AMONG MODULARIZED CONTENTS

S. Weiss and H. Birkhofer

*Keywords: modularization, content management, semantic net, design methods*

## 1. Introduction

Knowledge is an essential success factor [Stewart 1998] [Wiesenbauer 2001]. On the one hand, industrial product development profits from the knowledge of its employees and works out a potentially competitive advantage based on this. On the other hand, the academic sector is also faced with knowledge in to a certain extent (e.g. research and teaching).

The term ‘knowledge’ is discussed broadly in literature. Therefore, a working definition shall clarify the concept here. In this paper, knowledge has an explicit and transferring character and is mapped in specific, well-defined content structures. As regards the content, this must be assigned to the early phase of product development. Examples of relevant contents are definitions, design theory, methods, and cases of descriptions of solutions. This works intention is not to suggest a new approach in the field of knowledge representation. Rather it is our way to derive quickly high quality documents from existing content fragments and to publish them in the WWW. So in this paper, the term ‘knowledge’ stands for all contents that may be defined and memorized according to the modularization approach used (explained below).

This work refers to the ***pinn****gate* project. In the field of product development a broad range of knowledge and competences from "market to market" is covered and a variety of methods, processes, objects and structures are included. To manage this huge amount of knowledge in the range of product development the aim of the ***pinn****gate* project is to build up a holistic teaching, learning and training system (for details refer to www.pinngate.de). It is intended to provide unified, well structured and high quality contents and tools to support different users (e.g., students, product developers) in a goal-directed, flexible and individual way.

This paper presents an approach towards analyzing and (re-)structuring contents. Coherencies between content fragments are identified and published in the WWW automatically. In this approach, contents are saved electronically in a modular content base. This base is used as a content source to provide various application scenarios (industrial projects, teaching, learning). The high quality of the contents, as well as the scenario-optimized preparation, are of central importance. The basis for this is a modularization approach conceived in then *thekey*-project [Albers et al. 2001] [Birkhofer and Berger 2002] which was further enhanced by Berger and Birkhofer [Berger et al. 2002]. This formalized approach forces a clear structuring of the contents despite far-reaching flexibility. Thus, the high quality of the content is favoured. But there is also a constraint to work out and to coordinate different terminologies (e.g. various sources). Therefore, this approach does not only serve the memorization of contents. In the early phase of the work with modular contents, inconsistencies are identified and corrected. But content modularization leads to a variety of content fragements being able to be combined in diverse ways. This may have the result that the effort for linkage exceeds the qualitative added value of modular contents. Therefore, this paper shows one way to solve this problem. This approach automates the linking of content fragments algorithmically with very low effort of human activity. It is based on a given modularization approach.

## 2. State of the Art

A modularization approach takes objects apart, reduces duplicates and puts the rest together again in a well-structured way. Thus, the focus of modularization may border on diverse areas such as products or contents [Lehtonen et al. 2003], [Hölttä et al. 2003], [Berger et al. 2002], [Weiss et al. 2003]. Here, the term 'modularization' always stands for content modularization. In this context, the multiple use of parts plays a decisive role. Though, the same fact may be expressed differently in human languages. Duplicates are not directly recognizeable. Therefore, a sustainable approach must offer recommendations for overcoming these difficulties.

As a basis, the so-called 3-Level-Model is suitable [Birkhofer, Berger et al. 2002]. This model is formalized and has already been evaluated at the department *pmd* in a variety of application cases. Figure 1 shows the 3-Level-Model and the derivation of modules and containers:

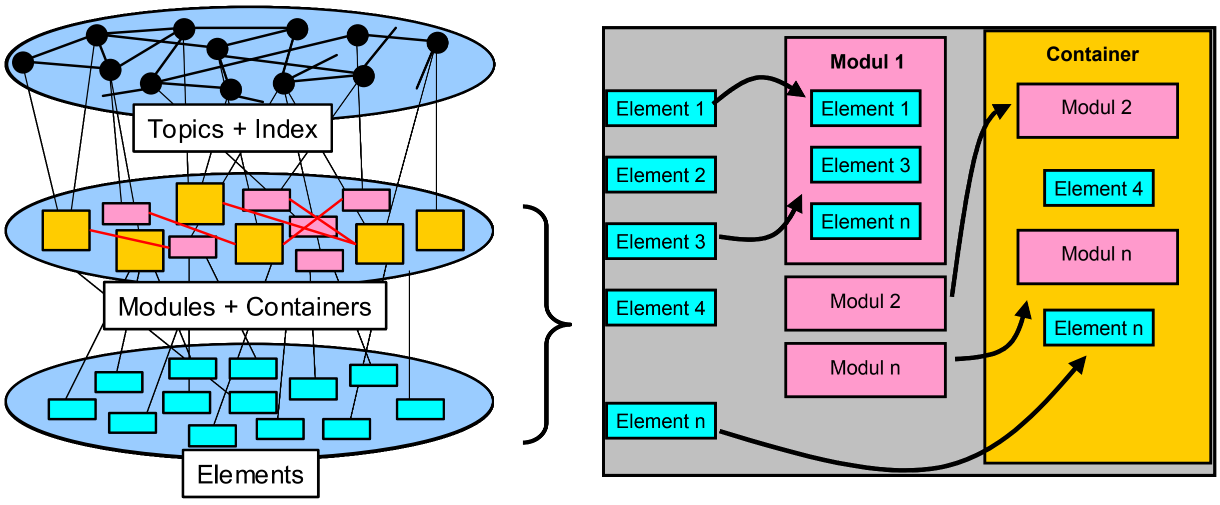


**Figure 1. 3-Level-Model and the derivation of modules and containers**

The lowest level is the element level. Elements form the base of this approach. They have no semantic character of their own. Examples of elements are text fragments, pictures, definitions, headlines and sketches. Two or more elements define a module. Modules have their own semantic character. Examples of modules are surveys, explanations, motivations and facts. Several modules constitute containers (also in combination with elements). Containers have a semantic character and are related to applications. Examples of containers are form-sheets, descriptions of design methods and chapters. The model requires attributes (meta-data) in defining **e**lements, **m**odules and **c**ontainers (EMC). These meta-data serve two different purposes: firstly, the classification and retrieval of elements, modules and containers and secondly, the allocation of process data for algorithms (context-sensitive knowledge extraction) [Weiss et al. 2003]. Examples of modularization meta-data are authors, topics and keywords. Here the process meta-data are ignored. The modularized contents are saved as independent objects in a database. This object contains all meta-data as well and is called 'knowledge-unit' [Weiss et al. 2003].
The contents saved in modules and containers are accessed via a semantic net. It is distributed across the module and container level. The concepts of the semantic net are linked with knowledge-units at the module and container level. This is done according to terms and content. Different ways how a user may be provided access to the stored contents are conceivable: a concrete search using keywords, system guidance (COSECO [Weiss et al. 2003]) or by using the semantic net intuitively.

This approach serves the creation of an intuitive access.

## 3. Deficits

This approach offers a series of advantages such as the constraint towards well-defined concepts, a clear structure, no duplicates and intra-structural dependencies. For example, a correction at the element level results in corrections in derived documents.

A serious weakness is the necessity to build up a semantic net for intuitive access. The effort concerning the definition of elements, modules and containers, as well as their occupation with meta-data, remains largely constant per knowledge-unit on average. On the other hand, the amount of links (between knowledge-units and nodes of the semantic net) to be set can increases exponentially as the number of knowledge-units and concepts increases. This overhead becomes very quickly complex until it can no longer be controlled. For this reason, it is necessary to refine the existing approach. The modularization model shall be used further. But the overhead concerning the construction of the semantic net and its links to modules and containers will be distinctly lowered. The approach described in this paper largely automates this process. Results may be published in the WWW automatically. Modifications are processed in real time. This is described in the following section.

## 4. Self Link Objects (SLO)

### 4.1 SLO basics

The basic idea behind **S**elf **L**ink **O**bjects (SLOs) is the extention of the modularization approach with regard to three points: an automatic (re-)structuring of knowledge-units, their automatic linking and the automatic derivation of a corresponding semantic net. Here, the knowledge-units contained in the content base form the basis. On the other hand, the definition of meta-data is necessary to determine on whose base an automatic linking may be executed. Moreover, specific algorithms must be developed. These algorithms process the meta-data, identify interdependencies among concepts, generate the links and publish the contents in the WWW. SLOs are derived from knowledge-units to provide contents (the real content), meta-data of the modularization (e.g. keywords) and meta-data of the automatic linking within one object. A Self Link Object, by definition, includes all data which are coupled to a content constituent (Figure 2).

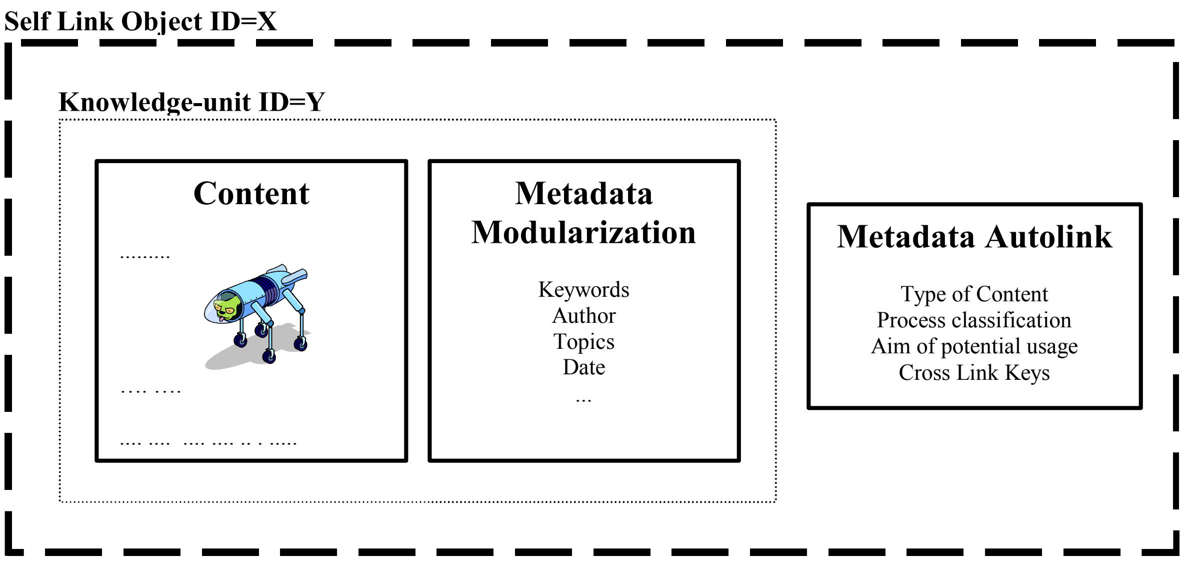


**Figure 2. Knowledge-unit and Self Link Object**

With the abbreviations KU (Knowledge-Unit), C (content), MM (Metadata Modularization), MA (Metadata Autolink) and SLO (Self Link Object) one can note:

$$KU = C \cup MM$$
$$SLO = KU \cup MA = C \cup MM \cup MA$$

The extension of the meta-data concerning the enabling of an automatic linking is carried out pragmatically in this approach. For this, four attribute classes were defined. These are frequently helpful and classify knowledge-units with respect to the kind and potential use of content. The classes are *type of content* (definition, theoretical background, example, general overview, method overview, solution overview, form-sheet, literature), *aim of potential use* (application, teaching, learning), *process classification according to VDI 2221* (clarify task, define functions, …) and *Cross Link Keys* (CLK). The Cross Links Keys of a Self Link Object A cause a linking between SLO A and another SLO B if Cross Link Keys of SLO A occur in contents of SLO B and vice versa. At this, all links are created by the gradual comparison of all Self Link Objects. Instead of linking an object under construction to already existing objects, CLK are defined. These may cause linkages to other objects later.

## 4.2 Cross Link Algorithm

The information to be published in the WWW is generated using the object "Self Link Engine". This object works based on Self Link Objects. So the prerequisite to run the Self Link Engine is the definition of Self Link Objects. First, the Self Link Engine creates a list of all CLK of all Self Link Objects in the database ($CLK_{total}$). CLK are suggested automatically by the system when objects are created. In addition, all technical terms of a knowledge-unit are identified. An algorithm examines the objects in question and isolates all words, reduces this set around items, pronouns, prepositions and other fragments. Fragments from the so-called Not-List are stricken from the remaining set. The Not-List is a list of words which will not serve as CLK. Cross Link Keys can also be defined manually. Suggestions by the system can be revised as well. The remaining list is extended by alternative concepts which are defined in a configurable thesaurus (Figure 3+4). If duplicates arise, these are rationalized away ($_{CLKtotal,\,T}$).

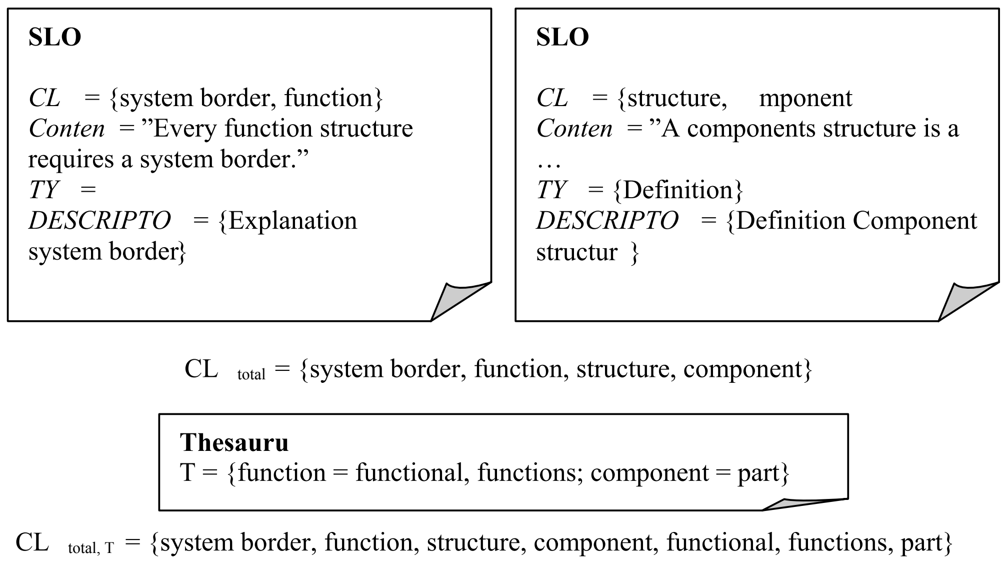

SLO
CL = {system border, function}
Conten = "Every function structure requires a system border."
TY =
DESCRIPTO = {Explanation system border}
SLO
CL = {structure, mponent
Conten = "A components structure is a …
TY = {Definition}
DESCRIPTO = {Definition Component structur }
CL total = {system border, function, structure, component}
Thesauru
T = {function = functional, functions; component = part}
CL total, T = {system border, function, structure, component, functional, functions, part}

**Figure 3. Generation of the Cross Link list**

The contents of all Self Link Objects are checked for CLK like these. A list is led to every single CLK. In this list the ID of Self Link Objects is registered if the respective CLK occurs in the contents of the

SLO. Moreover, the type of the SLO in question is filed in this list too. Furthermore, a so-called descriptor exists to outwardly present Self Link Objects.

| i | CLK | I | TY |
|---|---|---|---|
| 1 | syste border | SLO A | explanatio |
| 2 | function | SLO A | explanatio |
| 3 | structur | SLO A, SLO | explanatio ,definition |
| 4 | componen | SLO | n definitio |
| 5 | functional | SLO A | explanatio |
| 6 | functions | SLO A | nexplanatio |
| 7 | part | SLO | ndefinitio |

**Figure 4. Generation of Cross Link Data (before sorting)**

The generated list is organized by sorting according to type. From these data (Cross Link Data, CLD) Cross Link Files are generated. These serve as the basis for the production of the web code. Depending on the requirements, the Self Link Engine produces code in HTML, Dynamic HTML and Javascript. The coding is carried out on the basis of model files and rules which are taken by corresponding algorithms to harmonize with the CLD. Particularly, the contents of every SLO are analyzed word-by-word. Based on CLD links to identified CLK are set. Assuming one views SLO A of figure 3 or 4 in the web mode: By selecting the concept “structure” the consideration of the definition B is suggested.

Overviews are created after this whole process. These are: glossary, example collection, general overview, method overview, solution overview and literature collection. These can be generated very simply by processing the meta-data: every element of the type “definition” belongs to the glossary and every module of the type “example” belongs to the example overview.

## 4.3 Self Linker

The complete modularization environment is known as Self Linker. It is concept and software developed by the authors. A short explanation shall point out the main functions:
It serves the filling out, editing and configuration of the modular content base. The Self Linker manages the Self Link Objects. The modular database is the central unit of the Self Linker. Contents must be entered in the database so that elements, modules and containers can be derived. This is the task of the Base Document Editor (BDE). The BDE generates and imports (.doc, .pdf, .txt, .jpg, .html) Base Documents (BD) and sends them to the database. These BD are loaded into the Constructor from the database. The Constructor provides the generation of elements, modules and containers. So one can use existing parts of documents to arrange new ones. Base documents contain the real content, have no EMC-status and carry no meta-data. Base documents also can be provided with different formats in the constructor. Formats of elements can be overwritten at a higher level (modules and containers). The fundamental structure of a module or container is defined in the Constructor Code (CON). The Constructor Code does not contain the absolute contents (the real content) but merely the ID of the single elements, modules and containers, their sequence as well as formats. The Constructor Code is tagged with attributes (meta-data). Constructor Code plus meta-data (contents plus meta-data) form a SLO to be saved in the database. In this way, no absolute data get stored but only dependencies.

Self Link Objects can theoretically be eliminated by deleting necessary constituents. Therefore, all structures are checked before deleting objects.
The attributes of the SLO are configured by an editor. There are editors to define topics, keywords, authors, CLK and the thesaurus. Standard configurations can be stored or modified in the database,

e.g. authors or keywords. The real configuration of the SLO happens with the attribute editor. This editor uses standard configurations and allows modifications at run time.

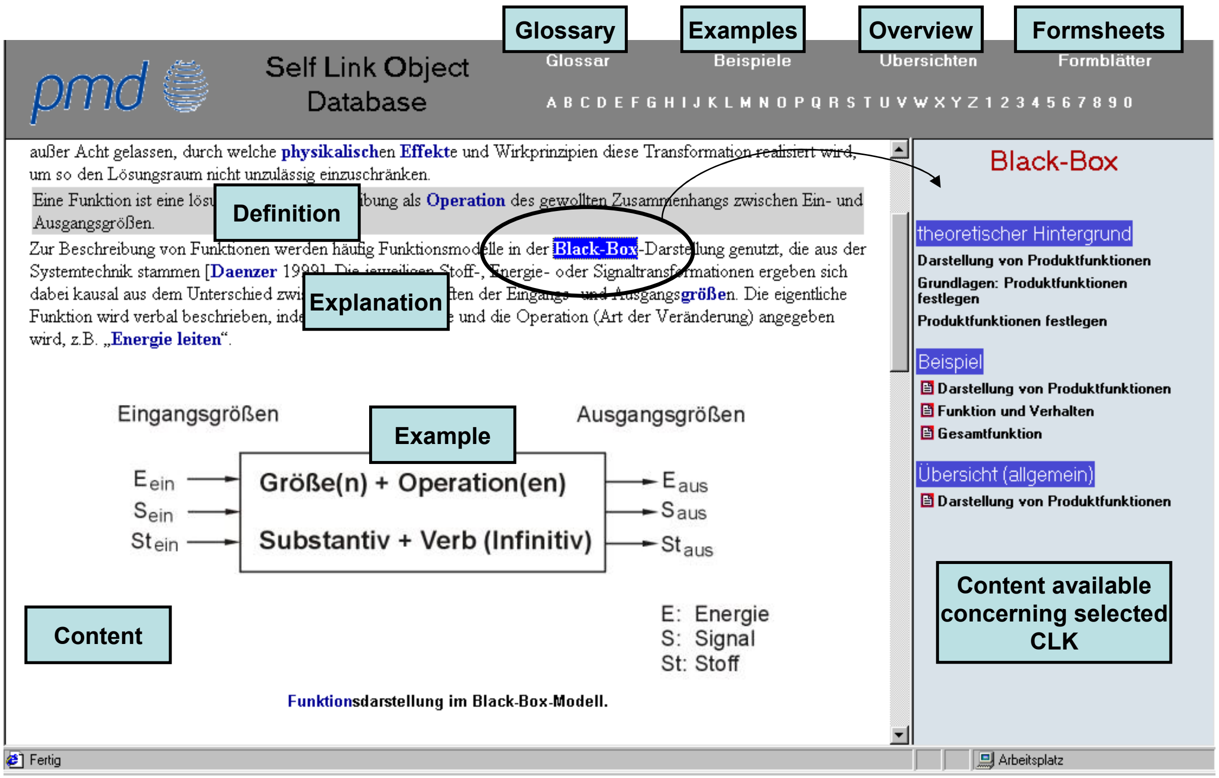
During the work, the so-called Copy List has proved to be practical. The Copy List buffers BD and CON. Thus, one has an overview and high-speed access to interesting or frequently used constituents. Generally, a preview of any object is possible at any time.

The Search Object provides an extensive search function across the whole database. The Search Object provides both arbitrary attribute combinations and full text. Diverse options may be used.

Besides the functions mentioned, a generator for PoMM (Process-orientated Method Model [Birkhofer, Kloberdanz et al. 2001]) is provided. The PoMM allows the description of methods especially for the application. The single areas of the PoMM can be defined over modules and containers by an editor. From this, a PoMM supported graphic is edited automatically for publication in the WWW. Arbitrarily, many PoMM can be defined and managed.

## 4.4 Frontend

The main object is the Self Link Engine which puts all defined SLO in relation and configures a web server in real time.The front end created by the Self Link Engine is shown in figure 5. Actually, the front end and its menus are provided in german language only. So the example is given in german langue too (sections are marked in english terms).


außer Acht gelassen, durch welche physikalischen Effekte und Wirkprinzipien diese Transformation realisiert wird, um so den Lösungsraum nicht unzulässig einzuschränken.

Eine Funktion ist eine lösu ibung als Operation des gewollten Zusammenhangs zwischen Ein- und Ausgangsgrößen.

Zur Beschreibung von Funktionen werden häufig Funktionsmodelle in der Black-Box-Darstellung genutzt, die aus der Systemtechnik stammen [Daenzer 1999]. Die jeweiligen Stoff-, Energie- oder Signaltransformationen ergeben sich dabei kausal aus dem Unterschied zwi ten der Eingangs- und Ausgangsgrößen. Die eigentliche Funktion wird verbal beschrieben, inde e und die Operation (Art der Veränderung) angegeben wird, z.B. „Energie leiten".

Funktionsdarstellung im Black-Box-Modell.

Glossary
Examples
Overview
Formsheets
pmd
Self Link Object Database
Glossar
Beispiele
Übersichten
Formblätter
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 1 2 3 4 5 6 7 8 9 0
Definition
Explanation
Example
Content
Eingangsgrößen
Ausgangsgrößen
$E_{ein}$
$S_{ein}$
$St_{ein}$
Größe(n) + Operation(en)
Substantiv + Verb (Infinitiv)
$E_{aus}$
$S_{aus}$
$St_{aus}$
E: Energie
S: Signal
St: Stoff
Black-Box
theoretischer Hintergrund
Darstellung von Produktfunktionen
Grundlagen: Produktfunktionen festlegen
Produktfunktionen festlegen
Beispiel
Darstellung von Produktfunktionen
Funktion und Verhalten
Gesamtfunktion
Übersicht (allgemein)
Darstellung von Produktfunktionen
Content available concerning selected CLK
Fertig
Arbeitsplatz

**Figure 5. Frontend of the Self Link Object database web portal**

The modular contents are represented in the large window. Highlighted CLK are available in some areas. An algorithm represents every CLK per display only once. By clicking CLK (here Black-Box) a menu appears at the right side. It contains all options enabled regarding the selected CLK in a well-structured way. At this the structure is based on the contents type. One has general navigation options like glossary, examples or overviews at the disposal in the upper area. All suggested objects refer to the selected CLK.

Figure 6 shows the linkage of the various sections of the PoMM on modules and containers. These links are transformed by the Self Link Engine into a semantic net. If one selects a PoMM section in the web mode, one reaches the contents deposited.

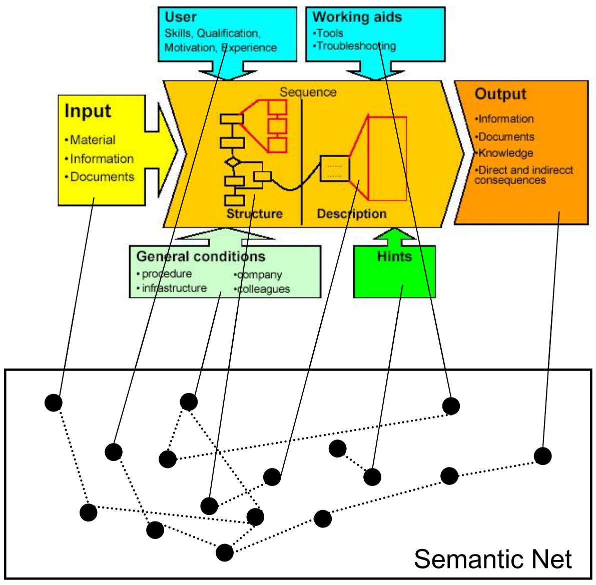


**Figure 6. PoMM links modules and containers in sematic net**

## 5. Summary and Outlook

The concept modularization is the core of current research and has several occurences. One of these occurences is the modularization of contents. This paper is based on a modularization approach and sets the focus on the early phase of product development. Despite spacious load-capacity of this approach there is a central problem: the potential number of links between the real contents and the semantic net. A proposal for a solution is suggested in this paper: The definition of Self Link Objects . These carry both the real contents and meta-data which meets the requirements of the modularization attempt and the requirements of an automatic construction of the semantic net. On this basis, a web-based portal which publishes contents modularly filed is generated automatically. This approach is sustainable and suitable. This has been proved by various applications so far. Currently, this approach hardly takes international standards into account. Therefore, this approach is suitable only for application cases in which new contents are created.

### References

*Albers, A., Birkhofer, H., Lindemann, U., Meier, M., "Product Development As a Structured and Interactive Network of Knowledge", in: Proceedings of the International Conference on Engineering Design, ICED 2001, Bd. 3, Glasgow, 2001.*

*Berger,B., Birkhofer, H., „Modularization of design knowledge as a basis for a high quality competence pool in product development", in: Proceedings of DETC 2002, Hrsg., ASME 2002 Design Engineering Technical Conferences, Montreal, 2002.*

*Birkhofer, H., Berger, B., „Thekey to innovation – modularisation of design knowledge as a basis for a scientific approach to design", International Conference The Sciences of Design, Lyon, 2002.*

*Birkhofer, H., Berger, B., Walter, S., „Modularisation of Knowledge – A New Approach in the Field of Product Innovation", International Design Conference – Design 2002, Dubrovnik, 2002.*

*Birkhofer, H., Kloberdanz, H., Berger, B., Sauer, T., „Cleaning Up Design Methods – Describing Methods Completely and Standardised", International Design Conference – Design 2002, Dubrovnik 2002.*

*Birkhofer, H., Lindemann, U., Albers, A., Meier, M., "Product development as a structured and interactive network of knowledge – a revolutionary approach", ICED 2001, Glasgow UK, WDK 28, Vol. 4., pp. 457-464.*

*Hölttä, K., Tang, V., Seering, W. P., "Modularizing Product Architectures Using Dendrograms", Proceedings of the International Conference on Engineering Design, ICED 2003, Stockholm, Sweden.*

*Lehtonen, T., Juuti, T., Pulkkinen, A., Riitahuhta, A., „Dynamic Modularisation – A Challenge For Design Process and Product Architecture", Proceedings of the International Conference on Engineering Design, ICED 2003, Stockholm, Sweden.*

*Stewart, T. A., "Der vierte Produktionsfaktor", Carl Hanser Verlag, München, Wien, 1998.*

*VDI-Richtlinie 2221, "Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte", Düsseldorf, VDI-Verlag, 1993.*

*Weiss, S., Berger, B., Jänsch, J., Birkhofer, H., „COSECO (Context-Sensitive-Connector) – A Logical Component For a User- and Usage-Related Dosage of Knowledge", Proceedings of the International Conference on Engineering Design, ICED 2003, Stockholm, Sweden.*

*Wiesenbauer, L., "Erfolgsfaktor Wissen", Beltz-Verlag, Weinheim u.a., 2001.*

Dipl.-Wirtsch.-Ing. Sascha Weiß
Departement of product development and machine elements
Darmstadt University of Technology
Magdalenenstr. 4, 64289 Darmstadt, Germany
Telephone: +49 6151 16 2660, Telefax: +49 6151 16 3355
E-mail: weiss@pmd.tu-darmstadt.de