Evolving KobrA to Support SPL for WebGIS Development

Juan Manuel Moreno-Rivera[1,2], Elena Navarro[3], and Carlos E. Cuesta[4]

[1] Earth Observation and GIS Department, University of Castilla-La Mancha, Albacete, Spain
[2] SIGTel Geomática, Centro I+D de Empresas – PCYTA, Albacete, Spain
JuanManuel.Moreno@sigtel.es
[3] Computing Systems Department, University of Castilla-La Mancha, Albacete, Spain
Elena.Navarro@uclm.es
[4] DLSI II, Rey Juan Carlos University, Spain
carlos.cuesta@urjc.es

Abstract. SIGTel is a SME aimed at developing WebGIS application. It has a portfolio of products which are a combination of Open GIS components and its own developed components. Although most of these products share a common architecture and common features, they have to be customized according to the user requirements. Now, this SME is working at the improvement of the software development process by introducing a Software Product Line (SPL) approach to automate the process of development and deployment based on that common architecture, as well as reduce time to market and improve quality. The first step has been taken, and KobrA approach has been chosen as the basic methodology to start working. Now, the on-going work is to perform the Domain Engineering, being the management of variability one of the first issues to be resolved. Here, we present a combination of KobrA containment tree and Orthogonal Variation Model (OVM), as the selected alternative to model the variability by using the architecture of our WebGIS SPL.

Keywords: KobrA, SPL, WebGIS, component-based development, open GIS.

1 Introduction

The complexity of software development is increasing nowadays, and the competition between enterprises is harder than ever. This situation becomes worst when we get into the context of Small and Medium Enterprises (SME). In this context, SIGTel Geomática [18] portfolio is based on the production chain provided by the Open GIS community, where open source components are combined with their own ones. It has been detected that a high percentage of the development of new products is repeated along the whole production chain and that the 90% of software development projects share a common architecture. This philosophy fits perfectly with the approach of Software Product Lines (SPL). Considering that situation, an SPL-based approach has been chosen, intending to help us to automate the software development process of WebGIS applications [11] as far as possible, as well as managing variability and traceability.

After this first step, an evaluation of existing approaches and tools has been done, deciding on the KobrA approach [12] as the final choice. KobrA was selected to guide the development of the SIGTeL SPL due to several reasons. First, the KobrA philosophy suits almost perfectly to the WebGIS development process because it is a component-based approach, and WebGIS development focuses on components as their basic unit of development. Thus, a component-based approach, as KobrA [3], makes the models easier to understand to WebGIS developers. This approach also takes into account third party components, in the form of plug-ins within the model.

An improvement to the KobrA containment tree is presented here, analyzing which are the aspects of KobrA that needs to be customized in order to be applied in the WebGIS domain. In addition, we present a new approach using KobrA tools combined with Orthogonal Variability Model (OVM) [13] notation. Our proposal reduces the amount of work to be done when solving variability and makes more easy application engineer work. We also introduce a prototype tool developed to support the work presented here.

In this work, Section 2 presents the domain and the basic concepts. Section 3 describes the related works. Then, in Section 4, the application of KobrA methodology within our domain is analyzed by applying it to a case study, the core components of the SPIDER system [10], in order to establish the advantages and disadvantages of this approach within a real world project. Next, after identifying which are the improvements and/or adaptations that this approach needs in order to be applied in the context of WebGIS applications, our proposals is described in Section 5: the adaptations of the KobrA containment tree to improve the variability management and Section 6 presents the case study. Finally, in Section 7, the next steps of our Software Process Improvement (SPI) roadmap, based on the results here presented and the expectations of SIGTel, are described.

2 WebGIS Domain

WebGIS is the acronym for Web-based Geographical Information System. WebGIS's central feature is a map along with a system to analyze, edit and produce spatial information. These systems are becoming more popular every day. Nowadays, just about everybody uses WebGIS, Google Maps© [5] being the most popular example.

Market changes have led to an explosion of the demand with a lot of large and small companies fighting for its place in this emerging market. New users are becoming more familiar with WebGIS and are requesting new features. Complex desktop processes are being adapted to a web-based system as new technologies become available. Besides the increasing complexity of WebGIS, the Open GIS community is growing and offering more possibilities for developing new products. Software development companies are now expected to develop new WebGIS systems in a shorter time with better features and improved quality.

This context makes *reuse* into an even more difficult task because each component needs to be adapted and be more flexible for demands of new features. These difficulties are more critical for small companies, such as SIGTel, that not only have to maintain the production levels but also keep up to date with new market trends. However, the development time of final products is too long and the quality is often not what users expect because of the inherent complexity for testing WebGIS applications. This complexity remains in spite of the reuse of existing components, the development based on the adaptation of existing systems, and the use of widely used communications standards defined by the Open Geospatial Consortium (OGC) [15]. Another problem of WebGIS is that as systems increase their complexity, it becomes a hard task to manage the whole architecture in an efficient and productive way. This problem becomes worst when the phenomenon of *cloning* [4] emerges making more difficult the maintenance of deployed systems and the reuse of new components for specific customers. Finally, this complexity is also related to the difficulties for implementing test benchmarks for systems with so many interconnected components from different providers. Therefore, it is necessary to include some methodological and systematic approach for producing, deploying and maintaining this kind of systems that enable to share a common architecture and be able to manage the variability in an efficient and a more productive way.

3 Related Works

In WebGIS development there is a concept called *GeoStack* that also uses, similarly to SPL, a common architecture along with a family of products. Many enterprises and research groups work around a framework or combination of components that facilitates the adaptation based on customers' requirements of a limited set of optional features for their final products. The problems of this approach are three: (i) it is strongly linked to the technology these components have been developed for; (ii) the variability is constrained to a reduced group of final features included in the systems; (iii) the cloning phenomenon arises and due to the lack of traceability, each component needs independent maintenance efforts. Nowadays SIGTel's GeoStack works like that. The ideal solution would rely on managing the variability in a way that facilitates the assembly of different combinations of components according to specific customers' requirements. For example, if our family of products is based on Flash technologies and a new customer asks for a product without Flash, due to any kind of security or corporate decision, could we provide a solution based on our common architecture that supplies the same features but using non-Flash components? Although we could answer positively this question, the time needed to adapt this architecture to the customer's needs would too long and would increase the final cost of the product, as well as the time to market. Therefore, our company would not be competitive for managing this kind of situations. This situation led us to use SPL in SIGTEL.

SPL approaches can be classified into two different categories: *feature-based* and *component-based* approaches. Feature-based approaches use features as the basic unit and component-based ones focus on the component level. The first ones are more mature and have been used for the last twenty years. FODA [8] settled down the bases for this family of approach in the 90's. However, in WebGIS domain component is the basic unit of development, so component-based approaches, like KobrA or CoPaM [1], fit better to the domain of WebGIS.

No matter which approach is followed, all of them aim at providing suitable techniques to *manage variability*. KobrA, for example, uses UML stereotypes to manage variability. It works but it still needs improvements as explained in Section 5. There are notations, like Common Variability Language (CVL) [16] or OVM [13] that have been considered to improve variability management in KobrA. Both of them have an Eclipse plug-in available, Eclipse CVL plug-in [17] for CVL and VarMod-Editor [20] for OVM, which makes them more suitable to our needs, as this is the IDE used in SIGTel. CVL is mainly used to define feature diagrams. OVM is a standard notation focused on supporting variability in SPL for improving the testing and formal validation of models. This is the reason why the latter has been finally selected, as the proper definition of the testing process is one of the main problems to be resolved in the near future of SIGTEL.

4 KobrA in a Nutshell

KobrA [3] is a model-driven approach for component-based development that can be used as well for deploying SPLs. It is based on the concept that a system, and any element that makes it up, can be modelled as a component. This approach establishes a model that represents the interaction with the external environment and details the internal design of each of the components in a tree hierarchy. The KobrA modelling process of a component entails two different tasks: the *specification* of how a component interacts with the rest of the system and what does; and the *realization* of how the component performs this specification by means of its design. Therefore, KobrA models components in a similar way to WebGIS development approaches because these two different views resembles the way components are assembled in a WebGIS application. In WebGIS, components are connected through their APIs or by standard communication channels with other components or external elements and then are customized according to customer needs.

KobrA uses two tree structures: the *containment tree,* to specify the relationships between components at development time and the *composition tree,* to specify a run-time instance of an existing system. The main idea is that both of them are aligned so that it is easy to define a relationship between the design and the runtime execution of new systems. Considering these two notions of composition, KobrA allows us to model both the design and the execution of our systems, which can be very useful in order to define a testing approach for this kind of systems. The containment tree is used during the *domain engineering phase* to represent the structure of the system, that is, its design and the components distribution. This tree is instantiated during the *application engineering phase* to deploy different products by resolving the existing variation points. Afterwards, each instantiated tree is refined and transformed into another tree structure to represent the logical and physical distribution of the specific product following the *KobrA flattening process*, and its modelled components are replaced by code components applying the *KobrA translation process*.

During the application engineering phase, the engineer goes throughout the containment tree solving the variability starting from the top node, which represents the whole system, to the leaf component nodes, till all the variability is solved for each specific product. The first consequence of this approach is that there is more redundancy and verbosity than necessary. due to the *localized view* of each component. The reason is that KobrA establishes a set of models for each component, to ensure its independent and homogeneous treatment. However, the number of models required for modelling components and the verbosity along the tree, turns both variability resolution and models maintenance into two error-prone and tedious tasks. In addition, the decision models used to resolve variability are text-based, and this makes the process too complicated, even with tool support. For this reason, it would be worthwhile to provide engineers with a means to deal with the variability in a more agile way during the application engineering phase. In addition, reducing the number of models and diagrams will also reduce the efforts needed to maintain and update the framework of our WebGIS SPL.

In addition, there is a lack of automation in KobrA, especially traumatic at the implementation phase as there are no defined methods or processes to transform design models to implementation models. Finally, there is no tools support for KobrA approach. By the time this work has been written, the KobrA developers are working at the development of a supporting tool for KobrA approach. However, this tool is not available so that it was not possible to test and validate it. For this reason, we have had to develop our own prototype, based on GMF [6], to support the modelling of KobrA compliant components.

5 Evolved KobrA Containment Tree for Supporting WebGIS Domain Engineering

KobrA approach fits perfectly with the way WebGIS systems are designed. But it is a general purpose approach and some customization needs to be applied to improve the performance of this approach. For example, as pointed out in [3], KobrA context realization, a.k.a. domain engineering, is focused on Enterprise Resource Planning (ERP) domains. KobrA uses package containment of UML to model the architecture and the hierarchy of components of a system. It also models the relation, run-time and design dependencies, between components from different branches of the tree but in this components tree no variability is modelled. As mentioned before, this means that application engineers need to go along the containment tree, node by node, resolving all the decision modelled for each node. By locally managing variability, as KobrA does, engineer loses the global view of the system offered by the containment tree. Apart from that, variability dependencies are only managed through the tree hierarchy from the components tree which is not enough to manage horizontal variability dependencies. More than that, decision models are text-base, which is not very helpful during application engineering process. Finally, variability is only used to manage the inclusion of features, but there is no variability management during implementation phases which offer an opportunity of choosing between different implementations of components, to provide a more adaptable architecture for SIGTel's SPL.

In the WebGIS domain, *variation points* emerge at the components level, while designing new instances of our system, because it is at this moment when engineers decide which of the existing physical components are chosen to implement selected features. For third party components, getting inside each component node is not necessary because most of them are integrated in members of our family of products as they are provided. In fact, in the WebGIS domain there is an initial variability decision process that focuses on deciding which components are going to be included to implement features of new systems taking into account the user requirements. Therefore, the enhanced containment tree presented here should allow engineers to navigate throughout the system in a simplified and clear way, identifying in every single moment how components and variation points are connected and their dependencies during the application engineering phase. It would also facilitate reducing the number of models necessary for each component, so that the interaction with external components would be modelled in this enhanced containment tree, and each node would focus on the realization of its elements. Also, third party components would not need, in most cases, an internal definition because they would be used as they are provided by just considering their APIs.

The KobrA notation for variability uses the tag <<variant>> together with components and attributes to model each node of the containment tree by means of class diagrams. We propose an enhancement to improve the KobrA notation and exploit the benefits of the global overview of the system provided by the containment tree. This enhancement will allow engineers to interact with it while maintaining the SPL reference architecture and while deriving new products during the application engineering. The chosen notation to develop this enhancement is based on OVM.

In the context of WebGIS, a containment tree can become very huge so that if an application engineer wants to instantiate this model following KobrA guidelines, he/she has to go throughout the components hierarchy. This can be a very hard and tedious task even by using a modelling tool or a model-driven process. Here one of the problems of the component-based approaches arises, as described in [12]. By improving the variability notation of the containment tree, the fast prototyping skills of the SPL approaches based on KobrA are promoted. In [3], it is established that ”there is no reason to support variability anywhere in a composition tree unless they have an effect on the properties of the overall system represented by the root“. The idea is that KobrA manages the variability of the whole system at top of the containment tree. However, with the enhancement proposed, the work to be done

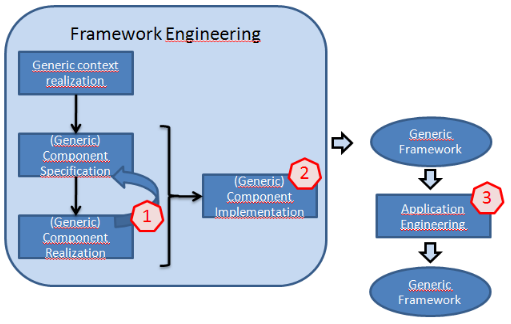
Framework Engineering
Generic context realization
(Generic) Component Specification
(Generic) Component Realization
(Generic) Component Implementation
1
2
Generic Framework
Application Engineering
3
Generic Framework

Fig. 1. Product Line life-cycle according to the KobrA Approach [3]

during the creation, maintenance and update of the SPL framework can be reduced (see point 1 at Figure 1). In addition, this enhancement facilitates the application engineering tasks by making more agile the flatting and translation process of the KobrA approach by using OVM notation (see point 3 at Figure 1). By considering the variability in the containment tree, the adaptability of our SPL can be improved. As our WebGIS SPL is based on Open GIS components, it is necessary to manage not only which features should be included or not, but also which component should be used for implementation (see point 2 at Figure 1). These decisions can be based on user requirements, expertise of the engineer or even changes in the community of any of the third party components. We need to make our SPL more adaptable so that it does not depend on a concrete component and its implementation, because the open software community is one of our providers, and therefore our SPL should not depend on any specific component.

6 Case Study

As mentioned before, our case study is SPIDER [10]. Here, a reduced version of this system is presented focusing on its core elements. It is a good example of the architecture shared by many WebGIS projects developed in SIGTel, but at the same time it represents one of the more complex systems that can be developed. This can be considered a good example because even only with the core components, it is complex enough to be considered a real example.

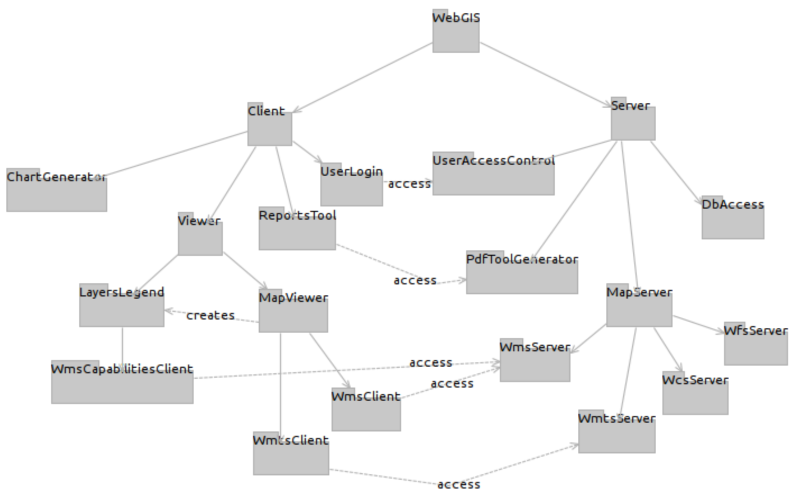

Fig. 2. SPIDER core components architecture with KobrA

Figure 2 illustrates a containment tree of the core components of SPIDER architecture modelled with the previous version of KobrA components tree. It shows in a glance the whole system architecture. As can be observed, it is not possible to identify the relationships of variability (following the KobrA approach) without getting into each component, where UML diagrams model the *localized* variability for each node of the tree. Furthermore, KobrA relationships between components do not consider the <<excludes>> relationship, which is critical for managing variability, as far as this relationship is basic for variability management. One component removal from the system can make other components incompatible with the instantiated version of the architecture. The only way it can be specified is by including in the decision model of each node which components or features have also to be removed. However, this is a difficult task as these decision models are text-based, making a difficult task its inclusion. This is another reason to include OVM because this notation considers this relationship. As can be observed in Figure 3, it facilitates to easily identify the Variation Points existing in your containment tree, what is not the case for the previous KobrA version. Finally, one important contribution of using OVM is that it does not only also consider relationships between components, but also between Variation Points. There is no notation for variability in such an advanced way in the KobrA basic approach. Thus, an improved version of KobrA offers a most suitable philosophy SPL approach for WebGIS, but improving its usability for engineers in a real context as in SIGTel.

Figure 3 shows how now the Variation points "ChartGenerator" and "ReportTool" are linked, so that by solving the variability of one of them implies to solve the variability of the other, depending on the option chosen. At the same time, we not only specify the variability of the components and features to be included, but also of implementation alternatives using different components. In addition, the application engineer can solve the variability of logical functionality and part of the flatting process at the same time by means of this model. For example, the MapViewer now is a variation point and we can start deciding on features and code component in the same process, reducing part of the required process in KobrA. Also, using this approach, it is not necessary to describe detailed models of the third party components, as the engineer is only interested in the dependencies they have and the basic configuration for their deployment.

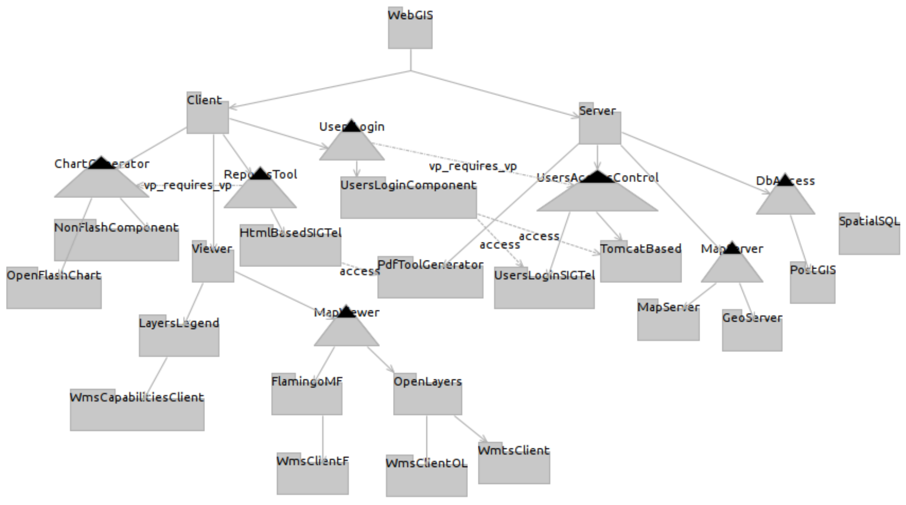

Fig. 3. SPIDER core components architecture with KobrA + OVM

6.1 Tool Support

As pointed out in [2], there is no tool support to ensure that the models are compliant with KobrA. The existing UML modelling tools cannot be used to model any system based on the KobrA approach. The group that created KobrA is working on the development of a tool to support this approach, however it is not available and just some examples of models done with this tool can be found in [2]. Therefore, SIGTel had to start working on the development of a plug-in to support the modelling tasks that were performed during this project, in order to avoid doing all the models manually, which is very tedious and time consuming.

The tools that have been considered to support the development of this work are Eclipse-based tools / frameworks, like MOSKitt [14] or VarMod-Editor. TOPCASED [19] was also considered as it offers an interesting framework for critical and embedded systems, but for this work GMF [6] was used to develop the prototype tool for this work, as it is the main tool to develop graphical modelling tools in Eclipse.

7 Conclusions and Future Work

After evaluating KobrA by means of our case study, we decided to improve its variability management in order to improve its potential benefit for WebGIS SPL development, as well as to offer support for a more adaptable SPL. Combining KobrA containment tree with OVM makes the process clearer and the models more maintainable and reusable, giving an overview of the whole architecture variability in a glance. By adding the notation for managing variability to the containment tree, the agility to instantiate variability from the very beginning is improved.

In addition, the top node component of the containment tree provides us with a decision model to make decisions about general user requirements. This decision model is used as proposed by KobrA, using the new containment tree to solve the logical and physical configuration of each new product during the application engineering phase. In this way, developers are able to generate new products by using the enhanced containment tree as the graphical interface to interact with the SPL framework. Moreover, domain engineers have the opportunity of dealing directly with the containment tree to update and maintain variability in the SPL framework in an agile way, making the work done during the domain engineering phase significantly more profitable.

As future work, we consider that it is necessary to work with the variability for each of our own developed components the KobrA way. Also, it is important to define how to deal with the open GIS components modified by SIGTel and how to integrate them in our architecture. In the future, our plans include to study how to use and exploit model-to-text transformations existing in Eclipse (M2T, [9]) in order to automate of translation process (i.e. transformation from logical components to code components). The first steps towards the SIGTel's SPL have been taken, but there is still a lot of hard work to be done.

Acknowledgments. This work has been partially sponsored by SIGTel Geomática, the EOGIS group (UCLM) and the Spanish Ministry of Science and Innovation through the Torres Quevedo Program and the DESACO (TIN2008-06596-C02-01) and AT (CONSOLIDER CSD2007-0022) Projects.

References

1. America, P., Obbink, H., Ommering, R., Linden, F.: CoPaM: A Component-Oriented Platform Architecting Method Family for Product Family Engineering. SPLC (2000)
2. Atkinson, C., Bostan, P., Brenner, D., Falcone, G., Gutheil, M., Hummel, O., Juhasz, M., Stoll, D.: Modeling Components and Component-Based Systems in KobrA. In: Rausch, A., Reussner, R., Mirandola, R., Plášil, F., et al. (eds.) The Common Component Modeling Example. LNCS, vol. 5153, pp. 54–84. Springer, Heidelberg (2008)
3. Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., et al.: Component-based Product Line Engineering with UML, 1st edn. Addison Wesley (2001)
4. Dikel, D., Kane, D., Ornburn, S., Loftus, W., Wilson, J.: Applying software product-line architecture. Computer 30, 49–55 (1997)
5. Google, Google Maps (n.d.), http://maps.google.com/
6. Graphical Modeling Framework, http://www.eclipse.org/modeling/gmp/?project=gmf-tooling#gmf-tooling
7. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Anaheim, California, USA (2003)
8. Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. SW Eng. Institute (1990)
9. Model to Text (M2T) project, http://www.eclipse.org/modeling/m2t/ (last access July 08, 2011)
10. Moreno-Rivera, J.M., Osann, A., Calera, A.: SPIDER - An Open GIS application use case. In: First Open Source GIS UK Conference, Nottingham (2009)
11. Moreno-Rivera, J.M., Navarro, E.M.: Improving Component-based WebGIS Development by Adopting a Software Product Line approach. In: Information Systems Development, Praga, República Checa (2010)
12. Moreno-Rivera, J.M., Navarro, E.: Evaluation of SPL approaches for WebGIS development: SIGTel, a case study. In: 44th Hawaii Int. Conf. on Systems Science (HICSS-44 2011), January 4-7, pp. 1–10. IEEE Computer Society, Koloa (2011)
13. Pohl, K., Böckle, G., Linden, F.J.V.D.: Software Product Line Engineering: Foundations, Principles and Techniques, 1st edn. Springer, Heidelberg (2005)
14. MOSKitt, Home Page (n.d.), http://www.moskitt.org/eng/moskitt/
15. OGC Consortium, OpenGIS Implementation Specification for Geographic information - Simple feature access (n.d.)
16. OMG Common Variability Language (CLV), http://www.omgwiki.org/variability/doku.php (last access July 05, 2011)
17. OMG CVL Tool from SINTEF, http://www.omgwiki.org/variability/doku.php?id=cvl_tool_from_sintef (last access July 08, 2011)
18. SIGTel Geomática, http://www.sigtel.es/ (last access June 27, 2011)
19. TOPCASED - Home, http://www.topcased.org/ (last access June 27, 2011)
20. VarMod-Editor Web Page, http://www.sse.uni-essen.de/wms/en/index.php?go=256#zi