

Precise WIS Development *

Francisco Javier Lucas
Martínez, Fernando
Molina Molina, Ambrosio
Toval Álvarez
Software Engineering
Research Group
Department of Informatics and
Systems
University of Murcia (Spain)
fjlucas@um.es,
fmolina@um.es,
atoval@um.es

Maria Valeria De Castro,
Paloma Cáceres,
Esperanza Marcos
Kybele Research Group
Rey Juan Carlos University,
Madrid (Spain)
valeria.decastro@urjc.es,
paloma.caceres@urjc.es,
esperanza.marcos@urjc.es

ABSTRACT

In recent years, Internet has become the platform that supports most areas in organizations. This fact has led to the appearance of specific tools for the construction of Web Information Systems (WIS). However, in these tools an absence of functionalities for verification and validation (V&V) of the models built has been detected. This work aims to redress this absence with the definition of a strategy for the specification of the models used in the WIS development that can be used with V&V objectives. The approach has been validated in a specific methodology (MIDAS) and an associated tool (MIDAS-CASE), both aligned with the MDA proposal. One of the diagrams used in this methodology has been formalized: the *extended navigation model*, achieving a precise specification of this diagram. This specification permits the definition and formal verification of properties related to this diagram, as well as its validation. The specification and verification of a concrete property are also shown.

Categories and Subject Descriptors

D.2.4 [Software Engineering]: Software/Program Verification; H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia

*This work was partially supported by DYNAMICA/PRESSURE (TIC 2003-07804-C05-05) and DAWIS (TIC 2002-04050-C02-01) projects financed by the Ministry of Science and Technology of Spain, and URJC (PPR-2004-15) project financed by the Rey Juan Carlos University. The second author was partially supported by an FPI grant from the Fundación Séneca (Spain).

General Terms

Design, Verification

Keywords

Web Information Systems, Verification and Validation, UML, MDA

1. INTRODUCTION AND MOTIVATION

In the last decade, the web has become the principal media available to all kinds of organizations to support and to spread information. Initially, the web was only considered as an element that permits organizations to have presence in Internet. However, today the web is the fundamental platform that supports most organizations areas [8]. Recently, the Web Information System (WIS) has undergone an exponential increase. Two reasons for this growth are that the potential web audience is bigger than any other media and that the web has demonstrated that it is a good media to access information for non technical users.

Initially, we might think that WIS development is done in the same way as traditional information system (IS) development. However, there are critical factors that distinguish this development [8, 7]. Moreover, studies like [5] show that the traditional methodologies can be successful in the development of simple WIS, but they are not appropriate for the development of complex systems. [5] shows how a very high percentage of WIS development projects do not satisfy the needs for which they were developed. For these reasons, specific methodologies for WIS development and, also, tools to support them have appeared. However, the most of these tools usually do not include the activities for the analysis of the consistency of the models built or, if they exist, they are not very powerful. This could cause that the WIS built contains errors that should have been detected in the design phase [14].

This work proposes to perform the analysis of the models consistency of a WIS using validation and verification activities (V&V). Validation analyzes if the observable behaviour of a system is in agreements with the requirements established for it [1]. With regard to UML models, validations are normally carried out through scenarios simulation. On

the other hand, the verification process consist of checking that a model satisfies a set of properties [1]. This verification can be made using formal or informal methods. Checklist and algorithms which analyze a model looking for errors or abnormal situation are informal verification techniques. Theorem proving and *model checking* are formal verification techniques [3]. This work is focused on formal verification, using an approach based on term rewriting, which is close to theorem proving strategies. The proposed strategy is based on the algebraic specification of the metamodels of the models used during the development process and the necessary properties implementation among them. To do this, we will use the Maude language [4]. This is an algebraic specification language based on equational and rewriting logic. This language has been used in numerous study cases [13].

One methodology used for WIS development is MIDAS [10], a methodological framework based on MDA [15] which proposes different models, and transformation rules among them, for the complete modelling of WIS. This work proposes a precise verification strategy of the models used in the construction of these WIS which hides all the formal features that are integrated within the tool (MIDAS-CASE). The objective of this strategy is to increase the reliability and quality of the models used in WIS development. In order to demonstrate the practical utility of this proposal, its application in one of the models of MIDAS is shown.

The rest of the paper has the following structure: section 2 gives a brief introduction to MIDAS methodological framework. Section 3 shows in depth, on the one hand, the advantages obtained of including this formal approach in MIDAS and, on the other hand, the main characteristics of the navigation diagram and how its has been formalized. Section 4 shows the V&V module that the MIDAS-CASE tool includes. In Section 5, we analyze the formalization and formal implementation of properties for one of the diagrams used in MIDAS. Finally, conclusions and further work are shown.

2. MIDAS FRAMEWORK

MIDAS is a methodological framework for agile development of WIS, which proposes an architecture based on the MDA proposal [15]. MIDAS proposes to model the WIS according to two orthogonal dimensions. On the one hand, by taking into account the platform dependence degree (based on MDA approach): firstly, specifying the whole system by Computation Independent Models (CIMs), Platform Independent Models (PIMs) and Platform Specific Models (PSMs) and, secondly, by specifying the mapping rules between these models. And on the other hand, according to three basic aspects [10]: hypertext, content and behaviour. MIDAS also suggests using the Unified Modeling Language (UML) [16] as a unique notation to model both PIMs and PSMs. Among the models proposed in MIDAS is the *extended navigation model*, which will be explained in depth in the Section 3. The models proposed in MIDAS framework are supported by a case tool named MIDAS-CASE

The MIDAS framework has been partially presented in others papers. Proposals for modelling the content of the system can be found in [12]; the hypertext modelling method was presented in [2] and research on modelling the behaviour of the system in [11].

3. EXTENDED NAVIGATION MODEL

3.1 Introduction

In this section, we show the main features of the *extended navigation model*, which is based on the extended slices model. In this last model, a system is decomposed into significant parts, named *slices*, and *hyperlinks*, that join these slices. There are two kinds of slices: *structural slices*, which show a piece of information which will be shown in groups; and *functional slices*, which show another kind of information or functionalities that allow us to represent interaction between the user and the WIS. Each one of these slices is represented with a class stereotyped with «SS» (*Structural Slice*) and «FS» (*Functional Slice*). For example, in Figure 1, the information about the available flights is represented using «SS» and services as *Pay Flight* or *Look for a flight* are represented using «FS».

The slices are linked with directed relations, which indicate the possible routes that could be followed from here on. MIDAS proposes to define a route for each conceptual user service. Each route is represented by an association stereotyped with «route» and a tagged value *RouteName*. If the route *R1* has a subroute *R2*, this subroute will have the tagged value *R1.R2*.

Figure 2 shows a simplified version of the metamodel of this kind of models, and an extended navigational model example, that models a web of flight shopping, is shown in Figure 1.

3.2 Extended Navigation Model Formalization

In order to carry out this activity, we propose a formal specification strategy based on metamodel formalization of the diagrams involved in the development using the formal language *Maude*.

With the metamodel formalization of a diagram, we have the following advantages: more precise models (because it is necessary to specify them without ambiguities in languages based on mathematical formalisms), possibility of verifying formally whether the models fulfil some properties, possibility of making precise transformations between models or validation of our models through simulation. *Maude* allows us to create prototypes automatically, because the metamodel specification is directly executable. The use of this feature in WIS will be tackled in a future work. In the following sections, we will see the most important elements of these models and its precise specification in Maude.

3.2.1 Routes

MIDAS proposes to define a route for each conceptual user service. A route represent the sequence of steps that a user must follow to complete the service. In the formalization of a route there are two elements involved (see Figure 2) : *ServiceRoutes* and *Routes*. In the formalization developed, a *ServiceRoutes* list is used as a route identifier. Furthermore, this list also represents the hierarchy that can exist between routes, that is, a route can have subroutes. This hierarchy has been formalized by a *ServiceRoute* list, which allows subroutes of any length to exist.

Now, we will see how a route has been formalized. A *Route* contains, on the one hand, a *ServiceRouteList* which indicates the route or subroute that it represents, and, on the other hand, information about whether it can finish. Figure 3 shows the route and route list formalization.

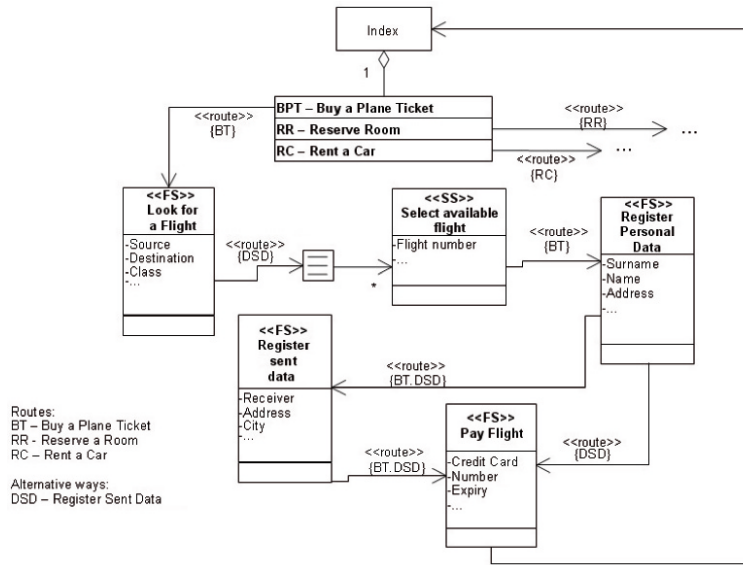


Figure 1: Example of an extended navigation model

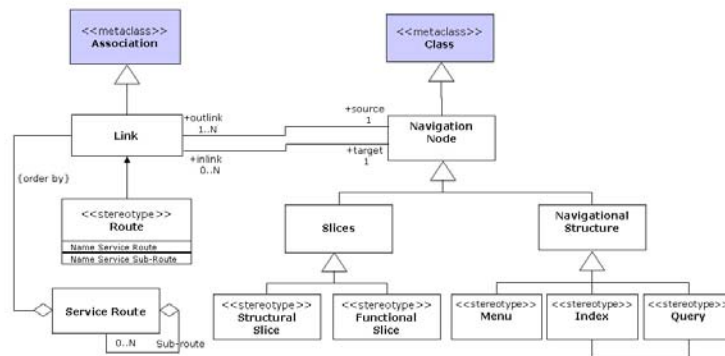


Figure 2: Simplified version of the metamodel of the *extended navigation model*

```
(fmod ROUTE is sort Route .
pr SERVICEROUTELIST .      pr ROUTEEND .
op route : ServiceRouteList RouteEnd -> Route [ ctor ] .
op getSRL : Route -> ServiceRouteList .
...
endfm) ...
(fmod ROUTELIST is ...
*** Returns true, if the route is included in the route list.
op isIn : Route RouteList -> Bool .
...
endfm)
```

Figure 3: *Route* formalization

3.2.2 Navigation Nodes

In the metamodel of this diagram, the *navigation nodes* represents significant units, named slices, mentioned in Section 3.1. These elements are made up by a *name* and a *route list*. The route list indicates which routes can finish in this node. The formalization of this element appears in Figure 4.

3.2.3 Links

A *Link* is used to connect two slices. A link is also made up by a route list that indicates which routes can cross through the link. Furthermore, a link also informs us if a route can finish in its destination node. Figure 5 shows part of the Link formalization.

3.2.4 Extended Navigation Diagram

Once we have formalized these elements, the complete navigational model can be formalized. Its constructor receives as input a *NavigationNodes* list and a *Links* list. Figure 6 shows the module that formalizes this model.

With this formalization it is already possible to represent any navigational model that we develop.

4. FORMAL VERIFICATION OF MODELS IN THE MIDAS-CASE SCOPE

The previous section showed how the extended navigation model has been formalized. In this section, we will see the place where the verification phase should be located within MIDAS-CASE tool architecture and how useful properties for *extended navigation model* can be formalized.

The MIDAS-CASE architecture includes a verification and validation module whose function is to inform users as to whether the models used are complete and correct. This module receives as input the XML document that contains the semantics of the model and parses it against its *XML Schema*, which defines the metamodel of the diagram, that is, it analyzes the *XML* file and checks that it follows the structure of the *XML Schema* used for its definition. In this way, it is possible to verify that a model is correct from the point of view of its static semantics, i.e. the way that the constructs defined in the metamodel of the diagram must be used and related between themselves. However, it is necessary to emphasize that a syntactically correct model can still contain errors and ambiguities from the point of view of the dynamic semantics of the model with regard to the *well-formedness rules* of its metamodel or, also, with regard to specific properties related with the specific domain to which the model belongs. Briefly, all these properties can be related with the static semantics of the diagram in agreements with the rules of the notation used, the dynamic semantics of the diagrams and, specially, with the semantics of the specific domain to which the diagrams belong. These last are some of the most interesting properties because, generally, conventional CASE tools and specific tools for WIS development do not support them.

The proposal developed in this work tackles this third group of properties. Previous works [9, 6] tackle properties related with the first two groups. This formal approach is based on the algebraic formalization of the diagrams involved in the development process.

5. PROPERTIES FORMALIZATION

Once we have seen the extended navigational model formalization, we now show an example of a property specification. Some auxiliary operations that will make the information management and the property specification easier have been created (for example, 'nextLinks' that returns the Links which have a certain NavigationNode as source; or 'getTargetNodes' that returns the source nodes of each Link of a Links list). Some of the most interesting properties that can be defined on this model are related with routes and possible inconsistencies and ambiguities among them.

5.1 Example: Analysis of Valid Routes

The statement of this property would be: '*All the routes that appear in a navigational model must be valid*'.

This is one of the basic properties which this kind of models must fulfil. A route is a succession of links that starts in a menu and finishes at a valid end. Two conditions must occur in order to fulfil the property:

- All the routes must have a valid end.
- If in a route one of its links is tagged with *R1* and arrives at a node, the route must be able to continue through another link also tagged with *R1*, except if the node is a valid end for this route.

This property can be guaranteed if the second condition is fulfilled for all the link-nodes that made up the route, starting at the initial node and arriving at a valid end. Hence, the property has been specified in this way. Figure 7 shows the Maude module that specifies the operations and equations that verify the property.

The schema used in this property is identical to the definition and verification of other domain properties. Firstly, the property statement is enunciated in understandable terms for the WIS developers (for example, in natural language). Then, the algorithm that verifies the property is defined informally. And finally, the property is implemented formally using *Maude*.

In order to illustrate how this property is verified, an error has been included in the model shown in Figure 1. The error introduced is the following: when a route tagged with *BT* arrives at the node named '*Look for a Flight*', there is no link tagged with *BT* leaving the node and this node is not a valid end for this route. In Figure 8, we can see the output of *Maude* when the property is verified using the operation *testValidRoute*.

It is important to emphasize that all those aspects related with the underlying formal techniques (like Maude, algebraic specifications, ...) are completely transparent to users of WIS development tool.

6. CONCLUSIONS AND FURTHER WORK

This work shows the viability and advantages of the use of formal methods in order to provide WIS development tools with a precise and powerful formal verifier. In our case, the verifier is based on Maude.

Similarly, the validation of the models used in the WIS development is possible thanks to the instantaneous executability of Maude specifications, which can be used as functional prototypes of the WIS that is being built. Thus, the tool can help modellers to build a WIS that contains

```

(fmod NAVIGATIONNODE is sort NavigationNode .
  pr NAVIGATIONNODENAME . pr ROUTELIST .
  op navigationNode : NavigationNodeName RouteList
    -> NavigationNode [ ctor ] .
  ...
  *** Returns true, if the node is a valid end for the route.
  op isValidEnd : NavigationNode Route -> Bool . ...
endfm)

```

Figure 4: *NavigationNode* formalization

```

(fmod LINK is sort Link .
  pr LINKEND . pr LINKNAME . pr ROUTELIST .
  op link : LinkName LinkEnd LinkEnd RouteList -> Link [ ctor ] .
  op getLinkName : Link -> LinkName .
  op getLinkEndSource : Link -> LinkEnd .
  ...
endfm)

```

Figure 5: *Link* formalization

```

(fmod NAVIGATIONNDIAG is sort NavigationDiag .
  pr NAVIGATIONNODELIST . pr LINKLIST .
  op navigationDiag : NavigationNodeList LinkList
    -> NavigationDiag [ ctor ] .
  op getNavigationNodeList : NavigationDiag -> NavigationNodeList .
  op getLinkList : NavigationDiag -> LinkList .
  ...
endfm)

```

Figure 6: *Extended Navigational Model* formalization

```

(fmod VERIFICATIONS is
  pr UTILSDNAV . pr STRING .
  op testValidRoute : NavigationDiag NavigationNodeList
    ServiceRouteList -> String .
  var NN : NavigationNode . var>NNL : NavigationNodeList .
  var R : ServiceRouteList . var ND : NavigationDiag .

  *** This route finishes in a valid final node. So, this partial route
  *** does not have errors.
  eq testValidRoute (ND, nullNavigationNodeList, R) = "" .
  eq testValidRoute (ND, NN>NNL, R) =
    if selectLinks(nextLinks(ND, NN), route(R,noRouteEnd))
      == nullLinkList
      and not(isValidEnd(NN, route(R,noRouteEnd))) then
      "ERROR, the route " + ... + " which is not a valid final node."
    else testValidRoute(ND,
      getTargetNodes(ND, selectLinks(nextLinks(ND, NN),
      route(R,noRouteEnd)))>NNL, R)
    fi .
endfm)

```

Figure 7: Module that verifies that the routes of a model are valid

```

reduce in ejemploNavigD :
  testValidRoute(reserveND,index,'BT)
result String :
  "ERROR, the route BT finishes in the slice lookForFlight which is
  not a valid final node. "

```

Figure 8: Property reduction in Maude

less errors, with more intuitive and more useful navigation structures for users, etc.

As further work, a list of interesting properties to verify in this diagram has also been identified, like verifying that cycles do not exist in the routes defined by the diagram, checking that unconnected routes do not exist, verifying that all the routes (or subroutes) that appear in a menu are available from it, or verifying that all valid ends reachable from a menu correspond to the ends indicated in the menu where the route starts. These properties will be formalized soon.

Another future work consists of the formalization of others diagrams proposed by MIDAS methodology for the WIS development (like models for web services definition, extended use cases models, ...), which permits us to offer a formal support for verification of the others models involved in the WIS development. Furthermore, the transformations that are carried out between the different models of the MIDAS methodology are being tackled.

7. REFERENCES

- [1] R. Binder. Testing object-oriented systems: models, patterns and tool Massachusetts. *The Addison-Wesley object technology series. A. Wesley*, 1999.
- [2] P. Cáceres, V. De Castro, and E. Marcos. Navigation Modeling from a User Services Oriented Approach. *Proceedings of the Third Biennial International Conference in Advanced in Information Systems (ADVIS 2004)*. T. Yakhno (eds.). LNCS 3271. Springer Verlag, pp. 150-160, 2004.
- [3] E. M. Clarke and J. M. Wing. Formal methods: state of the art and future directions. *ACM Computing Surveys (CSUR)*, Volume 28 Issue 4, 1996.
- [4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcote. *Maude 2.0 Manual. Versión 1.0*. <http://maude.csl.sri.com/>, 2003.
- [5] M. Epner. Poor Project Management Number-One Problem of Outsourced E-Projects. *Research Briefs, Cutter Consortium*, November 2000.
- [6] J. L. Fernández Alemán and A. Toval Álvarez. Improving System Reliability via Rigorous Software Modeling: The UML Case. *Proceedings of the 2001 IEEE Aerospace Conference (track 10: Software and Computing)*, Montana, USA IEEE Computer Society, March 2001.
- [7] A. Ginige. Web Engineering: Managing the Complexity of Web Systems Development. *Workshop on web engineering: Web engineering: managing the complexity of web systems development. Proceedings of the 14th international conference on Software engineering and knowledge engineering*, July 2002.
- [8] T. Isakowitz, M. Bieber, and F. Vitali. *COMMUNICATIONS OF THE ACM*, volume 41, No. 7. Guest Editors, July 1998.
- [9] F. J. Lucas Martínez and A. Toval Álvarez. A Precise Approach for the Analysis of the UML Models Consistency. *BP-UML'05: 1st International Workshop on Best Practices of UML, dentro del 24th Int. Conf. on Conceptual Modeling (ER 2005)*, Klagenfurt (Austria). 24-28 octubre, 2005. ISBN: 3-540-29395-7. LNCS 3770 Springer, November 2005.
- [10] E. Marcos, P. Cáceres, and V. De Castro. An approach for Navigation Model Construction from the Use Cases Model. *The 16th Conference On Advanced Information Systems Engineering. CAISE'04 FORUM*, pp. 83-92, 2004.
- [11] E. Marcos, V. De Castro, and B. Vela. Representing Web Services with UML: A Case Study. *The First International Conference on Service Oriented Computing (ICSOC03)*. M.E. Orłowska, S. Weerawarana, M.P. Papazoglou, J. Yang (eds.). Springer Verlag, pp.15-27, 2003.
- [12] E. Marcos, B. Vela, and J. Cavero. Methodological Approach for Object-Relational Database Design using UML. *Journal on Software and Systems Modeling (SoSyM)*. Springer-Verlag. France, R., Rumpe, B. (eds.). Volume SoSyM 2, pp.59-72., 2003.
- [13] N. Martí-Oliet and J. Meseguer. Rewriting logic: Roadmap and Bibliography. *Theoretical Computer Science 285(2): 121-154.*, 2002.
- [14] B. Meyer, C. Mingins, and H. Schmidt. Providing Trusted Components to the Industry. *Computer*. pp. 104-105, May 1998.
- [15] OMG. *MDA Guide Version 1.0.1*. <http://www.omg.org/mda>, 2001.
- [16] OMG. *Object Management Group. UML Superstructure 2.0. Draft adopted Specification*. Retrieved from: <http://www.omg.org/uml>, 2004.