Journal of Digital Information, Vol 5, No 2 (2004)

WIED: A Web Modelling Language for Modelling Architectural-Level Information Flows

Rachatrin Tongrungrojana and David Lowe
Faculty of Engineering, University of Technology Sydney,
P.O. Box 123 Broadway Sydney, NSW 2007, Australia
Email: rachatrin.tongrungrojana@uts.edu.au, david.lowe@uts.edu.au

Abstract

The ability to reliably and consistently develop systems that utilise Internet and Web technologies has become increasingly important. These systems are typically both functionally complex and information-rich, and have a number of unique characteristics that should imply specific changes to the development processes, methods and models that are adopted. One aspect that has received increasing attention is information modelling for these applications, particularly with respect to aspects such as navigation models and their relationships to the underlying content. These models have typically focussed on modelling at a relatively low-level, however, and have failed to address higher-level aspects, such as architectural and even business process modelling. The paper introduces a formal information modelling set which can be considered as a companion to to an existing modelling language - WebML - that facilitates information modelling at this higher level of abstraction. We argue that this modelling approach will provide a clearer connection between an understanding of business models and processes, and the lower-level designs typically represented in existing models.

1 Introduction

From its introduction in the early 1990s, the Web has continued to evolve at a fast pace. Online systems are becoming increasingly crucial to almost all sectors in society. It is becoming common practice for organisations to utilise Web and Internet technologies in the development of new systems and in adapting the deployment of existing systems. Web systems are often seen as part of an external marketing strategy or as a tool to enhance the internal communication and, as such, the efficiency of the organization.

These systems are much more complex than simple Web sites containing static pages. They typically employ Web technologies to provide a complex distributed front-end combined with high-performance back-end software systems that integrate new components with existing legacy applications to support critical business processes (Powell 1998, Burdman 1999, Pressman 2001, Shelford and Remillard 2002).

Various approaches have been developed or adapted for representing these complex Web systems from different points of view and for different purposes. For example, the e3-value™ business modelling method (Gordijn *et al.* 2000b, Gordijn *et al.* 2001b) emphasizes a business modelling perspective, UML (Booch *et al.* 1999, OMG 2004) focuses on functional aspects, and WebML (Ceri *et al.* 2000) concentrates on the informational aspects of Web systems.

The emergence, over the last decade, of informational modelling approaches is a reflection of the increasing awareness of the significance of this aspect of Web-enabled systems. Example approaches, such as RMM (Isakowitz *et al.* 1995) and OOHDM (Schwabe and Rossi 1998), and more recently WebML (Ceri *et al.* 2000) and adaptations of UML (Baumeister *et al.* 1999, Conallen 1999, Baresi *et al.* 2001, Hennicker and Koch 2001, Koch and Kraus 2002), have provided the ability to model the content that underpins these applications, and (to a limited extent) the way in which we interact with this information. However, these approaches still have various limitations. For instance, all current approaches lack the ability to model Web systems at higher levels of abstraction.

This paper proposes a companion notation to an existing modelling language, WebML (Ceri *et al.* 2000), to address this limitation. This companion model is referred to as the Web Information Exchange Diagram (WIED). A key point in this model is that the WIED approach is built around the notion of information flows at the level of understanding business processes. This enables the models to form a link between higher-level models (specifically, business models) and lower-level detailed design models – a characteristic that is crucial in Web development, where the systems under development often lead to fundamental changes in business processes and, indeed, in business models.

We focus on the presentation of WIED. The remainder of this paper is organized as follows. Section 2 presents the background to our research and WIED, considering related work in this area and establishing the need that our work addresses. Section 3 introduces WIED through an illustrative example, and then provides the formal WIED definitions. In Section 4 we briefly present a set of rules to generate low-level WebML designs from a higher-level WIED model. Finally, in section 5 we discuss the implications of this research and present some ideas for further work.

2 Background

As discussed above, the last decade has seen the rapid emergence of systems that utilise Web technologies to support the integration of complex functionality with rich information handling. This emergence has been accompanied by the development of modelling languages capable of representing some, though not all, of the aspects of the design of these systems. Existing modelling approaches have been evaluated and discussed in numerous articles (Christodoulou *et al.* 1998, Kappel *et al.* 2000a, Kappel *et al.* 2000b, Kappel *et al.* 2001, Schwinger 2001, Gu *et al.* 2002). As an example of this work, Figure 1 shows an evaluation of modelling approaches in terms of the strength of their support for both functional and informational characteristics.

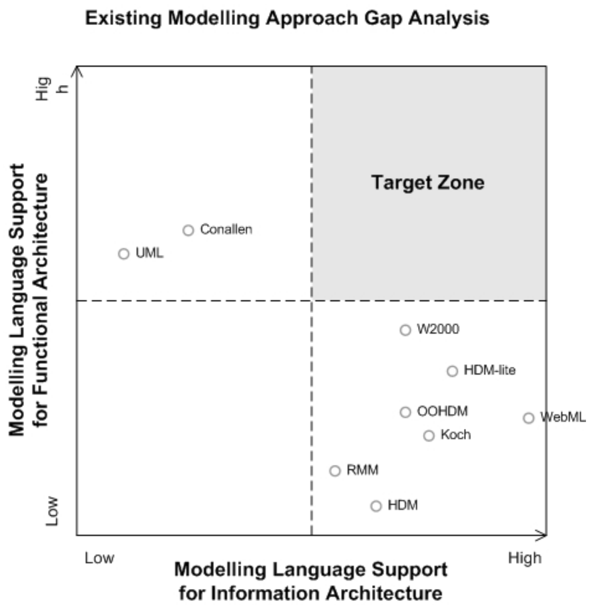


Figure 1. Evaluation of modelling approaches (from Gu *et al.* 2002)

While the literature has considered the extent of support for modelling the design of Web systems (ranging through both functional and informational characteristics), it has not adequately considered the extent of modelling support at various levels of abstraction, particularly in terms of the various levels that may exist between an original business need and the ultimate detailed design. To illustrate this, Figure 2 shows several different levels of modelling that might typically occur in representing the design of Web-enabled systems. At the top level we can model the actual business (business goals, business processes, etc.) that is utilising these systems. As an example, a typical model at this level might represent the value exchanges between the organisation and other entities that enable the organisation to achieve its business goals. While modelling notations at this level are quite diverse, and often *ad hoc* (or at least relatively informal; for example, it is common to simply use natural language or simple flow-charting), some more formal approaches do exist. A typical example is the e3-value™ business modelling notation (Gordijn *et al.* 2000a, Gordijn *et al.* 2000b, Gordijn *et al.* 2000c, Gordijn *et al.* 2001a, Gordijn *et al.* 2001b, Gordijn 2002a, Gordijn 2002b, Gordijn and Wieringa 2003). This model focuses on the core concept of value, and expresses how value is created, interpreted and exchanged within a multi-party stakeholder network.

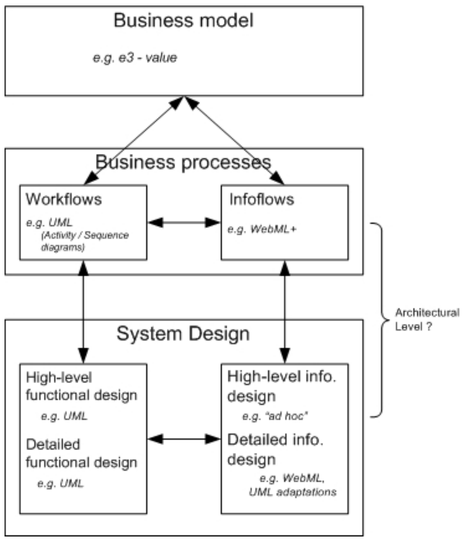


Figure 2. Relationships between modelling domains

Conversely, at the lowest level in this diagram are models of the detailed design. These models typically capture design elements that have a direct correspondence to specific implementation artefacts. For example, in the implementation of a Web system, detailed design models might describe concrete functional elements (such as code modules and communication interfaces) and information elements (such as page content and navigational links). Design models to represent functional elements are relatively well-established both within the software literature and within current commercial practice. The dominant modelling language within conventional software development is (arguably) now UML (Kobbryn 1999, OMG 2004). This can be used to model both detailed design and higher-level designs through a complex suite of diagrams that are all treated as views on to a consistent underlying model (the issue of higher-level modelling will be discussed in much more detail in section 3). Further, various approaches have been created to adapt UML for modelling the functional aspect of Web systems. For example, User Interaction Diagrams (UID) (Vilain *et al.* 2000) can model the way in which users interact with a Web system. It has been argued, however, that UML does not provide a particularly clear connection between business models and the functional design of systems (de Cesare *et al.* 2002, Lowe 2003).

In terms of modelling the information design, the situation is somewhat less mature. At the business modelling level, if information is considered at all then it is typically only in terms of the exchange of information that occurs between the various stakeholders. The stakeholders needs are often modelled, however, as is the exchange of value that occurs. At the level of detailed design, Web information models (such as RMM, OOHDM, WebML, WSDM (Troyer and Leune 1998)) typically capture aspects such as the content viewpoints, interface metaphors, and navigational structures. We will argue that to connect these two sets of models, we need to model not only the information itself, but also the relationship between the underlying content and the user-perceived views of that content, the interactions with those views (such as navigational aspects), and the ways in which the information is represented and presented to the users. This modelling tends to be much more complex than traditional "data modelling" (which used approaches such as entity-relationship and data flow modelling) since representing the information context becomes much more important (in terms of the way in which the content is interpreted and therefore the value it provides).

While existing standard modelling techniques (particularly UML) can be used to represent the functional aspects, they are not as effective at representing these informational aspects. In general, UML was originally conceived as a general-purpose language capable of modelling, specifying, visualizing, and documenting models of software systems in a way that meets all of these requirements. It defines 12 types of diagrams, divided into three categories: four diagram types represent static application structure; five represent different aspects of dynamic behaviour; and three represent ways you can organize and manage your application modules (OMG 2003). It can be seen that these 12 types of diagrams originally did not address information modelling while some of them mainly focus on functionality modelling (Ziegler 2002). For example, *use case* diagram describes how external actors interact with the system and apparently focuses on functionality (Ek 2001, Canyonblue 2003), *class* diagram is used to refine the *use case* diagram and define a detailed design of the system. Each class in the *class* diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed methods of the class (Chitnis *et al*. 2003).

Although some attempts have been made to adapt UML to support information models (e.g. interesting approaches include WAE by Conallen (1999), and work by Hennicker and Koch (2000)), these are still relatively simplistic and often suffer from a notational confusion. For example, modelling constructs (such as classes) normally associated with functional elements are used to represent information elements, or modelling constructs (such as stereotypes in UML) are used incorrectly or inconsistently. Further, some other approaches based on UML (such as Mandel *et al*. 1999, Dolog and Bieliková 2002) and software engineering methods (such as WebOPEN (Graham *et al*. 1997, Henderson-Sellers *et al*. 1998)) tend to focus only on modelling lower-level constructs.

Beyond approaches based on UML, a number of other modelling approaches that are specific to information design have appeared over the last decade. The earliest of these emerged out of the hypertext community, and include approaches such as Relationship Management Methodology (RMM) (Isakowitz *et al*. 1995), OOHDM (Schwabe and Rossi 1998), and others. More recently a number of approaches have been developed that utilise and adapt software modelling approaches, and in particular UML (Baumeister *et al*. 1999, Conallen 1999, Baresi *et al*. 2001, Hennicker and Koch 2001, Koch and Kraus 2002). Of particular interest in this paper, as an exemplar of these modelling approaches, is WebML (Ceri *et al*. 2000). WebML incorporates both an XML-based formal description of a model as well as a graphical notation. While the authors of WebML (and of many other current approaches) claim to address the full development spectrum, including higher-level descriptions of the informational elements of Web sites, we contend that their focus is primarily on lower-level information modelling. These models typically incorporate modelling of the underlying data (in WebML this is referred to as the structural model), the way in which this data is composed into pages (in WebML, the compositional model) and the topology of the page inter-relationships (in WebML, the navigational model). On top of these can be layered presentational and adaptation models.

At the higher level in the diagram (the middle level) are models of the high-level system architecture that capture the domains of functionality and the domains of information that are needed to support the value exchange in the business domains. As discussed previously, functional aspects at this level are well supported by UML. Some UML models (such as activity diagrams) can be used to represent the business and operational workflows of a system.

Modelling informational aspects is a problematic issue, however. While standard software modelling approaches such as UML support functionality modelling, they do not address particularly well the modelling of information (which is more crucial and emphasised in Web systems). One good example of a UML diagram that focuses on high-level modelling is the *activity* diagram. This diagram shows the activity and the events that cause an object to be in a particular state. It can be used to represent the business and operational workflows of a system. While *activity* diagrams can be used for modelling some aspects of business processes, we contend that it is not appropriate for modelling information aspects of business processes. For example, *activity* diagrams mainly focus on the representation of activities and events (which can be seen as functional aspects of the business process). However, some researchers argue that activity diagrams can be used for information modelling at this level of abstraction (Lieberman 2001). While this is valid to a certain extent, activity diagrams are unable to capture accurately the rich domain context that is important in understanding information relationships and flows (as distinct from data flows) that support value exchanges in business. This includes the relationships between underlying content and the user-perceived views of that content, the interactions with those views, and the ways in which information is represented and presented to the users.

To elaborate, while there has been considerable attention recently on the area of information architectures, the notations and models used to represent these information architectures are rarely consistent with those used for lower-level information modelling or higher-level business models, and as a result these are rarely integrated effectively (a good source on information architectures is the Argus Centre for Information Architecture, or Rosenfeld and Morville (1998) for more specific information).

Similarly, while information architectures often develop an understanding of user interactions and engagement with a site (Lamar 2001, Haverty 2002) and the way in which this influences the information organisation, this is often only modelled at the level of information needs and usage scenarios,adopting modelling techniques such as *use cases* (Dutoit and Paech 2000, Baresi *et al*. 2001). The nature of the information exchange and the internal inter-relationships between these information domains is often overlooked or not modelled explicitly. Nor do the low-level information models and information architectures usually effectively consider the information environment in which the system exists.

So this raises the question of what information we really should be modelling? The functional elements of business processes are well captured in work flows, so we believe in the same way that the informational elements should be able to be captured in information flows. These information flows should describe the conceptual exchange of information within the organisation and between the organisation, the system and the external actors at the business process level, and also the relations between these information "units". This includes the external information context and domain, the source and sinks of the information being designed, and the way in which users interact with this information. This is analogous to data flow modelling in conventional structured software design except that we are interested in information rather than data, and hence the context used to understand the information is important.

To elaborate the information flows concept, the implementation of software/Web systems can be seen as dependent on an ability to derive detailed information designs that support business processes and models. As we discussed earlier in this section, a business model shows the way of doing business in terms of stakeholders creating and exchanging objects of value with each other. A low-level detailed design represents design entities that relate to specific implementation artefacts. To have a low-level detailed design that supports a high-level business model, we should be able to model how value-creating activities are carried out.

Also mentioned earlier, modelling of business processes is well supported in term of functional (activity) modelling. Various approaches have been developed for representing the business and operational workflows of a system such as activity diagrams for presenting the sequential flow of activities, and data flow diagrams for presenting the flow of data from external entities into and within the system. However, in term of information modelling, there is no approach focusing especially on this area. This might not be a critical problem in a conventional software system that mainly addresses functionality, but this is a serious problem in Web system development as its low-level detailed design contains significant information aspects. Some researchers show there is a gap in Web system development, especially information modelling at the business process level. This requires some work to fill the gap and support the information aspect of the exchange of value in business domains, and also to be able to drive low-level design.

In response to this problem, we propose a new modelling notation, called WIED - Web Information Exchange Diagram - which aims to represent exchanges and transformations of information between actors for supporting value exchanges in businesses. The proposed model also represents the relationships between the underlying information, the actor interaction with the information, the ways in which the actor influences the information, and the ways in which information is represented and presented to the users at higher levels of abstraction. In effect, our model forms the bridge between business concerns and detailed information designs that we have argued is largely lacking in existing models.

As an aside, it is also important to note we adopt a custom notation. This is so that it can form a notational "bridge" between two existing modelling languages (e3-value for representing business value exchanges, and WebML for representing low-level information designs). The formal model described below is, however, quite transportable and it would be quite straightforward to map the formal model to a notation based on other standards such as UML.

3 WIED Model

This section presents the core concepts of WIED through a hypothetical example and then provides the formal WIED national specification.

3.1 Overview

The WIED enables developers to express the core informational features of a system at a higher level of abstraction (when compared to normal Web design notations), without committing to detailed architectural details. It can be considered as a companion to WebML, which is an interesting, widely-adopted notation of visually specifying complex Web systems at the conceptual level. The purpose of WIED modelling is to define Web system information flows (both internal and external) at business process level for supporting the exchange of value between stakeholders in the domains of business (which is represented in business model). As with WebML, we have defined both a graphical notation and an XML-based formal notation for representing WIED models. The graphical notation is designed to allow it to be effectively communicated to non-technical members of development teams. Next, the WIED is introduces through an illustrative example.

3.2 WIED by example

Figure 3 shows an example of a WIED model for a hypothetical example: TransAir is a fully-fledged airline which takes an innovative look at lowering air fares. The strategy adopted by TransAir is to lower operating costs (such as ticket processing and maintenance of physical office spaces) by running their business activity through an online Web system. The TransAir system allows customers to purchase tickets and enquire about flight and booking information through a Web interface.

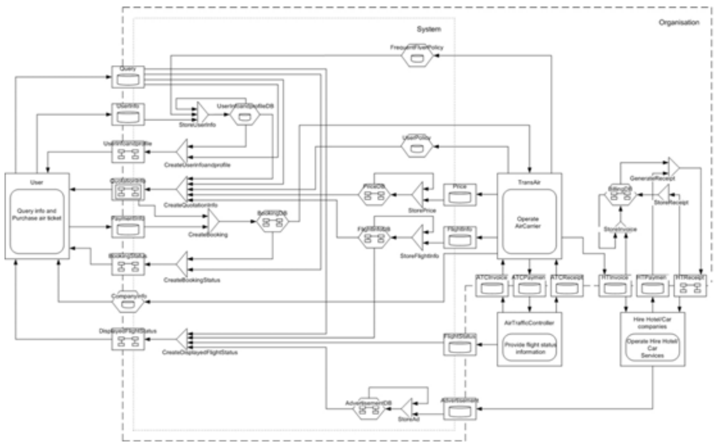


Figure 3. Example WIED model (for TransAir business/system)

This example has four participating actors: the TransAir organisation itself as an internal actor, and three external actors: users, an air traffic control organisation, and hotel/car hire companies. In brief, internal actors such as TransAir provide information directly to the system – in this case, flight formation, price information, a set of user policies, a set of frequent flyer policies and company information. TransAir also has financial information exchanges between its organisation and external actors (e.g. invoice and receipt information). In terms of external actors, the air traffic control organisation provides flight status (e.g. flight arrival times) to the system and exchanges financial information with TransAir; hotel/car hire companies provide advertisements for their services to TransAir to be included in information provided to users. Users provide queries for information, payment information as well as user information (such as personal information) to the system while they receive user information and profiles, quotation, booking status and flight status from the system.

Consider what is represented in the model. A set of information units are located within and also on the organisation boundary (shown as a dashed geometrical polygon). These units represent coherent and cohesive domains of information that are managed or utilised by the organisation. All information within a single unit shares a common context and a common derivation (this is explained shortly). They do not map directly to pages or sets of pages: a single Web page may contain partial information from multiple units. Similarly, an information unit may be distributed over multiple pages. Different types of information units are represented graphically using different types of icons.

Some information units are provided directly by actors. For example, flight status is provided by air traffic controllers, and frequent flyer policies are provided by TransAir. However, many information units are derived from other units rather than being provided explicitly. These derivations (shown as triangles with incoming and outgoing arrows) capture the inter-relationships between the information units. For example, QuotationInfo is derived from the flight information database, the price database, user policies, user profile information and query information provided by the user.

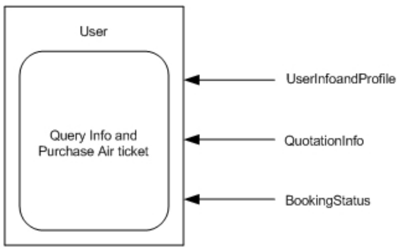
Furthermore, the system boundary (shown as a dotted geometrical polygon) encloses only the set of information units that are utilised and/or managed by the system under consideration (i.e. the elements enclosed by the system boundary are a subset of the elements enclosed by the organization boundary). The organisation boundary captures the interface between the organisation and external stakeholders (i.e. it is crucial in supporting the business modelling). Conversely, the system boundary captures those elements of the organisation that can be managed by the system (i.e. it is crucial in supporting the detailed system design).

3.3 WIED notation specification

The WIED supports a number of different modelling entities. The full formal definition of these entities and their inter-relationships is provided by the WIED DTD in Appendix 1. A full example of the application is given in Appendix 2.

3.3.1 Actor unit

An Actor unit is defined to show roles that users play with respect to the system. In WIED, Actor units show how users participate with information flows. Users can be either internal or external to the system and play two main roles: information supplier and information consumer. The formal definition of an Actor unit requires both information about the actor (e.g. name and role) and specification of the information sources that flows to the actor. Syntactically, Actor units are defined using the ACTORUNIT element, which includes an INCOMINGFLOW element and attributes such as the role of the user. Figure 4 shows the WIED graphic notation for representing an Actor unit and its underlying entity.

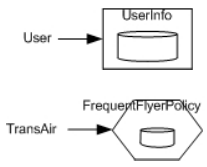


```
<ACTORUNIT
id="User" role="QueryInfoandPurchaseAirticket">
<INCOMINGFLOW from="UserInfoandProfile"/>
<INCOMINGFLOW from="DisplayedEmail"/>
<INCOMINGFLOW from="UsageInfo"/>
</ACTORUNIT>
```

Figure 4. Example of WIED graphical and XML-based notations for ActorUnit

3.3.2 Supplied information unit

A supplied information unit presents information about a single information object that is provided by a supplier (an actor who supplies information). There are two types of supplied information unit, differentiated by their persistency: a supplied transient unit exists only for the duration of its immediate utilisation, whereas a supplied persistent unit remains past its immediate use. Syntactically, a supplied information unit is represented by the SUPPLIEDINFOUNIT element, which has an attribute for identifying the type of unit (transient or persistent). Each supplied info unit is allowed to have only one supplier, which is represented by the supplier attribute. Figure 5 shows the WIED graphic notation for representing a supplied information unit and its underlying entity.



```
<SUPPLIEDINFOUNIT id="UserInfo"
supplier="User" type="transient"
description="User information provided by User"/>

<SUPPLIEDINFOUNIT id="FrequentFlyerPolicy"
supplier="TransAir" type="persistent" description="FrequentFlyerPolicy provided by TransAir"/>
```

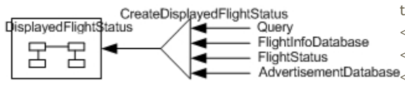
Figure 5. Example of WIED graphical and XML-based notations for SuppliedInfoUnit

3.3.3. Derived information unit

Certain types of information units are not supplied directly, but are derived from other information units. This derivation may ultimately be implemented as a process that either dynamically or statically generates the underlying content (i.e. the derivation can be seen as a process or activity that make changes to information, e.g. CreateQuotationInfo is a process that collects information from the flight information database, price database, user policies, user profile information and query information provided by the user and then uses this information to generate QuotationInfo). In general, the derivation can be performed by an actor or by the system itself, and can be performed automatically or manually. For example, TransAir automatically performs CreateBooking from QuotationInfo and PaymentInfo after PaymentInfo is provided by the user.

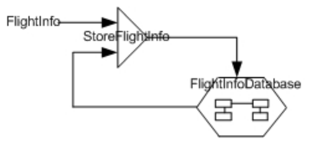
A derived information unit presents information about a single information object that comes from a derivation. There are two types of derived information units (again identified by their persistency): a derived transient unit exists only for the duration of its immediate utilisation, whereas a derived persistent unit remains past its immediate use.

A derived information unit specification has two mains parts: the attributes used to provide information about the information unit; and an indication of the source of the information that is used to derive the unit. Each derived information unit must have only one derivation. Syntactically, a derived information unit is represented by the DERIVEDINFOUNIT element, which includes a child DERIVATIONUNIT element. Inside the DERIVATIONUNIT element, the INCOMINGFLOW element is used to specify the source information flows. The type of the derivation (automatic or manual) is represented by the type attribute. The following example shows two derived info units for representing email presented to a user and a database for storing incoming emails. Figure 6 shows the WIED graphic notation for representing a derived information unit and its underlying entity.

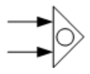


```
<DERIVEDINFOUNIT id="DisplayedFlightStatus"
type="transient"
description="DisplayedFlightStatus">
<DERIVATIONUNIT id="CreateDisplayedFlightStatus" description="Create displayed
flight status information by integrating from 4 sources of information"
type="auto">
<INCOMINGFLOW from="Query"/>
<INCOMINGFLOW from="FlightInfoDatabase"/>
<INCOMINGFLOW from="FlightStatus"/>
<INCOMINGFLOW from="AdvertisementDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>


<DERIVEDINFOUNIT id="FlightInfoDatabase"
type="persistent"
description="Storage of Flight information">
<DERIVATIONUNIT id="StoreFlightInfo" description="Store Flight information into
database" type="auto">
<INCOMINGFLOW from="FlightInfo"/>
```



```
<INCOMINGFLOW from="FlightInfoDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>
```



```
<DERIVATIONUNIT id="..." type="manual">
<INCOMINGFLOW from="..."/>
<INCOMINGFLOW from="..."/>
</DERIVATIONUNIT>
```

Figure 6. Example of WIED graphical and XML-based notations for DerivedInfoUnit

3.3.4 Information Flow

Information flow describes the flow of information within the system and between the system and the external actors. Syntactically, information flows are only modelled indirectly as part of the information and actor units (mostly seen as IMCOMINGFLOW attributes). A flow is represented as an arrow from the source to the destination.

4 WIED: linking business models to detailed designs

For most Web or e-commerce systems development, systems tend to evolve over time, at least partially as a result of the system itself, leading to changes in the business models and processes (Lowe 2002a, Lowe and Eklund 2002, Lowe 2003). As such, it is important that we can understand the relationship between low-level information design, business processes and business models. This should be supported by an ability to identify changes to a business model arising out of changes to detailed design, and vice versa.

We have considered how WIED can be linked to models at different levels of abstraction, particularly business models and detailed system information designs, and hence how we can provide a bridge between the system design and the business operation. This implies the ability to understand the association between the WIED model and models at other levels, and to support the derivation of a suitable WIED model from/to the business model, and an appropriate WebML detailed design model from/to the WIED model.

The details of these transformations are not given in this paper but have been described by Tongrungrojana and Lowe (2003). It is, however, worth providing some examples of the transformation. As discussed above, modelling notations at the business model level are quite diverse and often *ad hoc*. A typical example of the more structured models is e3-value. This approach is built around the notion of value networks. e3-value focuses on the core concept of value, and expresses how value is created, interpreted and exchanged within a multi-party stakeholder network and can be used for expressing the strategic intent of the way of doing business. More elaborate explanations and applications of e3-value are given in Gordijn *et al.* (2000a), Gordijn *et al.* (2000b), Gordijn *et al.* (2000c), Gordijn *et al.* (2001a), Gordijn *et al.* (2001b), Gordijn (2002a), Gordijn (2002b), and Gordijn and Wieringa (2003).

Whilst an e3-value model focuses on the exchange of value, the WIED approach is constructed around the concept of information flows. This means that information exchanges in WIED can be related to value exchanges in e3-value. We can illustrate the way in which business models relate to WIED by looking at a particular example that of mapping e3-value to WIED. Using the algorithm defined in Tongrungrojana and Lowe (2003) we can convert the e3-value model shown in Figure 7 to the WIED model shown in Figure 3 (or vice versa).

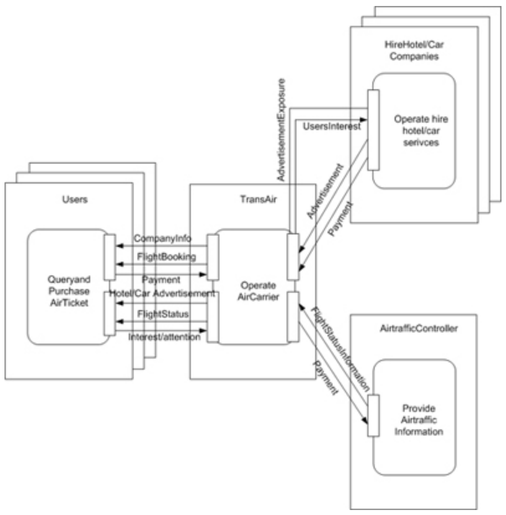


Figure 7. Example of e3-value model

As an example of how this approach can be used to understand business implications, consider the situation where the information derivations (shown in Figure 3) are modified so that the DisplayedFlightStatus is no longer derived from the advertisements (i.e. this information did not contain advertisements) but instead the QuotationInfo derivation included advertisements. The value exchanges that will be affected in the e3-value model can be readily identified. In effect, the Interest/Attention value exchange will be diminished (and in particular, will not provide the type of attention that will be useful to advertisers, thereby affecting the AdvertisementExposure value exchange, and ultimately the payment than can be sought from advertisers).

Similarly, we have defined an algorithm for mapping between WIED models and WebML detailed design models. In WebML the structural schema defines the data domain for the application. The hypertext model contains two elements: the composition and the navigation models. Together these define the abstract information interface. The process begins by generating a WebML structural schema from the WIED model and then continues into the definition of the WebML site view. The information units that exist on the system boundary provide a basis for identifying the information with which the external actors interact and hence the Web pages which users are likely to see and navigate between. An example of the result of this process is shown in Figure 8.

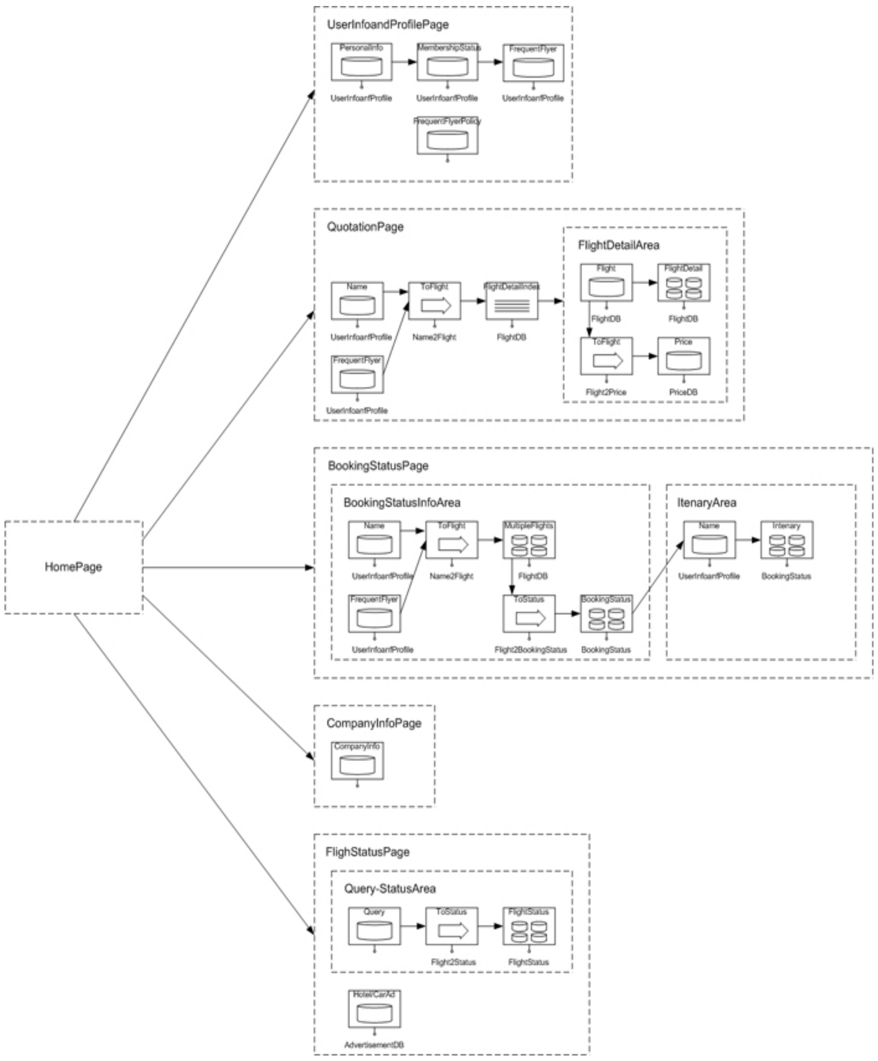


Figure 8. WebML site view

5 Conclusions and further work

This paper has presented WIED, a high-level specification language for defining Web system information flows visually based on WebML. As has been discussed and illustrated by example, WIED addresses the definition of the flows of information at a level that links to business processes, and particular value exchanges. It therefore provides a bridge between business models and detailed information designs. We argue that WIED can be used for modelling information aspects that cannot be appropriately modelled by standard modelling approaches such as UML. WIED also provides a clearer view of an information architecture than the typical site maps that are often adopted. In particular, this approach clarifies the context within which the information exists and the inter-relationships between the various sources and sinks of information. This is also an important step in providing a clearer connection between information and functional perspectives of a system.

The WIED approach still has some limitations, however. For example, while WIED provides linkages to some modelling approaches (e.g. a widely-adopted low-level information modelling approach such as WebML and a typical business modelling approach such as e3-value), it doesn't support linkages to standard modelling approaches such as UML. We have also yet to define clearly the relationship to functional modelling (which can be appropriately represented by the UML model suite).

Furthermore, in terms of concept evaluation, we have conducted experiments aimed at evaluating the extent to which WIED models are able to (when contrasted with purely textual descriptions) provide more rapid and consistent communication of information flows during the Web system design process. The experiment design and results were given by Lowe and Tongrungrojana (2003). In brief, we found that participants are able to understand a system based on the WIED model more consistently than based on a purely textual description. Moreover, development tools for creating and drawing WIED models have also been created.

Ongoing work is focusing on refining the model (such as including different types of information provision and derivation) and clarifying the relationships with the business models and processes. We are also undertaking a further case study focusing on whether the WIED approach can be used effectively to facilitate modification of business models and processes to reflect changes that are made to the underlying designs. The outcomes of this evaluation will be reported elsewhere.

Acknowledgements

This paper is a part of Rachatrin Tongrungrojana's PhD thesis supervised by Professor David Lowe. We particularly thank all the participants of the study. Their participation and feedback contributed in great part to this study. We also thank the anonymous reviewers for their helpful comments.

References

Baresi, L., Garzotto, F. and Paolini, P. (2001) "Extending UML for Modeling Web Applications". In *Proceeding of the 34th Hawaii International Conference on System Sciences*, Hawaii, pp. 1285-1294

Baumeister, H., Koch, N. and Mandel, L. (1999) "Towards a UML Extension for Hypermedia Design". In *Proceeding of <<UML>> 1999: The Second International Conference on The Unified Modeling Language*, Fort Collins, CO (IEEE), pp. 614-629

Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The Unified Modelling Language User Guide* (Addison-Wesley)

Burdman, J. (1999) *Collaborative Web Development* (Addison-Wesley)

Canyonblue (2003) "Canyonblue's Introduction to UML", CanyonBlue Inc. http://www.canyonblue.com/uml_introduction.pdf

Ceri, S., Fraternali, P. and Bongio, A. (2000) "Web Modeling Language (WebML): a modeling language for designing Web sites". In *Proceedings of the ninth International World Wide Web Conference*, Amsterdam, May, pp. 137-157 http://www9.org/w9cdrom/177/177.html

de Cesare, S., Lycett, M. and Patel, D. (2002) "Business modelling with UML: Distilling directions for future research". In *Proceedings of the 4th International Conference on Enterprise Information Systems* (Kluwer Academic)

Chitnis, M., Tiwari, P. and Lakshmi, A (2003) "UML Overview" *developer.com* http://www.developer.com/design/article.php/1553851

Christodoulou, S., Styliaras, G. and Papatheodourou, T. (1998) "Evaluation of Hypermedia Application Development and Management Systems". In *Proceedings of ACM Hypertext'98 Conference*, Pittsburgh

Conallen, J. (1999) *Building Web Applications with UML* (Reading, MA: Addison-Wesley)

Dolog, P. and Bieliková, M. (2002) "Hypermedia Modelling Using UML". In *Proceedings of ISM 2002 - Information Systems Modelling*, edited by Petr Hanáček (Ostrava, Czech Republic: MARQ) http://www.dcs.elf.stuba.sk/~bielik/publ/abstracts/2002/ism2002.pdf

Dutoit, A. and B. Paech (2000) "Supporting Evolution: Rationale in Use Case Driven Software Development". In *Proceedings of International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'2000)*

Ek, A. (2001) "Real-time UML powered by SDL" http://www.systemes-critiques.org/ek.pdf

Gordijn, J. (2002a) "e^3-value in a Nutshell". In *Proceedings of International Workshop on E-business Modeling*, Lausanne http://www.cs.vu.nl/~gordijn/bmns.pdf

Gordijn, J. (2002b) "Value based requirements engineering: Exploring innovative e-commerce ideas". PhD thesis http://www.cs.vu.nl/~gordijn/thesis.pdf

Gordijn, J., Akkermans, H. and Vliet, H. V. (2000a) "Value based requirements creation for electronic commerce applications". In *Proceedings of the 33rd Hawaii International Conference On System Sciences*, Hawaii (IEEE) http://www.cs.vu.nl/~gordijn/hicss-web.pdf

Gordijn, J. and Akkermans, J. M. (2001a) "A Conceptual Value Modeling Approach for e-Business Development". In *Proceedings of the First International Conference on Knowledge Capture, Workshop Knowledge in e-Business (K-CAP 2001)*, pp. 29-36 http://www.cs.vu.nl/~gordijn/er2001.pdf

Gordijn, J. and Akkermans, J. M. (2001b) "e3-value: Design and Evaluation of e-Business Models". *IEEE Intelligent Systems,* Vol. 16, No. 4, 11-17 http://csdl.computer.org/dl/mags/ex/2001/04/x4011.pdf

Gordijn, J., Akkermans, J. M. and Vliet, J. C. V. (2000b) "Business Modelling is not Process Modelling". In *Proceedings of ECOMO 2000: Conceptual Modeling for E-Business and the Web*, LNCS Vol. 1921, (Springer-Verlag) http://www.cs.vu.nl/~gordijn/ecomo-gordijn.pdf

Gordijn, J., Akkermans, J. M. and Vliet, J. C. V. (2000c) "What's in an electronic business model?". In *Proceedings of the 12th International Conference on on Knowledge Engineering and Knowledge Management (EKAW 2000)*, Juan-les-Prins, France (Springer-Verlag), pp. 257-273 http://www.cs.vu.nl/~gordijn/ekaw2000.pdf

Gordijn, J. and Wieringa, R. J. (2003) "A value-oriented approach to e-business process design". In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, Klagenfurt/Velden, Austria (Springer Verlag), pp. 390-403 http://www.cs.vu.nl/~gordijn/caise03.pdf

Graham, I., Henderson-Sellers, B. and Younessi, H. (1997) *The OPEN Process Specification* (Addison-Wesley)

Gu, A., Henderson-Sellers, B. and Lowe, D. (2002) "Web Modelling Languages: The Gap between Requirements and Current Exemplars". In *Proceedings of the 8th Australian World Wide Web Conference*, Gold Coast, pp. 362-375 http://ausweb.scu.edu.au/aw02/papers/refereed/lowe/index.html

Haverty, M. (2002) "Information Architecture Without Internal Theory: An Inductive Design Process". *Journal of the American Society for Information Science & Technology,* Vol. 53, No. 10, 839

Henderson-Sellers, B., Simons, A. J. H. and Younessi, H. (1998) *The OPEN Toolbook of Techniques* (Addison-Wesley)

Hennicker, R. and Koch, N. (2000) "A UML-based Methodology for Hypermedia Design". In *Proceedings of UML 2000 - The Unified Modeling Language. Advancing the StandardThird International Conference*, York, UK

Hennicker, R. and Koch, N. (2001) "Systematic Design of Web Applications with UML". *Unified Modeling Language: Systems Analysis, Design and Development Issues,* edited by T. Halpin (IDEA Group Publishing)

Isakowitz, T., Stohr, E. and Balasubramanian, P. (1995) "RMM: A Methodology for Structured Hypermedia Design". *Communications of the ACM*, Vol. 38, No. 8, 34-44

Kappel, G., Proll, B., Retschitzegger, W. and Hofer, T. (2001) "Modeling Ubiquitous Web Applications - A Comparison of Approaches". In *Proceedings of the International Conference on Information Integration and Web-based Applications and Services,* Austria

Kappel, G., Retschitzegger, W. and Schwinger, W. (2000a) "Modelling Customizable Web Applications - A Requirement's Perspective". In *Proceedings of the International Conference on Digital Libraries: Research and Practice*, Koyoto

Kappel, G., Retschitzegger, W. and Schwinger, W. (2000b) "Toward Modelling of Data Web Applications - A Requirement's Perspective". In *Proceedings of the Americans Conference on Information Systems*, Long Beach, CA

Kobbryn, C. (1999) "UML2001: A Standardization Odyssey". *Communications of the ACM*, Vol. 42, No. 10

Koch, N. and Kraus, A. (2002) "The expressive Power of UML-based Web Engineering". In *Proceedings of the Second International Workshop on Web-Oriented Software Technology (IWWOST2)*, Malaga

Lamar, L. (2001) "Introduction to a user interface design/information architecture process for Web sites". In *Proceedings of IEEE International Professional Communication Conference (IPCC 2001)*, Sante Fe, NM (IEEE)

Lieberman, B. (2001) "UML Activity Diagrams: Detailing User Interface Navigation" http://intranet.cefetcampos.br/portalProjetos/dirModSis/sitFic1061943804_diagAtiv2.pdf/render_raw_content

Lowe, D. (2002a) "Characterisation of Web Projects". In *Proceedings of the Eighth Australian World Wide Web Conference (AusWeb'02)*, Sunshine Coast, Australia (Southern Cross University), pp. 58-68 http://ausweb.scu.edu.au/aw02/papers/refereed/lowe2/index.html

Lowe, D. (2003) "Web Requirements: An Overview". *Requirements Engineering Journal*, Vol. 8, No. 2, July, pp. 102-113 (London: Springer-Verlag)

Lowe, D. and Eklund, J. (2002) "Client Needs and the Design Process in Web Projects". *Journal of Web Engineering*, Vol 1, No. 1, 23-36

Lowe, D. and Tongrungrojana, R. (2003) "WebML+ for Improving Conceptual Communication of Information Flows: An Empirical Study". In *Proceedings of the Third International Conference on Web Engineering (ICWE'03)*, Oviedo (Springer), pp. 218-231

Mandel, L., Koch, N. and Maier, C. (1999) "Extending UML to Model Hypermedia and Distributed Systems" http://projekte.fast.de/Projekte/forsoft/intoohdm/

OMG (2004) "OMG Unified Modeling Language Specification, Version 1.5". Object Management Group, Inc. http://www.omg.org/technology/documents/formal/uml.htm

OMG (2003) "Introduction to OMG's Unified Modeling Language". Object Management Group, Inc. http://www.omg.org/gettingstarted/what_is_uml.htm

Powell, T. A. (1998) *Web Site Engineering* (Prentice-Hall)

Pressman, R. (2001) *Software Engineering: A Practitioner's Approach* (McGraw Hill)

Rosenfeld, L. and Morville, P. (1998) *Information Architecture for the World Wide Web* (O'Reilly)

Schwabe, D. and Rossi, G. (1998) "Developing Hypermedia Applications using OOHDM". In *Proceedings of Workshop on Hypermedia Development Processes, Methods and Models (Hypertext'98)*, Pittsburgh

Schwinger, W. (2001) "Modelling Ubiquitous Web Applications". PhD Thesis, Department of Information Systems, University of Linz, Austria

Shelford, T. J. and Remillard, G. A. (2002) *Real Web Project Management: Case Studies and Best Practices from the Trenches* (Addison Wesley Professional)

Tongrungrojana, R. and Lowe, D. (2003) "WebML+: Connecting Business Models to Information Designs". In *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE)*, San Francisco (Knowledge Systems Institute: Skikie), pp. 17-24

Troyer, O. D. and Leune, C. (1998) "WSDM: A user-centered design method for Web sites". In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, April http://www7.scu.edu.au/programme/fullpapers/1853/com1853.htm

Vilain, P., Schwabe, D. and de Souza, C. S. (2000) "A Diagrammatic Tool for Representing User Interaction in UML". In *Proceedings of <<UML>> 2000: The Third International Conference on the Unified Modeling Language*, York, UK http://www-di.inf.puc-rio.br/schwabe//papers/UML2000.pdf

Ziegler, A. (2002) "Data Modeling/Object Modeling". ruby-talk email list, 19 December 2002 http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/59573

Links

Argus Centre for Information Architecture http://argus-acia.com/ia_guide/index.html

Appendix 1. WIED DTDs

WIED Document Type Definition

```
<?xml encoding="UTF-8"?>
<!-- ============================================================= -->
<!-- WIED Document Type Definition -->
<!-- -->
<!-- This DTD dictates the interface of elements involved in -->
<!-- information flow modeling of an organisational structure -->
<!-- -->
<!-- -->
<!-- Version 0.2r1 (July 30 2002) -->
<!-- ============================================================= -->
<!-- ============================================================= -->
<!-- WIED -->
<!-- ============================================================= -->

<!ELEMENT WIED ((ACTORUNIT | SUPPLIEDINFOUNIT | DERIVEDINFOUNIT)+)>
<!ATTLIST WIED
version CDATA #REQUIRED
modelName CDATA #REQUIRED
>
<!-- ============================================================= -->
<!-- ACTORUNIT -->
<!-- ============================================================= -->
<!ELEMENT ACTORUNIT (INCOMINGFLOW)*>
<!ATTLIST ACTORUNIT
```

```
id ID #REQUIRED
name CDATA #IMPLIED
role CDATA #REQUIRED
description CDATA #IMPLIED
>
<!-- ============================================================== -->
<!-- SUPPLIEDINFOUNIT -->
<!-- ============================================================== -->
<!ELEMENT SUPPLIEDINFOUNIT EMPTY>
<!ATTLIST SUPPLIEDINFOUNIT
id ID #REQUIRED
name CDATA #IMPLIED
supplier IDREF #REQUIRED
description CDATA #IMPLIED
type (transient | persistent) "transient"
>
<!-- ============================================================== -->
<!-- DERIVEDINFOUNIT -->
<!-- ============================================================== -->
<!ELEMENT DERIVEDINFOUNIT (DERIVATIONUNIT)>
<!ATTLIST DERIVEDINFOUNIT
id ID #REQUIRED
name CDATA #IMPLIED
description CDATA #IMPLIED
type (transient | persistent) "transient"
>
<!-- ============================================================== -->
<!-- DERIVATIONUNIT -->
<!-- ============================================================== -->
<!ELEMENT DERIVATIONUNIT (INCOMINGFLOW)+>
<!ATTLIST DERIVATIONUNIT
id ID #REQUIRED
name CDATA #IMPLIED
description CDATA #IMPLIED
type (auto | manual) "auto"
>
<!-- ============================================================== -->
<!-- INCOMINGFLOW -->
<!-- ============================================================== -->
<!ELEMENT INCOMINGFLOW EMPTY>
<!ATTLIST INCOMINGFLOW
from IDREF #REQUIRED
description CDATA #IMPLIED
>
```

Appendix 2. Example WIED model

The following document illustrates the formal WIED model (expressed in XML) of the Web-based email system.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WIED SYSTEM "WIED.dtd">
<WIED version="0.2r1" modelName="Web-BasedEmail">

<SUPPLIEDINFOUNIT id="Advertisement" supplier="AdCompany" type="transient" description="Advertisement provided by AdCompany and
displayed to User as part of Email"/>

<DERIVEDINFOUNIT id="AdDatabase" description="Database for storing Advertisement">
<DERIVATIONUNIT id="StoreAd">
<INCOMINGFLOW from="AdDatabase"/>
<INCOMINGFLOW from="Advertisement"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<ACTORUNIT id="Hotmail" role="ProvideAccesstoEmails"/>
<ACTORUNIT id="AdCompany" role="DistributeAd">
<INCOMINGFLOW from="Invoice"/>
</ACTORUNIT>

<SUPPLIEDINFOUNIT id="UserPolicy" supplier="Hotmail" description="ManuallyCreatedbyHotmail" type="persistent"/>

<ACTORUNIT id="User" role="ReadandSendEmail">
<INCOMINGFLOW from="UserInfoandProfile"/>
<INCOMINGFLOW from="DisplayedEmail"/>
<INCOMINGFLOW from="UsageInfo"/>
</ACTORUNIT>

<SUPPLIEDINFOUNIT id="Email" supplier="User" type="transient" description="Email message sent from User"/>

<SUPPLIEDINFOUNIT id="UserInfo" supplier="User" type="transient"/>

<DERIVEDINFOUNIT id="UserInfoDatabase" type="persistent">
<DERIVATIONUNIT id="StoreUserInfo">
<INCOMINGFLOW from="UserInfo"/>
<INCOMINGFLOW from="UserInfoDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="EmailDatabase" type="persistent" description="Database for storing incoming Emails">
<DERIVATIONUNIT id="StoreEmail" description="Store Incoming Emails into database" type="auto">
<INCOMINGFLOW from="Email"/>
<INCOMINGFLOW from="EmailDatabase"/>
</DERIVATIONUNIT>
```

```
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="DisplayedEmail" type="transient" description="DisplayedEmailContent">
<DERIVATIONUNIT id="CreateDisplayedEmail" description="Create Displayed Email by integrating from 3 sources of information">
<INCOMINGFLOW from="UserInfoDatabase"/>
<INCOMINGFLOW from="EmailDatabase"/>
<INCOMINGFLOW from="AdDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="UserInfoandProfile" type="transient" description="UserInfoandProfileandNumberofEmails">
<DERIVATIONUNIT id="CreateUserInfoandProfile">
<INCOMINGFLOW from="UserInfoDatabase"/>
<INCOMINGFLOW from="EmailDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>
```

```
<DERIVEDINFOUNIT id="UsageInfo" type="transient" description="UsageofStorageandQuota">
<DERIVATIONUNIT id="CreateUsageInfo">
<INCOMINGFLOW from="UserPolicy"/>
<INCOMINGFLOW from="EmailDatabase"/>
<INCOMINGFLOW from="UserInfoDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="Impression" type="transient">
<DERIVATIONUNIT id="SetImpression">
<INCOMINGFLOW from="DisplayedEmail"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="ImpressedAdDatabase" type="persistent">
<DERIVATIONUNIT id="SetImpressedAd">
<INCOMINGFLOW from="AdDatabase"/>
<INCOMINGFLOW from="Impression"/>
<INCOMINGFLOW from="ImpressedAdDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>

<DERIVEDINFOUNIT id="Invoice" type="transient">
<DERIVATIONUNIT id="CreateInvoice">
<INCOMINGFLOW from="ImpressedAdDatabase"/>
</DERIVATIONUNIT>
</DERIVEDINFOUNIT>
</WIED>
```