A Method for Analysing Web Systems Model Alignment

Andrew Bucknell[HREF1], Senior Research Assistant, Centre for Real-Time Information Networks[HREF2]; University of Technology, Sydney[HREF3]; PO Box 123 Broadway NSW 2007. Andrew.Bucknell@uts.edu.au

David Lowe[HREF4], Director, Centre for Real-Time Information Networks[HREF2]; University of Technology, Sydney[HREF3]; PO Box 123 Broadway NSW 2007. David.Lowe@uts.edu.au

Didar Zowghi[HREF5], Director, Requirements Engineering Laboratory[HREF6]; University of Technology, Sydney[HREF3]; PO Box 123 Broadway NSW 2007. Didar.Zowghi@uts.edu.au

Abstract

The introduction or modification of IT systems will often lead to the need for consequential changes to the associated business processes. This is particularly true for Web systems, where the system can be much more deeply intertwined with the business processes. The failure to identify these potential impacts early on can lead to major delays or cost increases in the development process, and subsequent redevelopment as the IT system and business processes are appropriately modified to bring them into alignment.

We believe that early modelling of the existing business processes, and their relationship to the IT systems, as well as the potential changes to these processes will allow the identification of misalignments in the model which are a reflection of likely misalignments which would exist in the actual processes if the changes were to be implemented. This then will allow rectification of the misalignment much earlier than is currently the case. It is worth noting that this does not necessarily require new modelling notations, but rather the use of existing notations in new ways combined with appropriate analysis tools.

This research aims to demonstrate that this identification of misalignments can be achieved using existing modelling notations by applying algorithmic analysis to the information about the business process captured in the model. This can be achieved by integrating this analysis with modelling tools, and hence supporting the identification of likely flaws in a proposed changed business process before that process is actually implemented. This paper presents a proof of concept that demonstrates an approach to analysing the properties of model elements using a constraint-based rule system in conjunction with an existing widely-used modelling tool.

Introduction

Current software development is often characterised by significant early uncertainty in system scope, particularly where the Web is leveraged in supporting changes to business processes. This uncertainty can then lead to poor alignment between business processes and IT systems, substantial ongoing system redevelopment, and client and customer dissatisfaction with the resultant systems. The lack of appropriate approaches to address these problems leads to significant costs in web development projects [Lowe 2000], which is a significant business activity in its own right [Houghton 2005], with cost overruns being a significant waste of resources.

The approach to identifying mis-alignments described in this paper is being developed to support research which investigates an innovative approach to reducing this uncertainty and the resulting volatility, and through this support a more rapid resolution of the development scope for software systems. Specifically, the research will utilise recent progress in the development of high-level modelling approaches [Lowe 2004, Tongrungrojana 2004a] which more effectively link system information management and functional behaviours with business processes to enable the automated identification of potential discords between the software system being developed and the organisational context in which the system exists. These discords arise when aspects of the (proposed) system or business processes are changed without appropriate consideration of the impacts on the complex inter-relationships which exist within the composite software/business environment.

Process Modelling

This project will build upon our earlier research in several key areas. The first area, related to the earlier stages of the Web development process, includes research into Web characterisation models [Lowe 2000], development processes [Kong 2005b], requirements volatility [Zowghi 2000]. Web Impact analysis [Lowe 2003, Yusop 2003, Yusop 2005] and – most recently – our work on the role of issue resolution in supporting the determination of system scope in Web projects. This last body of research is crucial in the context of this project. For the commercial Web projects which we studied, once an initial project brief had been established the key trigger for almost all subsequent adjustments to the project scope was the resolution of issues. These issues related to the way in which the system environment impacts on, and is impacted (or changed) by the introduction of the system. This leads us to an approach to scope refinement which will be based on the dualistic modelling of the system and environment as they currently exist prior to the development (an as-is view), and as they are desired to be (a to-be view). This approach to the problem emerges from the area of impact analysis [Aversano 2005].

Information Flows

The second area of our previous research which feeds into this project is our development of high-level information flow models [Tongrungrojana 2004a]. Whilst the detailed behaviour and low-level information design of a Web system tends to be extremely volatile during (and usually after) the system development the flow of information between the system and its environment tends to be much more stable. We subsequently showed [Tongrungrojana
2004b] that a representation based on high-level information flows provided a clearer basis for system design, and—most significantly—facilitated identification of ways in which a proposed design change may result in impacts upon the systems environment. Further, the information flow models, when combined with conventional work flows, appeared to capture a key set of interfaces between the system and the organisation in which it is embedded—those interfaces which, when changed, lead to changes in the scope, and vice versa. The interfaces do not uniquely and completely define the system, but they do appear to provide a source of crucial information for identifying and resolving the discordances that were discussed in the background section above, and which are the focus of this research.

Discordances

Merging the above two threads of previous research provides a clear approach to supporting the automated identification of those discordances between an IT system and its environment that have the potential to affect decisions on the system scope and its corresponding boundary. These identified discordances can then be used to raise issues within a linked issue-tracking system. The issues, when resolved, will allow a progressive clarification and elaboration of the agreed system scope (as well as the concomitant changes to the associated organisational workflows and business processes).

The approach will be based on the development of a modelling notation which links information flows, functional boundaries and work flows—and which supports the development of models that initially represent the current (as-is) domain. The models are then progressively adjusted to show the incorporation of variations on the proposed IT system, with discordances being automatically identified and issues being raised with the developer as the models are modified. The value of a visual notation that can be evolved can be seen in the success of approaches such as Threat Model Analysis (TMA). Visualising the system helps to make the boundaries more readily apparent, and thus less likely to be overlooked. The existence of the model also allows the impact of changes in the relationships between elements on the security threats to the system to be assessed and addressed. A prototype tool which forms part of this research will allow evaluation of the effectiveness of the models and algorithms in managing the construction and evolution of a system, and identifying boundary conditions that require negotiation between the client and the developer. We believe this approach will lead to a much more rapid convergence of the agreed system scope and a substantial reduction in overlooked adverse misalignment between the system and its environment.

Understanding Discordances

As was mentioned above, a key characteristic of much software systems development is early uncertainty in the system scope. There exists a significant and growing body of research into the early stages of the development cycle when the development scope is nominally resolved—mostly focussing on domain modelling and requirements capture and analysis. Most of this research however assumes an initially unknown but largely invariant system scope which simply must be elicited and analysed. Both research and practice often overlooks the complex interdependencies through which the emerging definition of a system can directly or indirectly affect the environment in which that system exists, and thereby create a feedback loop which leads to subsequent changes in the scope of that system [Yusop 2005]. This type of interdependency between system and environment is most common where part of a business process is being fundamentally changed or replaced by a software system. Whilst this characteristic of system development is not uncommon in most (if not all) software systems, recent technologies, and especially web-based systems, are exemplars of where the scale of the feedback mechanism makes addressing it early an imperative.

Example

Before we consider existing research which is relevant, let us illustrate the above problem with a simplified illustrative example. Consider an existing business process (variants of which exist in many small businesses) that involves casual employees completing a paper-based timesheet at the end of each week which is subsequently checked by a supervisor. The manual process is shown in Figure 1 using a simplified process modelling notation. The following figures adopt a simplistic notation derived from Conalians Web Extensions for UML [Conallen 1999] and also drawing on our previous work on modelling information flows in web-systems [Bucknell 2007 Tongrungrojana 2004a]. The model is created using a modelling tool to capture the existing process. In this model, all the documents are paper based, and the activities assume that the input and output documents are paper.
The initial proposed scope of the system to be implemented is shown in Figure 2. The shaded ellipse indicates that the documents and activity it encloses will be implemented as a software system. In this case, we are proposing that employees will be able to use an electronic system to enter their timesheets. The grouping mechanism implemented by the shaded region is used to represent the relationship between the business workflows and the existing (or anticipated) software systems. Blank timesheets and completed timesheets will be electronic, and the Complete Timesheet activity will have electronic inputs and outputs.

The scope of the proposed system shown in Figure 2 gives rise to a discordance. The Completed Timesheet is now electronic as part of the proposed system, while the Review Timesheet activity expects an input document that is paper. It is important to note that in this work discordances do not arise from errors in the model, but rather from models that if implemented would result in organisational dysfunction. The model shown in Figure 2 is correct and accurately represents the organisation as it would be if the system were deployed. In this example, implementing the system as shown in Figure 2 would mean that the Employer would not be able to review the timesheets. If the proposed system were implemented and deployed, then the organisational process would be interrupted. By identifying this discordance in the modelling phase it is possible for the system designers to take steps to resolve this discordance.

Once a discordance has been identified, the model must be restructured to resolve it. Our method does not prescribe how discordances should be resolved. The purpose of our approach is to support system designers in identifying these potential sources of organisational dysfunction so that they can address them before the model is implemented. Figures 3 and 4 show alternative approaches to resolving the discordance identified in Figure 2. In the first approach, shown in Figure 3, the discordance is resolved by expanding the scope of the system to include the Review Timesheet activity and the Reviewed Timesheet document. This means that all elements of the model are now part of the system and there is no mis-alignment in how the elements relate to one another. Thus, the discordance is removed.
Figure 3 – Scope of Proposed Timesheet System With Discordance Resolved

In the second approach, shown in figure 4, the functionality of the system to be implemented is extended without extending the scope of the system to be implemented with regard to the existing organisational process. The Review Timesheet activity remains a manual process. The organisational process is extended to include an activity where an Assistant prints out a copy of the timesheet for review by the Employer.

Figure 4 - Alternative Scope of Proposed Timesheet System With Discordance Resolved

As this example shows, when we move from a system as it currently exists (the as-is view of the business/system) to the definition of a potential new or changed system (the to-be view of the business/system) we introduce potential discordances between the system and the business processes to be supported by the new system. Resolving these discordances leads either to changes in the scope of the system, or changes in the associated business processes – either of which can lead to further changes to either or both. Much of the complexity of the early stages of project scoping resolves around understanding and negotiating these changes and defining the boundary of what functionality should “the system to be” support and which it should not. Whilst this form of scope resolution is well accepted and understood, it is our contention that it is invariably overlooked, and there has been little research specifically addressing how it can be most effectively managed. We contend that by undertaking richer forms of the modelling shown in the above examples, based on a merger of existing process modelling techniques and our own information flow modelling formalism [Tongrungrojana 2004b], it is possible to automatically identify potential discordances early in the project scoping / specification, and to then raise these as development issues that need to be resolved – thereby leading to more rapid convergence onto an agreed system scope which is integrated with the modified organisational workflows and business processes.

Concepts

As mentioned above, there has been substantial research over the last 30+ years, focusing on the early stages of software systems development and the relationships between software systems and business processes. Requirements Engineering (RE) is an active research area which focuses on approaches to capturing, analysing and modelling requirements for software systems. The majority of research in this field assumes a fixed, though initially unknown, scope which must be discovered, analysed and documented. Where requirements are recognised as
varying (or volatile [Zowghi 2000]) this is usually attributed to uncertainty on the part of the client [McKeen 2003] or changes occurring in the domain independently of the introduction of the system-to-be. Very little research has considered the way in which the introduced system itself can lead to changes in the domain – and hence create a positive feedback loop leading to changes in the system. Our earlier research, which has had a specific focus on Web systems development, has shown that often Web systems and the organisation domain in which the systems exist co-evolve, with the nature and extent of this evolution often not being clearly understood until early design prototypes are available [Kong 2005a, Kong 2005b].

In many respects, this research is also closely related to work on IT-business alignment, which focuses on ways in which a software system can be most appropriately designed to seamlessly integrate with, and support, existing or proposed business activities. Of particular interest is research on strategic alignment [McKeen 2003]. Research has shown that strategic alignment can have substantial positive impact on business performance [Croteau 2004].

Whilst the desired end result is similar (the absence of discords between the system and the business processes – or the business objectives which are supported by those processes), the focus of work on IT-business alignment is typically on how to ensure that software systems appropriately support a given set of business objectives, rather than the identification of specific aspects where an existing business process requires modification as a consequence of the introduction or modification of a software system to which it interfaces. Furthermore, the research in this area has not paid due attention to the nature and extent of impact that the "system-to-be" may have on the corresponding business rules that govern business processes and workflows within organisations.

Similarly, work in areas as diverse as soft-systems methodologies (SSM) [Checkland 1990], problem frames [Jackson 2000] and COTS (Commercial Off-The-Shelf) development [Torchiano 2004] also provides insights into the interdependence of software systems and the organisational processes within which they are embedded. For example, rich pictures – a tool used within SSM and elsewhere – can be used to understand the relationships between software systems and the contexts within which they exist. Nevertheless, again, these techniques are useful in supporting effective system design, but not in identifying specific points of discordance. This identification is typically assumed to occur as a natural consequence of the system design process, and hence has lacked any focus as a research topic. Indeed, this assumption is partially true – the discords do indeed become obvious – but often not until later in the development cycle, well after design or indeed implementation has commenced and scoping contracts have been agreed upon and signed off.

Given that there is much work on what defines a software system effectively aligned with organisational processes or business goals, but not on specific techniques for identifying particular points where they are potentially misaligned, the obvious question is how such discords can be discovered as early in the development as possible and what strategies could be employed in early resolution of issues arising from the identification of these discords. Answering these questions is the core of this research project.

Our particular approach to answering this question emerges from the convergence of two of our earlier research contributions: investigation of the ways in which issues that emerge during software system design (and Web development in particular) have influenced the developers understanding of the system scope [Lowe 1998, Yusop 2005]; and the potential role in web system design of a high-level information flow model [Tongrungrojana 2004a]. Taken together, these two areas of work indicate the importance to defining system scope of understanding the flow of information into and out of a system, especially when coupled with an equivalent process flow. These two areas of research have informed our development of a method to identify discords in system models.

Identifying Discords

The premise of this work is that identifying discords during the design phase reduces the costs associated with failing to identify the required scope of implementing or modifying a software system. In order to support the identification of discords we propose a method based on analysis of attributes of the modelling entities being used to represent a system. Many modelling notations exist that allow business process flows to be modelled, such as BPMN, UML, and IDEF. Similarly, there are many notations that can be used to model web systems [Conallen 1999, Tongrungrojana 2004a]. In this work we aim to build on top of these techniques to provide a mechanism for identifying discords.

A discordance arises when a model describes a system that, if implemented, would result in a dysfunctional business process. The research question investigated in this work asks how these discords can be identified. We theorised that discords could be identified by analysing attributes of the model against a set of constraints. The constraints specify the conditions under which a model is in concord. Any violation of these constraints represents a discordance that should be resolved before the system is implemented. To investigate the validity of this approach, we have developed a simple system that demonstrates how attributes and constraints can be applied to the kinds of problems represented by the timesheet system described above.

Our approach has two key requirements. First, we need a way to specify the constraints that apply to a model in order for it to be concordant. Second, we need a way to add information about the attributes to a model. Both of these requirements must be met in a way that supports algorithmic analysis of constraints.

To state the first requirement another way, we need a way to describe rules about a model, and we need a way to describe the attributes that apply to a model. We describe the rules about a model using a notation. Notations are expressed in XML using the DTD shown in Figure 5.

<DOCTYPE notation [
Figure 5 - Notation DOCTYPE

The Notation DTD extends a simple Entity-Relationship meta-model that we developed previously [Bucknell 2007]. In this meta-model, we specify that a notation describes a set of entities and a set of relationships between these entities. Each entity can have a set of attributes associated with it. In order to support identification of discorances we have extended this model to include constraints. A constraint is defined on a relationship, and specifies which attributes of the entities at the endpoints of the relationship must be aligned. An instance of this DTD that represents the notation used in Figures 1-4 above is shown in Figure 6.

<notation>
  <entities>
    <entity type="Actor"/>
    <entity type="Activity">
      <attribute name="inputmedia"/>
      <attribute name="outputmedia"/>
    </entity>
    <entity type="Document">
      <attribute name="media"/>
    </entity>
  </entities>
  <relationships>
    <relationship type="Invoke" from="Actor" to="Activity"/>
    <relationship type="Input" from="Document" to="Activity">
      <constraint fromattr="media" toattr="inputmedia"/>
    </relationship>
    <relationship type="Output" from="Activity" to="Document">
      <constraint fromattr="outputmedia" toattr="media"/>
    </relationship>
  </relationships>
</notation>

Figure 6 - Notation Instance

To satisfy the second requirement, we need a way to represent models in a way that captures the attributes that are used in the constraints that determine if a model has any discorances. The model structure we have chosen is based on the entity-relationship modelling we have defined previously for representing generic process models [cite ICWE again]. The DTD describing how models can be represented is shown in Figure 7. An instance of this DTD representing the Existing Timesheet System shown in Figure 1 is shown in Figure 8.

<!DOCTYPE model [
  <!ELEMENT model (entities|relationships)>]
<!ELEMENT entities (entity)+>
<!ELEMENT relationships (relationship)+>
<!ELEMENT entity (attribute)>
Figure 7 - Model DOCTYPE

```xml
<model>
  <entities>
    <entity eid="1" type="actor" name="employee"/>
    <entity eid="2" type="activity" name="submit timesheet">
      <attribute name="inputmedia" value="paper"/>
      <attribute name="outputmedia" value="paper"/>
    </entity>
    <entity eid="3" type="document" name="blank timesheet">
      <attribute name="media" value="paper"/>
    </entity>
    <entity eid="4" type="document" name="filled timesheet">
      <attribute name="media" value="paper"/>
    </entity>
    <entity eid="5" type="actor" name="employer"/>
    <entity eid="6" type="activity" name="review timesheet">
      <attribute name="inputmedia" value="paper"/>
      <attribute name="outputmedia" value="paper"/>
    </entity>
    <entity eid="7" type="document" name="reviewed timesheet">
      <attribute name="media" value="paper"/>
    </entity>
  </entities>
  <relationships>
    <relationship rid="1" type="invoke" froment="employee" toent="submit timesheet"/>
    <relationship rid="2" type="input" froment="blank timesheet" toent="submit timesheet"/>
    <relationship rid="3" type="output" froment="submit timesheet" toent="filled timesheet"/>
    <relationship rid="4" type="input" froment="filled timesheet" toent="review timesheet"/>
    <relationship rid="5" type="invoke" froment="employer" toent="review timesheet"/>
    <relationship rid="6" type="output" froment="review timesheet" toent="review timesheet"/>
  </relationships>
</model>
```

Figure 8 - XML Instance of Existing Timesheet System

The model DTD is quite similar to the notation DTD but the semantics of the two DTDs are quite different. Essentially, an instance of the notation DTD enumerates what things can be modelled, while an instance of the model DTD enumerates what things are modelled. The notation expresses the constraints on attributes that are used to identify discords, while the model captures the values of the attributes that are evaluated through the constraints. The notation and the model provide a way of expressing the rules and values that are used to identify discords in a model.

Application

With the model and notation concepts in place, the next step is to define and implement logic that actually uses the information in the notation and the model to evaluate whether any constraints are violated. To demonstrate that this
approach can be used we have developed a constraint analysis engine. This engine takes as its inputs an instance of the Model DTD and an instance of the Notation DTD. For each relationship in the Model, it iterates over the constraints that the Notation defines for that relationship. If the value of the from attribute is not equal to the value of the to attribute, then a discordance is identified. The details of a discordance are sent to an output log specifying the relationship for which the constraint was violated and the attribute that did not match. The tool does not suggest possible solutions. It is up to the analyst to update their model and then perform the constraint analysis again.

To investigate the usefulness of this approach to identifying discordances, we have developed a tool that integrates the Visio drawing component with an analysis engine based on the notation and model definitions we have outlined above. Visio models can be loaded into the tool and then analysed against a selected notation. The analysis is performed by converting the Visio Object Model for the diagram into an instance of the Model DTD. Custom Properties of the shapes in the Visio diagram with names corresponding to entity attribute names specified in the Notation DTD instance are extracted into the Model. Therefore the approach we use can be applied to any Visio diagram provided the custom properties have been added to the shapes and a Notation instance is available. Once the analysis has been performed, any discordances are reported as constraint violations in a log window. A screenshot of the simple UI we have implemented at this time is shown in Figure 9.

![Figure 9 - VisUI Modelling Interface](image)

Conclusions and Further Work

Our work so far has shown that discordances in a system model can be identified using attributes and constraints. We have done this by describing and implementing a simple model and meta-model that allows us to apply attributes and constraints. These attributes and constraints use a simple constraint semantic based on attribute equality to identify discordances. We have also shown that this approach can be integrated with an existing modelling tool, through our implementation of VisUI.

Building on this we have two areas of further research. The first area is to develop a richer set of constraint semantics. Presently discordances are limited to inequality of attributes. We will be investigating ways to express other constraint rules such as conditional equality, conditional inequality, set membership and pattern matching. This will require extending the Notation DTD and implementing a richer set of rules in the constraint engine.

The second area of investigation is to use our method for identifying discordances in real-life situations to investigate whether our approach is useful. This will involve integrating the use of this tool with existing development methods in live development environments to investigate whether the ability to identify discordances early in the development lifecycle yields the kinds of benefits we expect it will.

References


Hypertext References

HREF2 http://crin.eng.uts.edu.au/
HREF3 http://www.uts.edu.au
HREF5 http://www-staff.it.uts.edu.au/~dil达尔/
HREF6 http://research.it.uts.edu.au/re/index.html

Copyright

<Andrew Bucknell, David Lowe, Dider Zowghi>, © 2008. The authors assign to Southern Cross University and other educational and non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also
grant a non-exclusive licence to Southern Cross University to publish this document in full on the World Wide Web and on CD-ROM and in printed form with the conference papers and for the document to be published on mirrors on the World Wide Web.