Predicting Change Impact in Architecture Design with Bayesian Belief Networks

Antony Tang Yan Jin Jun Han
Faculty of ICT
Swinburne University of Technology
Melbourne, Australia
E-mail:{atang,yjin,jhan}@ict.swin.edu.au

Ann Nicholson
School of Computer Science and Software Engineering, Monash University
3800, Victoria, Australia
E-mail:annn@csse.monash.edu.au

Abstract

Research into design rationale in the past has focused on the representation of reasons and has omitted the connections between design rationales and design artefacts. Without such connections, designers and architects cannot easily assess how changing requirements or designs may affect the system. In this paper, we introduce a model called Architecture Rationale and Element Linkage (AREL) to represent the causal relationship between architecture elements and decisions. We further model AREL as a Bayesian Belief Network (BBN) to capture the probabilistic relationships between architecture elements and decisions in an architecture design model. Such probabilistic modelling enables architects to quantitatively predict and diagnose impact of change when part of the requirements or designs are changing. Using a partial design of a cheque image processing system, we illustrate how AREL is used to represent the decision model and how BBN is used to predict and diagnose change in the architecture design. We use a UML tool to capture the AREL model and a BBN tool to compute the probabilities of change impact.

1. Introduction

Design rationale plays an important role in the design and subsequent maintenance of large and complex systems. It supports reasoning and it captures why design decisions are made [18]. Design rationale has many benefits. It can be used to verify and trace design as well as support system enhancements. Despite its usefulness, design rationale is often not documented and the knowledge is evaporated or eroded after the design is completed [2, 21]. Without such knowledge, change impacts cannot be assessed accurately. This issue is accentuated when the designs that cater for the non-functional requirements cross-cut and are intertwined within the system. These types of requirements or designs are typically architectural in nature and have a profound impact on projects [7].

Architecture design deals with the *load-bearing* issue [21] or fundamental organisation of a system [14]. Changes in the architecture after its initial design can have a major impact on the system. Since design rationales of architecture decisions are often not documented, the knowledge about the design decisions such as assumptions, constraints and design alternatives are not communicated or retained in a systematic way. Problems would arise when the expertise is no longer available [13] or the knowledge is forgotten [28].

The first step to resolving this issue is to record the tacit knowledge about the architecture decisions in a structured way and enable these decisions to be traced. A graphical representation of the decision relationship will provide an explicit model of causality between decisions and related architecture elements. Additionally, we need to address how the captured knowledge could be retrieved. If a designer is not familiar with the architecture design, it is not obvious which areas of the system are likely to be affected by enhancements unless all related areas are studied thoroughly. The learning curve can be costly and time consuming. If the likelihood of change impact cannot be quantified, architects would have to make guesses on which areas are most likely to be affected without any evidential support. Therefore, another step is required for predicting and diagnosing change impacts in architecture designs. This paper introduces a method which will allow architects and designers to capture the knowledge about design decisions and use probability calculus to quantify the likelihood of change impact.

In our earlier work, we have proposed Architecture Rationalisation Method (ARM) [27] as a process to

Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)
0-7695-2548-2/05 $20.00 © 2005 IEEE

capture architecture rationale for decision traceability and decision verifiability. The context of the architecture design we consider follows the IEEE 1471-2000 standard where multiple viewpoints are required [14]. These viewpoints typically include requirements, information, computation and engineering. Architecture decisions are the results of this multitude of influences across views. When a decision is made, an Architecture Rationale (AR) captures the rationale of the design decision and links the rationale to the related *architecture elements* (AE) such as requirements, design components, data models or implementation components. This mechanism provides the traceability of design rationales to different parts of the architecture design.

In this paper, we refine the graphical representation of ARM to allow architects and designers to apply reasoning based on those decisions. We are inspired by Fenton [10] to use Bayesian Belief Networks (BBN) [15]. BBN can represent causal relationships by using probabilities for reasoning. This paper makes the following contributions:

- We provide a graphical model to represent the causal relationship between architecture decisions and architecture elements. This model is called the Architecture Rationale and Element Linkage (AREL).
- We represent the AREL model as a BBN. It captures the causal relationship between architecture elements and architecture design decisions using probabilities. This allows architects to diagnose and predict change impact in an architecture model based on probability theory.

Section 2 of this paper describes the background of related research. Section 3 provides a definition for the graphical representations of architecture decisions. Section 4 describes how BBN can represent architecture decision networks for predicting and diagnosing change impacts. We discuss tool support in Section 5. After discussing the potentials and limitations of this work in section 6, we conclude this paper in Section 7.

2. Background and Related Work

It is widely accepted by the industry standard [14] and the software architecture community [29] that design rationale is an important part of architecture design. In an empirical study [3], it was shown that design rationale can help designers to work faster and better in dealing with change requests. Despite its importance, the capture and documentation of design rationale is often omitted in practice [2]. Argumentation-based design rationale approaches attempt to tackle this issue by capturing design deliberation using different methods [5, 16, 17]. The argumentation-based rationale system such as gIBIS [5] represents issues, positions and arguments in a graphical way to form a network of nodes to deliberate the design process. QOC [17] uses questions and options to represent problems and the alternatives. It provides a representation of a reasoning structure. DRL [16] uses a language to represent the process of obtaining design rationale. These approaches facilitate issue-based requirements refinements by exploring typed relationships between issues, positions and arguments. The requirements or designs involved in the decision are not linked to the decision at a granularity where cross tracing between decisions and design artifacts can be performed easily [25]. We argue that it is the missing relationship between decisions and their affected design artifacts that prevents these models from being practical [24].

Other approaches such as Architecture Frame [23], KAOS [8], non-functional requirement refinement process [9] and non-functional requirement framework [4] provide various ways to decompose or refine requirements to build designs. All these structured approaches have limited support for design rationale: (a) they do not record design rationale in their models; and (b) they do not provide a way to carry out trade-off analysis amongst alternative designs. Ramesh and Jarke [22] identified different link types to deliberate design and rationale. This model serves to trace and explain designs but it does not predict change impacts.

Bosch [2] argues that design decisions can be cross-cutting and intertwined, a first-class representation of design decisions is therefore required to reduce the high maintenance costs. We have developed ARM [27] to focus on architecture design rationalisation and its representation. ARM enables architecture rationale, or AR, to be captured and recorded during architecture design. It assumes the use of architecture viewpoints to organize requirements, information model, software design and implementation in an architecture design [14]. AR captures quantitative rationale and qualitative rationale. The qualitative rationale captured in AR is an enhancement of what is in ATAM [1]. We provide three improvements to argumentation-based reasoning systems in our work on ARM and here: (a) content improvement - capture qualitative design rationale and quantitative rationale; (b) representation improvement - link directly between design artefacts and design decisions; (c) retrieval and analysis improvement – provide probability-based information to predict and diagnose change impact in architecture models.

Fenton [10] believes that management decision support tools must be able to handle causality, uncertainty and combining different (often subjective) evidence, and suggests a solution based on BBN. BBN have been used in many applications (see [15] for a recent survey) including a causal model to detect interactions that are prone to human errors [12]. In our case, we apply the properties provided by BBN to quantify the strength of the relationships between design decisions and design elements. This allows us to predict and diagnose the change impacts.

3. Architecture Rationale and Causality

Design reasoning and architecture rationale capture the tacit knowledge about the design. This knowledge enables designers to trace and understand reasons about a design for making further decisions. ARM uses architecture viewpoints and UML to support design rationale modelling. Graphical representation in UML enables ARM to provide associations between *architecture elements* (AEs) and architecture rationales (ARs). AEs are comprised of requirements, designs or implementation artefacts in different architecture views. ARs are UML objects that capture the reasons (qualitative and quantitative) of why a design decision is made at that point. Reasons are qualified by the assumptions, constraints, arguments and tradeoffs of the decision and are also quantified by the costs, benefits and risks about the decision [27]. The associations between ARs and AEs provide the traceability of causal relationships between them.

3.1. Architecture Rationale and Architecture Elements Linkage

Each decision in ARM has a rationale AR that relates the input AEs to the resulting AEs. The input or source AEs are the causes of the decision. The result or target AEs are the effects of the decision. The linkages from the source AEs to the AR are converging and those from the AR to the target AEs are diverging. In any architecture design, many decisions are made and therefore many interconnected ARs and AEs are formed. These interconnected linkages are represented in the Architecture Rationale and Element Linkage (AREL) model.

An example of an AREL model is shown in Figure 1. This is a high level design of a reporting sub-system. Functional requirement *Sales Report* and non-functional requirement *Response Time* influence decision *AR1* such that the *Sales* table has to be de-normalised and a *Generic Reporting Function* is used to meet performance requirements. Using *SQL Lookup* table to select SQL statements, a *Generic Reporting Function* that formats the report and a *de-normalised Sales Table*, they drive the *AR4* decision that embedded SQL statements are required. This process continues as architects make design decisions using requirements, design principles, data models, design models or implementation artefacts as inputs, and further refines them into finer detailed designs.

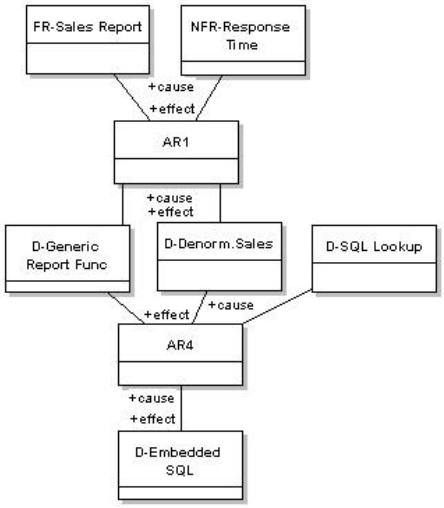

Figure 1 – An AREL Diagram

The representation of the model using UML would facilitate adoption by the software industry. The following definition of the AREL model formalises the relationship between ARs and AEs.

Definition 1. An *Architecture Rationale Element Linkage* (AREL) model is a tuple (*AE, AR, PL, SL*), where *AE* is a set of nodes representing architecture elements, *AR* is a set of nodes representing architecture rationales, $L = (AE \times AR) \cup (AR \times AE)$, and $PL \subseteq L$ and $SL \subseteq L$ *are disjoint sets of* directed links between the nodes denoting primary and secondary links, respectively, such that

(1) all rationale nodes must be associated by primary links with at least one cause and one effect: $\forall r \in AR, \exists e, e' \in AE$ such that $(e, r), (r, e') \in PL$;

(2) no subset of primary links in *PL* form a directed cycle;

Two kinds of unidirectional links are identified above: primary and secondary. An AR is connected to a minimum of two primary links, one from a cause AE and one to an effect AE. These primary links then represent the primary cause and effect of the rationale. The links shown in Figure 1 are all primary links.

Clause 2 of Definition 1 specifies that insertion of a primary link is not allowed if it would result in a directed cycle of primary links. Essentially, this means that primary links maintain the integrity of the causal modelling whereby something cannot cause itself, directly or indirectly. Primary links are used in the causality analysis described in Section 4.

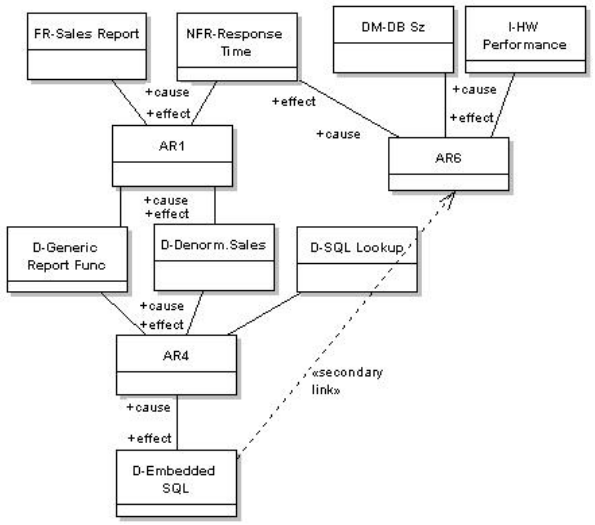

Figure 2 – An Example of Secondary Link

Secondary links are used to capture non-causal relationships for documentation purposes, but are differentiated from primary links so as not to affect the causal modelling integrity of AREL. In Figure 2, if the architect finds that the *Response Time* requirement cannot be met by the *Embedded SQL* design, the architect may decide that this requirement has to be changed. The fundamental causes to change the *Response Time* requirement are due to the processing constraints of the hardware and the increased size of the database. Though *Embedded SQL* design is not the actual cause to compromise the performance requirement, the lack of performance is exhibited through it. A secondary link between *Embedded SQL* and *AR6* can be used to depict this knowledge.

3.2. Avoiding Cyclic Decision in AREL Models

During the design process, architects may inadvertently cause a reasoning cycle to form as design rationales are incrementally constructed. This could happen when different teams of architects are constructing the design simultaneously. Examples of such cyclic decision rationales are shown in Figure 3a and Figure 3b. In Figure 3a, if *AR3* was to be created to link to *AE1*, this would create a directed cycle in the graph. Similarly, if *AR5* was modified and *AE10* was used as a cause in Figure 3b, it would create a directed cycle as well. These cases are illegitimate by the definition of AREL (section 3.1).

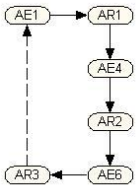

Figure 3a – AE Cyclic Case

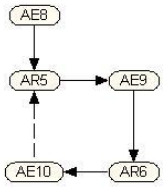

Figure 3b – AR Cyclic Case

There are two issues of a cyclic decision rationale: (i) it is ambiguous to determine which architecture element in the chain of reasoning is the primary cause in the directed cycle; and (ii) directed cycles mean that an AREL model cannot be represented as a BBN, which must be directed acyclic graphs for their belief updating algorithms to work. To prevent cycles from occurring in AREL, tool support is needed to detect that a directed cycle is being formed and disallow the primary link that would complete the directed cycle. In a tool implementation, architects would be warned about the logical problem and the resolution is by ways of human reasoning to modify the structure of the architecture to remove such design conflicts.

4. AREL and Bayesian Belief Networks

AREL is a model of causes and effects between architecture elements (AEs) and architecture rationale (ARs). Bayesian Belief Networks (BBNs) [15] is a well established method in the Artificial Intelligence community for reasoning under uncertainty, using (i) a graphical structure to represent causal relationships and (ii) probability calculus to quantify these relationships and update beliefs given new information. Representing an AREL as a BBN enables prediction and diagnosis of change impact to the system when certain elements of the architecture have changed.

A BBN is a graph with edges connecting nodes with no directed cycles (i.e. a directed acyclic graph). Its nodes represent random variables and its edges represent direct dependencies between variables. For our modelling purposes, we are interested in the causal relationships between variables representing architecture design rationales and elements. Nodes without parents are called *root nodes* and have an associated prior probability distribution. Each node with parents has a conditional probability table (CPT), which, for each combination of values of the parents, gives the conditional probability of its takings these

values. Thus the CPT can be considered to quantify the strength of the causal relationships. Users can set the values of any combination of nodes in the network and this newly inserted evidence propagates through the network, producing a new probability distribution over all the variables in the network. There are a number of efficient exact and approximate inference algorithms for performing this probabilistic updating [20], providing a powerful combination of predictive, diagnostic and explanatory reasoning. The following sections explore how AREL models can be represented by a BBN to provide causal reasoning under uncertainty for change impact analysis. The reader is referred to [15] for fundamentals of BBN.

4.1. Nodes: Representing Architecture Decisions and Elements

Each AE and AR node in an AREL model is represented by a discrete node in the BBN with two states. The probability contained in a state either measures the extent of dependencies (for non-root nodes) or the likelihood of change (for root AE nodes). The two states of an AE node represent whether or not the AE node is stable or likely to change. Its two states are:

- *Stable* – AE is not subject to change
- *Volatile* – AE is subject to change

The two states of an AR node represent the validity or otherwise of the rationale of the decision that correspond to AR node. Its two states are:

- *Valid* – the rationale of the decision is valid
- *Invalid* – the rationale of the decision is invalid

In the BBN representation of AREL, only AEs can be root nodes. AE root nodes are causes of the first-level architecture decisions. As such, they are either requirements (functional or non-functional), basic designs (such as standard design templates) or system constraints. There is no limit to how many root nodes can exist in an AREL model.

4.2. Edges: Representing Causal Relationships

Each primary link in an AREL model is represented by an edge in the BBN. As AREL primary links are defined to preclude directed cycles, this means that the corresponding BBN is valid, i.e. it is a directed acyclic graph. When an AREL model is formed, it is based on interconnected nodes with the fundamental structure of $AE_i \rightarrow AR_j \rightarrow AE_k$. This structure represents the unidirectional causal relationship between the nodes starting from AE_i. Figure 4 shows the three basic forms of relationships in the BBN model:

(A) AEs as causes of an AR;
(B) AEs as effects of an AR;
(C) multiple ARs as causes of an AE.

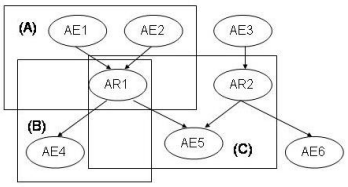

Figure 4 - Representing AREL in BBN

4.3. Probabilities: Quantifying the Causal Relationships

Each node in a BBN has either an associated prior probability (if it is a root node) or a conditional probability table. The BBN probabilities in the AREL context measure the extent in which an architecture element influence the validity of a decision, or the extent in which a decision may change a resulting architecture element. Prior probabilities assigned to the AE root nodes indicate the likelihood of change of these primary requirements. All AR nodes and all non-root AE nodes in the BBN representation of AREL are assigned CPTs. CPTs quantify the causal dependency on their parents. These probabilities are assigned by the architects using past organisational experience and assessment of current situation.

Given the fundamental structure of $AE_i \rightarrow AR_j \rightarrow AE_k$, AE_i is the root nodes with an assigned prior probability $P(AE_i)$. The probability of AR_j is conditionally dependent on the event that AE_i has occurred, represented by $P(AR_j \mid AE_i)$. The probability of AE_k is conditionally dependent on the event that AR_j has occurred, represented by $P(AE_k \mid AR_j)$. The joint probability is therefore the product of multiplying probability tables of AE_k, AR_j and AE_i. This calculation is called marginalisation and the resulting probability is the marginal probability [15].

Relationship A

If an AE is an input (i.e. a cause) to a decision and an AR is dependent upon it, any changes in the AE will affect the probability of the AR's validity. Figure 4 shows an example of a type A relationship where both *AE1* and *AE2* are architecture elements that influence the decision *AR1*. If one or more of the causes (AEs) of a decision (AR) change, then there is a possibility that the decision will no longer be valid. In this example, the conditional probability of *AR1* can be denoted by $P(AR1 \mid AE1, AE2)$. As all nodes are

binary in states, there are 4 possible combinations to consider for *AR1* being valid.

P(*AR1=valid* | *AE1=stable*, *AE2= stable*)
P(*AR1=valid* | *AE1=volatile*, *AE2= stable*).
P(*AR1=valid* | *AE1= stable*, *AE2= volatile*).
P(*AR1=valid* | *AE1= volatile*, *AE2= volatile*).

Note that in each case P(*AR1=invalid*) = 1 - P(*AR=valid*). For the situation where all the AE causes are *stable* and not subject to change, the CPT entry for P(*AR1=valid*) reflects the confidence in the correctness of the original rationale for the architecture decision. We would expect in most cases it would be set to 1, or very close to 1. Conversely, the CPT entry for P(*AR1=valid*) would be lowest in the situation where all the AE causes are *volatile* and with a high probability that it is subject to change.

For the intermediate situations where 1 or more, but not all, of the AE causes are subject to change, the CPT entries must be chosen carefully to reflect the relative weight of the individual causes on the architecture decision. For example, if there are two AE causes and both were of equal weight, say *x*, when making the decision, the CPT entries would be the same.

P(*AR1=invalid* | *AE1=stable*, *AE2=volatile*)=*x*
P(*AR1=invalid* | *AE1= volatile*, *AE2= stable*)=*x*

The actual value of *x* must then reflect the causal impact a single volatile AE is expected to have on the decision; *x* close to 1 means the decision will become invalid even with the single volatile cause, 0.5 means a fifty-fifty chance, while *x* close to zero means the decision is reasonably robust if only one AE cause is volatile. A more dominant (*AE2*) and less dominant (*AE1*) cause combination might be represented by:

P(*AR1=invalid* | *AE1=stable*, *AE2= volatile*)=0.8
P(*AR1=invalid* | *AE1= volatile*, *AE2=stable*)=0.3

In this example, if *AE1* is stable and *AE2* is volatile, the probability of AR1 being invalid is higher than if *AE2* is stable and *AE1* is volatile.

Relationship B

When an architecture decision is made, an AR is created to record the rationale and linked with one or more resulting AEs. If an AR is invalid, the resultant AEs are more likely to be volatile and subject to change. Note that there is no direct relationship between AEs caused by the same AR, so we need only consider the affect of the AR on each AE individually. This means specifying for each AE_i the CPT P(AE_i |AR), that is the values for:

P(AE_i=*volatile* | *AR=valid*)
P(AE_i =*volatile* | *AR=invalid*)

Note that in each case P(AE_i=*stable*) = 1 - P(AE_i =*volatile*). In most cases, while the AR is valid, we would not expect the AE to change, hence P(AE_i =*volatile*|*AR=valid*) would be close to 0. Conversely, if the AR is invalid, the AE is much more likely to change, hence P(AE_i=*volatile*|*AR=invalid*) would be close to 1. During architecture design, if a requirement changes then it is possible that the AR depending on it will be invalidated and so all resulting designs from that AR may be subject to changes and become volatile. The more an AE depends on an AR, i.e. AE's volatility is high, the more likely it is subject to change when AR becomes invalid. In Figure 4, *AR1* → *AE4* and *AR2* → *AE6* are examples of relationship B.

Relationship C

It is possible that an AE is the result of multiple separate decisions (ARs). For example, in Figure 4, *AE5* is the result of two separate decisions *AR1* and *AR2*. The conditional probability of *AE5* can then be defined as P(*AE5* | *AR1*, *AR2*). Since *AE5* is dependent upon both decisions *AR1* and *AR2* simultaneously, estimates for the CPT can be expressed as:

P(*AE5=volatile* | *AR1=valid, AR2=valid*)
P(*AE5=volatile* | *AR1=valid, AR2=invalid*)
P(*AE5=volatile* | *AR1=invalid, AR2=valid*)
P(*AE5=volatile* | *AR1=invalid, AR2=invalid*)

The causal relationship being modelled in this CPT is essentially an additive one: the more decisions that are invalid, the higher the probability that the common resultant AE will be volatile. While all of the decisions are still valid, the CPT entry for P(AE5= *volatile*) will be close to zero, and if all the decisions are invalid, it will be close to 1. For the intermediate situations where 1 or more, but not all, of the AR decisions are invalid, the CPT entries must be chosen carefully to reflect the relative weight of the individual decisions on the resultant AE. In the example, suppose *AR1* has more influence on *AE5* than *AR2* and the cause combination is represented by:

P(*AE5=volatile* | *AR1=invalid, AR2=valid*) = 0.8
P(*AE5=volatile* | *AR1=valid, AR2=invalid*) = 0.3

Therefore, when *AR1* is invalid and *AR2* is valid, the combined effect to cause *AE5* to be volatile is higher (0.8). Conversely, when *AR1* is valid and *AR2* is invalid, the probability that *AE5* is volatile is lower (0.3), signifying less influence by *AR2*.

4.4. Probabilities Calculation in AREL

In this section, we use a system designed by the first author to illustrate the probabilistic causal relationships between requirements and designs. The pilot system was designed five years ago according to the FSTC PACES standard [11] to process cheque images

transferred between banks and the clearing house. Presenting banks would send cheque presentment files containing cheque images and ECP data to the clearing house for settlement. This example illustrates the system reliability aspect of the cheque clearing system. Figure 5 shows the BBN representation of an AREL model for the system with probability tables to show the prior probabilities and CPTs of some selected nodes. There are extensive functional requirements and non-functional requirements in the system. The following is an excerpt to illustrate those requirements surrounding system reliability:

- R1.0 specifies the cheque presentment file formats using ANSI X9.37 and X9.46 standards.
- R2.0 specifies the cheque presentment processing requirements
- R11.0 specifies the performance requirements
- R12.0 specifies the reliability requirements
  - R12.1 specifies that a cheque should only be processed once and once only
  - R12.4 specifies that there could be no loss of cheque images or ECP data during processing

AR1 represents the rationale for designing the Single-Pass Presentment Process (denoted by design *D1.0*) to satisfy four functional and non-functional requirements. A single pass strategy is to read the cheque presentment file once only. A presentment file contains cheque data and images that are presented to the clearing house for clearing. A single pass will improve the performance by eliminating multiple reads of a file to decrypt, authenticate and decode the data. *AR2* represents the decision to select a platform in order to satisfy the performance requirements of processing 25,000 cheques every minute. The result is the implementation of a dual nodes system denoted by *I1.0*. Based on a dual nodes configuration and requirement *R12.4*, a decision is made in *AR3* to design a mechanism whereby if one node fails, the other node can take over processing without losing any data. This design involves using a common disk storage that is accessible by both nodes and employing failover software to detect failure of a node and activates switch over processing functions. In order to ensure all cheques are processed once and once only (requirement *R12.1)*, a decision is made in *AR4* to implement design *D3.0* using a checkpoint mechanism. A checkpoint mechanism ensures that all processed cheques are committed to permanent storage and any partial transactions that have not been committed when the system fails are rolled back.

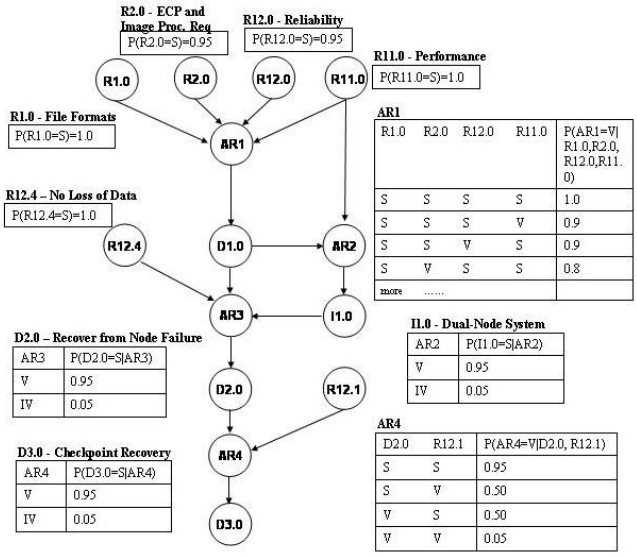

Figure 5 – A BBN Representation for the Cheque Image Clearing System (with Priors/CPTs)

Prior probabilities and conditional probabilities are assigned to the nodes to indicate the strength of their dependencies. Root nodes get assigned prior probabilities and all other nodes are assigned conditional probabilities based on the relationships to each node's parents. Prior probabilities for root node *R1.0*, *R11.0*, *R12.1* and *R12.4* are fixed at 100% because industry standards and reliability requirements dictate that they cannot be compromised.

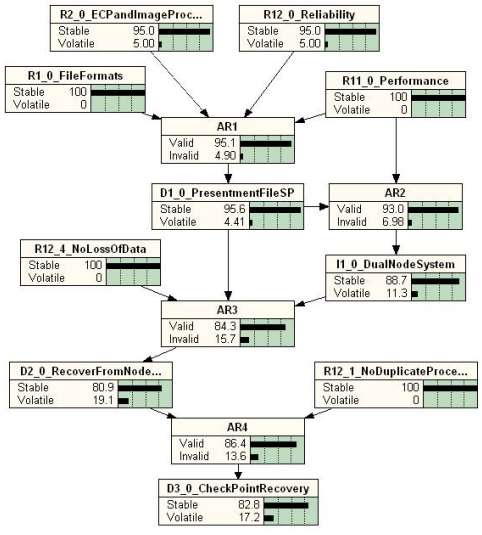

Figure 6 – Example BNN Shown with Beliefs Before Any Evidence is Entered.

The AREL model in Figure 6 is another representation of the same model in Figure 5 except it shows the marginal probability distribution when no

evidence has been added, with the visualisation provided by the Netica [19] BBN software. The visualisation includes the name of each node across the top, the state names down the left (*valid/invalid* for AR nodes, *stable/volatile* for AE nodes), marginal probability shown as percentages (e.g. 88.7% and 11.3% for *I1.0 Dual Node System*) by two horizontal bars. The marginal probabilities, for instance, of a leaf-node such as *D3.0 Checkpoint Recovery,* are calculated based on its own CPT and the CPTs and prior probabilities distributions of its ancestors. For instance, the marginal probability of *D3.0* indicates that based on the current requirements and design, there is a probability of 82.8% that the design is *stable*.

4.5. Predictive Reasoning

As requirement or system environmental changes are introduced to the system, we want to predict the effect of possible architecture design changes. Being able to identify the areas that are likely to change would improve both the accuracy and efficiency of architecture enhancements. Recall from the introduction to Section 4 that reasoning in BBN means that users first set the values of nodes in the network. This evidence is then propagated through the network, producing a new probability distribution over the remaining variables in the network [15]. Multiple evidences could be introduced at the same time. Let us consider how we can do this type of predictive reasoning in the BBN representation of AREL.

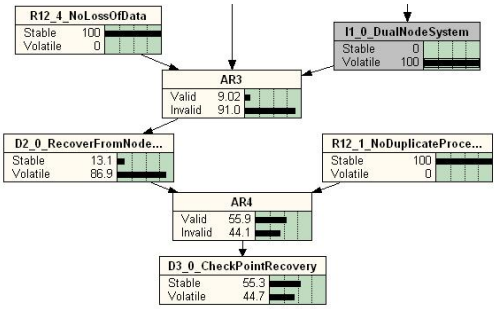

Figure 7 - Predictive Model

Architects could add one or more evidence, i.e. changes, to the AREL model. It means instantiating BBN with evidence that an AE takes new value for the *volatile* state. The BBN belief updating algorithms would then update the posterior probabilities for the descendant AR and AE nodes in the network. This means the architect will be provided with probabilistic predictions as to the effects of the AE change, in particular which decisions (AR nodes) are more likely to be invalid, and which other architecture elements (AE nodes) in turn become more likely to be volatile. We provide an illustration of this predictive reasoning for the cheque clearing system. .

Let us assume that there is a new hardware platform that has the processing capacity to satisfy the *R11.0* performance criteria using a single node design to replace the dual node design. See the highlighted node in Figure 7. This change creates the evidence that sets the *volatile* state of *I1.0* to 100%. Given this *evidence*, we can predict that there is a 91% chance that the decision *AR3* will be *invalid* and so there is a 86.9% chance that design *D2.0 Recovery from Node Failure* will be *volatile* and subject to change. Once *D2.0* is identified as likely to change, the design rationale that follows would identify any scripts that deal with dual node data sharing and switch over will have to be re-designed. The impact flows further down to decision *AR4* and design *D3.0* but with a lesser effect. *I1.0* has lesser influence here because *AR4* rationalises that *R12.1* is equally influential in this decision as *D2.0*. The non-functional requirement *R12.1* specifies the integrity of the transaction processing irrespective of platform implementation. Therefore, the effect of changing platform design in *I1.0* is balanced by the stable requirement *R12.1*, so *D3.0* is less volatile than *D2.0* with a marginal probability of volatility of 44.7%.

4.6. Diagnostic Reasoning

Another use of BBN reasoning in AREL is to diagnose the causes of change in architecture design. Given the evidence about a particular node, the BBN can diagnose possible causes by updating the posterior probabilities of ancestor nodes. Note that such diagnostic reasoning is "backwards" to the direction of the edges, which represent the causal relationships.

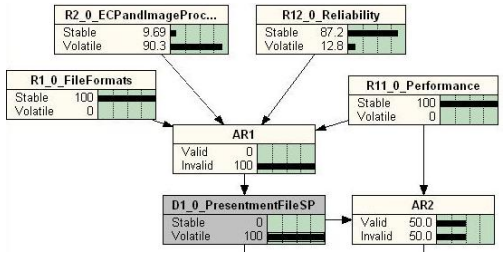

Figure 8 - Diagnostic Model

In AREL modelling, such diagnostic reasoning might take place if evidence is inserted into a non-root AE. For instance, if we change design *D1.0 Single Pass Presentment File Processing* (an AE node) to set its *volatile* state to 100% (Figure 8), we observe that

the posterior probabilities of decision *AR1,* and requirements *R2.0* and *R12.0* have changed.

The diagnostic capability of BBN can indicate all the possible reasons that might cause *D1.0* to change. In this case, the change of the requirement *R2.0* is the most likely cause, with probability of 90.3%, because it is a functional requirement that dictates the decision in *AR1*. Such diagnostic reasoning is useful because, before any changes are introduced to the system, an architect may investigate what possible causes might influence the element so as to investigate if those causes themselves require any considerations and possible modifications.

5. Tool Support

AREL uses an existing UML tool, Enterprise Architect [26], for capturing architecture rationales and representing the graphical decision network. It also uses the BBN software package Netica [19] to compute the probabilities of change impacts.

Even with existing toolsets, the usability of AREL can be improved by integrating the two tools. The following improvements are considered: (a) AREL should provide templates that allow users to enter AR qualitative and quantitative information speedily; (b) in order to manage the complexity of these graphs, the UML tool should be able to hide/display the AREL model through explorer-like function and selectively hide/display AEs according to classifications; and (c) integrate BBN computation into the graphical model to eliminate manual data re-entry. To achieve these goals, we are investigating the feasibility of enhancing Enterprise Architect and integrating it to Netica.

6. Discussion and Future Work

AREL provides direct linkage between AEs and ARs to represent the relationships between architecture rationale and architecture elements. This is an improvement over existing design rationale systems and solves the problem that design rationales are separated from design artefacts [24]. The introduction of AREL and its application using BBN has many benefits: (a) tacit knowledge can be captured and represented explicitly for decision tracing; (b) the graphical representation help architects detect design inconsistencies such as cyclical decisions; and (c) it provides a quantitative approach to predict and diagnose change impacts in architecture enhancements.

The probabilities provided by architects are estimates based on their experience and intuition. These estimates are semi-objective in nature and may not be very accurate. However, architects should be able to provide accurate estimates in the long run because feedbacks of any inaccuracy of these estimates perform would refine the way they are estimated.

It is obvious that not all system designs need to use AREL, smaller systems that are easy to trace and understand will gain very little from it. We suggest that long-lived, large and complex systems that contain intricate designs to cater for extensive non-functional requirements are candidates to use AREL. AREL targets primarily the area of architecture design and not detailed design because of the complexity and intricacy of architecture designs. As for the level of design details required in AREL, we suggest that architecture design is delineated through the architect's assessment that the risks of realising the outcome and the risk of implementing of an architecture are contained [27].

Although belief updating (i.e. computing the posterior distribution) in BBNs is known to be NP-hard in the general case [6], in practice there are efficient exact and approximate algorithms available in commercial BBN software packages. The computation time depends on: the number of nodes in the network; the number of values these nodes can take; and the topology of the networks -- the more highly connected the networks, the worse the computational complexity [15]. However, AREL networks have a reasonably simple tree-like structure which is not difficult for the updating algorithms to cope with.

The application of BBN to AREL provides a foundation for some further research opportunities in architecture design and software engineering: (a) to predict the cost of change impact to architecture enhancements; (b) to analyse the complexity of the system through observing the conditional dependency of architecture elements and decisions; (c) to consider the use of secondary links in BBN using Dynamic Bayesian Networks for temporal analysis as systems evolve over time; (d) to derive from AREL a more abstract and concise model that representing associations between decisions, which is currently being worked on and is called Architecture Decision Dependency Graph (ADDG); and (e) to enhance traceability support in multi-viewpoints architecture frameworks using design rationale and causality.

7. Conclusion

In this paper, we have defined AREL to provide a graphical representation of design rationales, architecture elements and their relationships. AREL models the direct causal relationship between design decisions and design artefacts using the UML notation.

We apply Bayesian Belief Networks to quantify the strength of dependency between architecture decisions and elements. The key benefits of this method are: (a) it enables architects to capture and represent design decisions and rationales, and to explicitly relate them to design elements in a graphical representation; (b) it enables architects to use this knowledge to predict and diagnose the impact of requirements and architecture design changes; and (c) it provides traceability for architecture elements and decisions through traversing the AREL model.

Based on AREL and its BBN application, further research opportunities such as creating models to calculate the cost of change impact and architecture complexity are possible. Although existing tools are available to support AREL and BBN, successful implementation in large-scale applications will depend on the usability of the tools. We are collaborating with tool vendors to enhance and integrate an UML tool with a BBN tool to address this issue.

Acknowledgment

We thank Sparx Systems and Norsys Software for providing us access to Enterprise Architect and Netica respectively. Both tools are invaluable aids to our work. We are further indebted to Sparx Systems for providing additional support for our research.

7. References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston: Addison Wesley, 2003.

[2] J. Bosch, "Software Architecture: The Next Step," Software Architecture: First European Workshop, EWSA 2004, St Andrews, UK., pp 194-199, 2004.

[3] L. Bratthall, E. Johansson, and B. Regnell, "Is a Design Rationale Vital when Predicting Change Impact? – A Controlled Experiment on Software Architecture Evolution," Second International Conference on Product Focused Software Process Improvement, pp 126-139, 2000.

[4] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Boston: Kluwer Academic, 2000.

[5] J. Conklin and M. Begeman, "gIBIS: a hypertext tool for exploratory policy discussion," Proceedings of the 1988 ACM conference on Computer-supported cooperative work, pp 140-152, 1988.

[6] G. F. Cooper, "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence*, vol. 42, pp. 393-405, 1990.

[7] L. M. Cysneiros and J. C. S. P. Leite, "Using UML to Reflect Non-Functional Requirements," Proceedigns of the 11 CASCON, Toronto, pp 202-216, 2001.

[8] A. Dardenne, A. Van-Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition," *Science of Computer Programming*, vol. 20, pp. 1-36, 1993.

[9] M. Denford, J. Leaney, and T. O'Neill, "Non-Functional Refinement of Computer Based Systems Architecture," The 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp 168-177, 2004.

[10] N. Fenton and M. Neil, "Software Metrics: Roadmap," Proceedings of the Conference on the Future of Software Engineering, pp 357-370, 2000.

[11] FSTC, "Paperless Automated Check Exchange and Settlement (PACES)," 2000, http://www.fstc.org/projects/paces/index.cfm.

[12] J. Galliers, A. Sutcliffe, and S. Minocha, "An Impact Analysis Method for Safety-Critical User Interface Design," *ACM Transactions on Computer-Human Interaction*, vol. 6, pp. 341-369, 1999.

[13] J. Han, "Designing for Increased Software Maintainability," International Conference on Software Maintenance, Bari, Italy, pp 278-286, 1997.

[14] IEEE, "IEEE Recommended Practice for Architecture Description of Software-Intensive System (IEEE Std 1471-2000)," IEEE Computer Society 2000.

[15] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence*: Chapman & Hall / CRC, 2004.

[16] J. Lee and K. Lai, "What is Design Rationale?," in *Design Rationale - Concepts, Techniques, and Use*, T. Moran and J. Carroll, Eds. New Jersey: Lawrence Erlbaum, 1996, pp. 21-51.

[17] A. Maclean, R. Young, V. Bellotti, and T. Moran, "Questions, Options and Criteria: Elements of Design Space Analysis," in *Design Rationale - Concepts, Techniques, and Use*, T. Moran and J. Carroll, Eds. New Jersey: Lawrence Erlbaum, 1996, pp. 53-105.

[18] T. Moran and J. Carroll, *Design Rationale: Concepts, Techniques, and Use*: Lawrence Erlbaum Associates, Publishers, 1996.

[19] Norsys Software Corp, "Netica - Application for Belief Networks and Influence Diagrams User's Guide," 1997, http://www.norsys.com/dl/NeticaMan_Win.pdf.zip.

[20] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[21] D. E. Perry and A. L. Wolf, "Foundation for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, pp. 40- 52, 1992.

[22] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," *IEEE Transactions on Software Engineering*, vol. 27, pp. 58-93, 2001.

[23] L. Rapanotti, J. G. Hall, M. Jackson, and B. Nuseibeh, "Architecture-driven problem decomposition," 12th IEEE International Conference on Requirements Engineering, pp 73-82, 2004.

[24] W. C. Regli, X. Hu, M. Atwood, and W. Sun, "A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval," *Engineering with Computers*, pp. 209–235, 2000.

[25] B. S. Shum and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?," *International Journal of Human-Computer Studies*, vol. 40, pp. 603-652, 1994.

[26] Sparx Systems, "Enterprise Architect V4.51," 2005, http://www.sparxsystems.com/.

[27] A. Tang and J. Han, "Architecture Rationalization: a Methodology for Architecture Verifiability, Traceability and Completeness," 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems ECBS 2005, U.S.A., pp 135-144, 2005.

[28] A. Tang, M.A. Barbar, I. Gorton, and J. Han, "A Survey of Architecture Design Rationale," Swinburne University of Technology SUTICT-TR2005.02, 2005.

[29] J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture," *IEEE SOFTWARE*, vol. 22, pp. 19-27, 2005.