7

Guiding complex design optimisation using Bayesian Networks

Artem Parakhine and John Leaney
University of Technology, Sydney, Faculty of Engineering and Information Technology
Australia

1. Introduction

In this chapter we aim to present an approach to addressing the problem of architectural optimisation of complex computer-based systems. Initially, we provide a survey of methods created with the direct aim of optimising system with respect to their non-functional qualities. An overview of commonalities and differences between these methods leads to exploration of reasons for their success in attaining their respective goals. The results of this discussion provide the information necessary to formulate the definition of the notion of *guidance* in the context of architectural design and optimisation.
Following that, we describe a new Heuristic-based Architectural Optimisation framework developed in an attempt to unify the lessons learnt from the successes and shortcomings of the surveyed optimisation approaches. Special attention is paid to the *guidance* mechanism employed by the new framework to control the progress of design optimisation. We propose to develop a suitable guidance mechanism using a Bayesian Belief Network (BBN) obtained from a combination of hybrid simulation modeling and BBN discovery algorithm.
Finally, the rest of the chapter is devoted to a description of a brief example problem. There, an attempt is made to show how the guidance mechanism can be applied to a fairly simple problem that nonetheless possesses an element of ambiguity that so often falters system design optimisation activities.

2. Complex System Design and Optimisation

The process of design optimisation is a special form of design synthesis characterised by a high level of specificity with respect to its goals. As such it relies on the following supporting elements:

measurable goals - clear and quantifiable statement of goals to enable evaluation of candidate designs.

design variability - the system design must present amenable elements. In the context of system design the change may be made at various levels: topology of component arrangement, interchangeable components, configuration within a component.

constraints - restrictions, such as budgeted cost and time, provide an essential definition of design feasibility which effectively reduces the search space to a manageable size.

Currently, all prevalent approaches to architectural system design can be separated into two groups based on their main artefacts: quality-driven and model-driven (Parakhine, 2009, p. 22).
The quality-driven design synthesis process focuses on extensive analysis of qualities and possible trade-offs. It necessitates creation of goal descriptions expressed partially in terms of choices required to attain them, which in turn contributes towards the evaluation of the candidate architectures. On the other hand, the model-driven methodology concentrates on representing the system design from the point of view of a specific domain (e.g. security) with consequent modification through application of well-defined domain-specific changes.
Although the two groups operate on different artefacts, both of them fit into the same general structure of an architectural optimisation process shown in Figure 1.

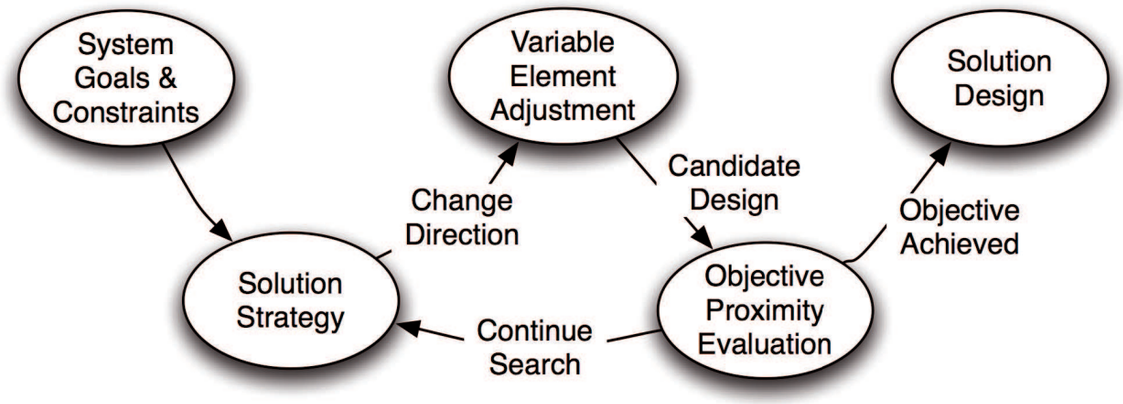


Fig. 1. A generic process of system optimisation at an architectural level.

In its simplest terms, the process of architectural optimisation can be characterised as one based on iterative derivation and evaluation aimed at satisfaction of multiple goals. The fields of computer science and system design offer a wealth of frameworks exhibiting features and capabilities of the optimisation process shown in Figure 1. Table 1 details how the requirements for facilitation of architectural optimisation have been addressed by some of the various frameworks built specifically for that purpose or built to solve general design problems in a way which makes optimisation possible.
The described optimisation approaches adopt various methods and techniques to attain their goals and produce the outcomes. Those relevant to the system design process are listed below. The presented features and techniques are grouped by the element of the optimisation process (Figure 1) whose function they were used to fulfill:

Genetic Algorithms - indicates that the framework performs evolutionary search that relies on encoding of system design in the form appropriate for breeding operators.

Meta-Heuristics - shows that the framework was created around a specific optimisation heuristic, such as simulated annealing or a tabu-search.

Function Analysis - identifies frameworks that employ a mathematical function analysis to drive the search process.

Configuration - only variations in the properties of the system components are considered.

Component - this approach to system change adopts components as variable elements.

	Goals & Constraints Focus			Solution Strategy			Design Variables			Evaluation Mechanism			End Result	
Architectural Optimisation Approach	Individual	Trade-off	Multiple	Genetic Algorithms	Meta-Huristics	Function Analysis	Configuration	Component	Structure	Calculation	Simulation	Runtime Monitoring	Design Suggestion	Complete Design
Diaconescu, Mos, Murphy (2003)	✓				✓			✓				✓		✓
Coit & Konak (2006)	✓			✓	✓			✓	✓		✓		✓	
Bucci, Streeter, Maio (1979)	✓					✓		✓	✓	✓			✓	
Bosch & Benngston (2001)	✓					✓		✓		✓			✓	
QDAD		✓			✓			✓	✓	✓	✓		✓	
Gockhale (2004)		✓		✓			✓	✓			✓			✓
Grunske (2006)		✓		✓				✓	✓	✓	✓			✓
MDAD			✓		✓		✓	✓	✓	✓	✓			✓
Sharma & Trivedi (2005)			✓		✓		✓	✓			✓		✓	
van Gurp & Bosch (2000)			✓		✓		✓	✓	✓	✓			✓	

Table 1. Summary of approaches to architectural system optimisation.

Structure - focuses on variations in the structural arrangement of system components.

Calculation - describes a mechanism relying on a evaluation of a mathematical function.

Simulation - performs assessment of candidate architectures by constructing simulations of system operation.

Runtime Monitoring - evaluation is performed by making changes to the system and collecting metrics during its subsequent operation.

Design Suggestion - indicates that the optimisation produces a prescriptive list of design modification.

Complete Design - shows that the framework produces a system architecture specified to a relatively high level of detail.

The approaches to architectural optimisation listed in the matrix in Table 1 are arranged in the order of increasing number of goals with the lowest level containing approaches aimed at a specific non-functional system qualities such as maintainability (Bosch & Bengtsson, 2001) or reliability (Gokhale, 2004).

The first entry in the table is the Automated Quality Assurance (AQuA) framework (Diaconescu & Murphy, 2005) which attempts to automatically tailor the system to the operational conditions by varying the choice and configuration of the system components. The AQuA framework is built to continuously check the system for *performance anomalies* and apply a

pre-defined *adoption decision* should such anomaly be detected. As such the framework is limited severely by the quality of the metric collection and the depth of analysis that can be performed in a timely manner during system operation so as to preempt possible reduction in performance.

Further to that, the problem of achieving specific qualities can be better addressed before the system is built. For example, an existing requirement for high reliability can be addressed by exploring variations in the choice and number of redundant components in each of the system modules (Coit & Smith, 1996). This formulation, in effect, links the single non-funcitonal property (reliability) to a single type of structural architecture change (connection of components in parallel).

However, introducing changes that may be applied in combination may render a large set of possible candidate architectures. As this approach to optimisation developed, a number of strategies such as Genetic Algorithms (Coit & Smith, 1996), integer programming (Coit, 2001) and Multiple Weighted Objective optimisation heuristic (Coit & Konak, 2006) were used to restrict and manage the pool of candidate solutions.

Bucci & Streeter considered three version of system distribution structure: centralised storage, individual storage at each node and no centralised database, or hierarchical storage where individual nodes synchronise with the centralised storage unit. At the core of this approach was a model developed to assess the cost per transaction for each of the possible three arrangements. The resultant model allowed to determine the cost-minimizing approach to structuring architecture and showed how it would affect the response time and, by extension, the overall throughput. This development has led to the creation of a more generalised model (Bucci & Maio, Sep 1982) built to handle the trade-off between performance and cost.

Another focused optimisation approach was proposed by Bosch & Bengtsson who attempted to formalise the relationship between the properties of the system design and the effort it would take to change it during maintenance cycle. The key relationship guiding the optimisation was expressed as *effort*, or a form of cost, based on the sum of productivity factors associated with each potential change.

Such focus on a single quality fails to address the situations presenting conflicting quality requirements. This led to the development of the Quality Driven Architectural Design (QDAD) family of frameworks such as Architectural Trade-off Analysis Method (Kazman et al., 1999) and Business IT Alignment Method (Chen et al., 2005). These approaches attempt to identify and manage the strong trade-off relationships existing in the design space. To achieve this, a typical QDAD method relies heavily on a collection of mature analytical models similar to the ones described above. Unfortunately, formulating, using and maintaining such combination of metrics would be a complex and arduous task as these models would have to be manually adjusted as system design changes in time and scale.

Another group of optimisation methods listed in Table 1 adopt an approach focused on design options and how they can be optimally combined to get the highest feasible level of overall system qualities.

For example the identification mechanism for “good” architectures shown by Grunske represents a culmination of the development of this type of change-centric approach (Grunske, 2003; 2006; Grunske et al., 2005). There, Grunske uses principles of graph representation, architecture refactoring and genetic algorithms to create an optimisation framework. In this framework, the mechanism of guidance relies on the genetic fitness function that is used to determine the successful mutations propagated into the next generation.

The genetic algorithm approach to multi-objective optimisation and the corollary use of fitness function is not without merit. The ways the concept of *fitness* was used by different approaches (Coit & Smith, 1996; Grunske, 2006) are combined and extended further by the detailed fitness function used by Gokhale to specifically address the trade-off between two non-functional qualities (Gokhale, 2004). Although it is recognised that the successful application of genetic algorithms relies greatly on many factors such as mutation rates and population sizes, Grunske and Gokhale show that the fitness procedures have potential to handle the complex link between architectural assessment and complex multi-objective process of architectural optimisation. Still, the problem of creating such a fitness function remains non-trivial and worsens with the increase in the number of goals.
Next, Table 1 lists the Model-Driven Architectural Design methods which are based on principles of Model-Driven Design (*MDA Guide*, 2003). Although the approaches in this group do not explicitly target optimisation, they nonetheless provide an opportunity to perform it. This is achieved by structuring the process as one comprised of models and transformations. Consequently, the optimisation can take place by applying different transformations to the underlying model. However, the separation into models and transformations ignores the possible conflicts between transformations.
Still, the static analysis of artefacts created as part of MDAD can be used to better understand the system. Additionally, it can be extended as shown by Sharma & Trivedi who used Discrete Time Markov Chain modelling to estimate the reliability, performance and security of a given system by examining the number of visits and times spent in that module calculated from a transitional probability matrix (Sharma & Trivedi, 2005). The probabilities within this matrix are affected by the architecture as different structures and behaviour would lead to the variation in time spent in each modules and the number of visits to specific modules required to finish processing. As such the matrix links compositional features of the system with he models used to estimate its individual qualities, in this case: reliability, performance and security.
The main limitation of the transitional probability matrix is that it does not provide an explanation of how design decisions affect the distributions comprising the matrix. Achieving such form of representation is, nonetheless, possible as shown by the SAABNet framework (van Gurp, 2003, p. 71). At the foundation of SAABNet is a Bayesian Belief Network or BBN which combines probabilistic and factual knowledge about the system features and understanding of its qualities into a single informational resource.
The nature of BBN notation has the power to include information from models, simulations and surveys of domain experts as well as records of architectural changes that have taken place in a given system or across multiple systems. Once assembled, the BBN can be used to determine which combination of factors is most likely to lead to a higher level of a desired quality while at the same time give the designer some idea regarding the effects the change in factors may have on other non-functional system qualities. Resultantly, the SAABNet can be employed to support both the QDAD and MDAD approaches mentioned earlier in this chapter. The SAABNet network and the corollary issues are discussed in further detail in Section 4.

3. Heuristic-based Optimisation Framework

As mentioned in the previous section, the design optimisation framework proposed to address the shortcomings of methods presented in Table 1 is focused around the iterative process borrowed from the field of control theory. The core of the framework is based on the

conventional control feedback loop which exists with the intent of providing a mechanism for continuous adjustment of the system behaviour based on changes detected in the set of control variables. Leaney et al. (2004) proposed an extension of this approach into the field of design optimisation with the design process itself being the system under control. Further advances in the fields of architectural refinement (Denford et al., 2004) and heuristic encoding (Maxwell et al., 2006) spurred a development of a more detailed version of the framework; its current state shown in Figure 2.

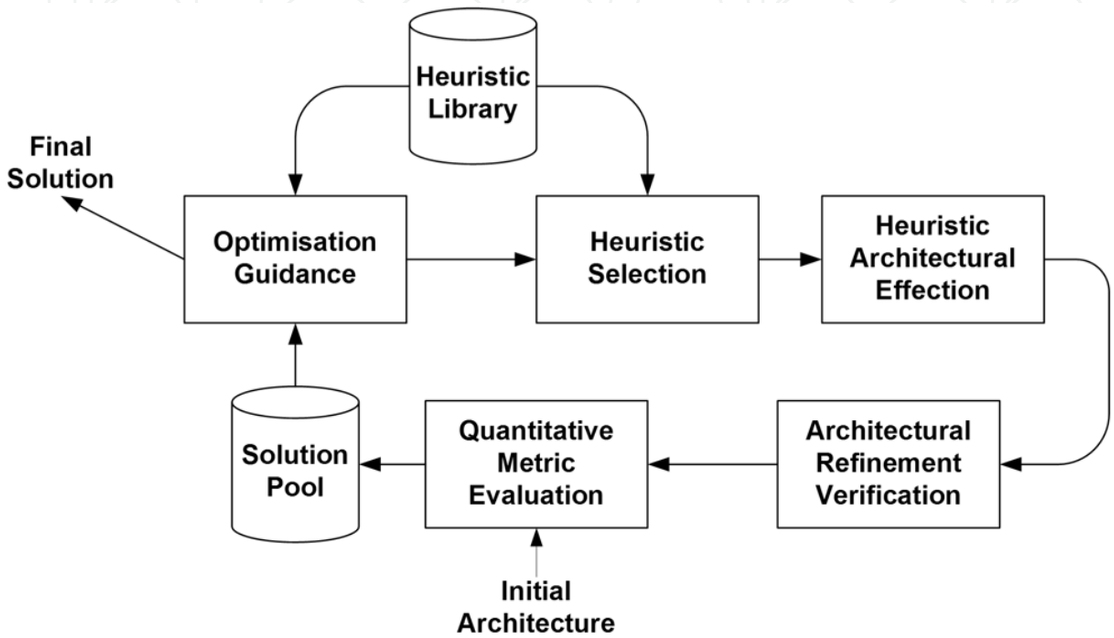


Fig. 2. General view of Heuristic-based Architecture Optimisation framework (Maxwell et al, 2005).

The Heuristic-based Architecture Optimisation (HAO) framework proposed by Maxwell et al. represents a combination of both *constructive* and *local search* methods of solution of approximation. It uses constructive heuristics to change the design of the system during optimisation while relying heavily on a guidance strategy to establish the *neighbourhood of solutions* for a current state of system design and select the appropriate change heuristic.
Specifically, it possesses the following features:

iteration The search for an optimal solution is conducted by iterating through a series of design candidates generated from the application of specific design heuristics.

memory The change heuristics, are, in essence, an abstract codification of specific transformations (Maxwell, 2007, p. 73). As such their sequential application may lead to compound effects that can provide important insights into properties of the system being designed. Hence, the HAO process records all generated solutions in the *Solution Pool* with the intent of allowing the designer to analyse and study properties of the architecture.

non-parametrised view of design Use of heuristics also allows the HAO framework to adopt a holistic view of architectural design without the need for an intermediate step of decomposition into a limited set of mutable parameters.

functional preservation The framework also ensures, through the use of *Architectural Refinement Verification* (Denford et al., 2004), that in the process of design generation none of the design's functional properties are affected in a negative way.

generality Domain-specific heuristics and quantitative metric evaluators can be used to support optimisation of systems from a multitude of domains.

decision support Finally, at its core, the HAO framework is oriented at organising and supporting the activities performed by the system designer during a search for a new design candidate

In its form, the HAO framework attempts to incorporate all the best features of the optimisation frameworks described in Section 2 whilst also exhibiting and addressing the shortcomings of those frameworks. For example, the proposed use of heuristics relates to the generality achieved by the *refactoring* approach described in (Grunske et al., 2005) and non-parametrised view of design is an extension of evolutionary chromosome-based search proposed in (Gokhale, 2004). Furthermore, the modular structure evident in the proposed framework was chosen to ensure that the modules could advance with a degree of independence. Thus, the framework can take advantages of different, possibly domain specific, libraries of heuristics as well as a range of externally defined quality evaluators.

The ultimate aim of the process is to find a solution that represents an optimal compromise on competing system qualities. To achieve a positive outcome the process requires priming, which includes obtaining a set of potential design changes encoded as heuristics (Maxwell et al., 2005), providing a baseline design, and a set of goals and constraints. Effectively, the goals and constraints must contain the following: the expression of desired non-functional qualities that should be present in the solution and the minimal levels to which these qualities must be achieved for a given solution to be considered valid.

The original design is then evaluated to establish a baseline measurements with respect to optimisation goals. Once assessed, the baseline design and its associated characteristics are placed into the *Solution Pool*, which is used establish a historical context that may be used by the designer to study the *evolvability* characteristics of the system (Rowe et al., 1998).

Consequently, the baseline design information is used by the *Optimisation Guidance* component to generate information regarding the type of change that is most likely to render positive results. This information is then applied within the *Heuristic Selection* module to find viable candidates from the options available in the *Heuristic Library*.

Following selection, the identified heuristics are applied the *Architectural Refinement Verification* is used to confirm the candidate designs preserve functional characteristics of the original architecture. Finally, the candidates are evaluated and the results are provided back to the *Optimisation Guidance* component for further analysis.

The role of *Optimisation Guidance* is to determine the best possible alternative architecture in the current solution pool and which combination of heuristics, when applied, would produce an outcome closest to the desired system form. To achieve this, the guidance mechanism uses a combination of Bayesian principles and Hybrid Simulation to form a new view of the system from the viewpoint of potential evolutionary changes directed at achieving the goals of optimisation process. The specifics of guidance composition and operations are described in the coming sections of this chapter.

4. Guidance Mechanism Theory and Application

The core operations of the *Optimisation Guidance* component are performed using a Bayesian Belief Network (BBN) representation of causal links between change heuristics and system qualities. A BBN can be viewed as a *directed acyclic graph* in which the *nodes* represent variables, or assertive propositions, and *edges* describe the presence and direction of influences which exist between connected nodes (Pearl, 1986, p. 358). The relationship where one variable A influences another variable B, denoted by the existence of a directed *arc* linking a *node A* to another *node B,* carries a nomenclature of A being a *parent* of B and, conversely, B being a *child* of A.

Each *node* in the graph can assume a number of known states according to an associated Conditional Probability Distribution (CPD). The CPD of each node defines the probabilistic features, states, of the *node* with respect to the combinations of states of its *parents*. Thus, a Bayesian Belief Network is a graph representation of a joint-probability model. This representation is obtained through the decomposition of the original model into a product of conditional probabilities for each of the comprising *nodes* (Pearl, 1986, p. 359). This definition is expressed formally in the Equation 1 below:

$$P(x_1, ..., x_n) = \prod_i P(x_i|S_i) \tag{1}$$

where:

$x_1, ..., x_n$	are	variables present in the graph
S_i	is	set of *parents* for variable x_i
$P(x_i\|S_i)$	is	conditional probability for variable x_i

The main advantage of this decomposition is that it enables the construction of a BBN to be undertaken in a modular format. Since the overall model does not have to be defined in its entirety it can be built gradually through repeated application of a simple process focusing on each individual variable x_i (Haddawy, 1999). The natural restriction of scale that results from such *localisation* means that the process can be based on human judgement to obtain qualitative relationships perceived to be significant in the context of the problem.

Generally, the process of BBN construction can be split into two phases: identification of variables and elaboration of their interrelationships. This breakdown means that in building a Bayesian network model, one can first focus on specifying the qualitative structure of the domain and then focus on quantifying the influences. When this process is finished, a complete BBN is guaranteed to be a complete specification of a joint probability distribution. This specification is subject, however, to the possible shortcomings in the awareness and judgement regarding the specifics of the domain. As a result, once completed a BBN should be verified and validated and only then can it provide a useful mechanism for probabilistic inference. This, in turn, led to the frequent use of BBNs for modelling domain knowledge and implementation of probabilistic reasoning capabilities in Decision Support Systems (Tsamardinos et al., 2006).

In recent years the concept of Bayesian Belief Networks has played a pivotal role in several attempts undertaken by researchers to represent knowledge about the non-functional qualities of systems. SAABNet or Software Architecture Assessment Belief Network, developed by van Gurp & Bosch (2000), was created with the aim of helping the designer perform qualitative assessment during the architecture design process. SAABNet utilitsed a static Bayesian Belief Network to describe the interrelationships between design qualities.

The interdependencies between various nodes described by Gurp and Bosch are defined within a framework similar to the Factor-Criteria-Metric model descirbed by McCall (1994, p. 1086). Figure 3 shows an example of a single slice of through the SAABNet model. The slice shows all the quality factors influenced directly, or indirectly, by "implementation language".
The belief network structure at the core of SAABNet is comprised of three types of nodes:

Architectural Characteristic nodes (AC) represent various system characteristics and their interdependencies, eg. *component granularity* which in SAABNet is shown as dependent on implementation language and architectural style. The states for the nodes of this type are expressed in terms of specific measurements for features, such as “C++” and “Java” for the node named *implementation_language* or “high” and “low” for the *dynamic_binding* variable.

Quality Criteria nodes (QC) represent composing features of the external quality factors. Due to their higher level of abstraction the QC nodes are assigned states that are qualitative and measurable in the context of the system. For example *vertical_complexity* can assume state “high” and “low” and *testability* can be “good” or “bad”.

Quality Factor nodes (QF) represent external quality requirements such as *reusability* and *complexity*. All of the QF nodes are given qualitative states such as “good” and “bad” that are meant to reflect the level of desired qualities that the system is perceived to possess.

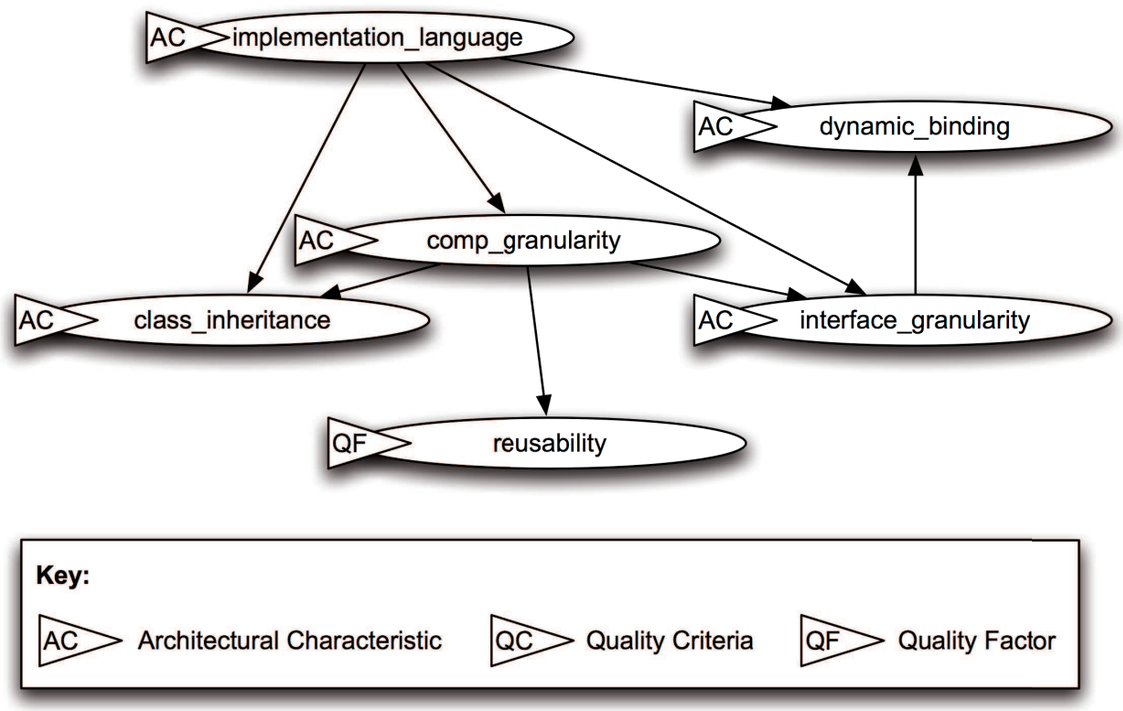


Fig. 3. An example slice thought the SAABNet described by van Gurp (2003, p. 73)

SAABNet was intended by van Gurp & Bosch to assist designers in identifying which system properties were most likely to effect their desired system qualities. In this way it would allow

them to select the goals they deemed desirable and, using the properties of the BBN, identifty specific levels, or values, of system parameters needed to meet those goals. For example, should the architect be faced with the task of increasing the *configurability* and *understandability* of the system, then it is possible to use SAABNet to obtain an advisory set of measurements for architectural characteristics. In this case *interface and component granularity* could be identified as a likely characteristic to contribute towards achieving the desired qualities.
Additionally, SAABNet can provide the designer with information about variables that will need special attention during the development process. This special attention is identified, and warranted, by the variables possessing a large number of dependency relationships. Furthermore, by examining the links between various nodes in SAABNet, the designer can identify possible conflicts and trade-offs which exist in the architecture. All of these applications can provide valuable aid to a process of complex system optimisation.
Fundamentally, SAABNet represents an attempt to construct a persistent and reusable belief model of influences between architectural characteristics, their combinations, and quality requirements. Certain parallels can be drawn between SAABNet and the SAU framework proposed by Folmer & Bosch and, in their motivation and focus, the two can be said to be related with SAABNet representing an attempt to take advantage of the representative and reasoning capabilities provided by BBNs.
Naturally, the success of this approach is rooted in the assumption that the non-functional *factors* contributing to the overall quality of the system can be defined in a generally applicable manner. However, finding a general definition for *scalability*, a common quality requirement in many system designs, might omit the nuances specific to the given problem with disastrous consequences.
One way of dealing with the generic nature of qualities is to use qualitative descriptions of measurements, using terms such as 'high'/'low' and 'present'/'absent'. Although somewhat vague in their descriptive power, the chosen names for states of qualitative variables featured in the resultant BBNs are useful in a decision-making process. The contrasting binary nature of their states helps to leverage the possibly incomplete empirical view of the system qualities and the facilitate the process of quality fulfilment (van Gurp, 2003, p. 81).

4.1 System Identification

The use of belief networks with a *static structure* in the context of system analysis, and even in optimisation, has shown some promise. Notably, BBNs can facilitate exploration of the causal relationships that exist between the system structure, the decisions that led to its formulation and the qualities affected by its form. The mathematical properties of BBNs, inherited by the described networks, allow the designer to explore, albeit to a limited extent, the effects of hypothetical design changes on the qualities exhibited by the system.
However, the networks do not address the problem stemming from the variations in the strengths of existing relationships, or the formation of new ones that may take place over time in different problem domains. Furthermore, the assignment of states, nodes and actual topology within the network is uniformly manual and leads to a very high cost that must be incurred during network construction. Due to the heavy involvement of personnel in this network construction phase, the resultant networks may be affected by the failings of individuals and political issues. Nevertheless, the Bayesian approach presents a number of avenues for addressing these issues.
Specifically, it is possible to reduce the residual effects of human errors and political bias by attempting to encode the assumptions held by system stakeholders in a simulation model

which is then used to produce output suitable for BBN construction. This is possible because the relationships between a system and its non-functional qualities is one of emergence. That is the features of the system interact to achieve its non-functional properties. Hence, in terms relevant to the implementation of an optimisation process, it can be said that a computer-based system design can serve as both the subject and the objective of optimisation. Its elements are used in evaluation of its fitness and present the context for choice and application of a solution strategy.

Due to these characteristics of a system architecture, the task of optimising it to achieve greater levels of system qualities relies heavily on knowing the specifics of how the qualities exhibited by the system link with the particulars of its features. As a result, the implementation of architectural optimisation guidance can be postulated as a problem comprised of two parts:

System Identification - It is necessary to recognise that any system (computer-based or otherwise) is a result of interactions of the elements composing it. The interactions result from the choice of components and their structural arrangement. Hence, to guide the optimisation process it is first necessary to identify the form for the system that takes into account the choices and interactions that lead to its exhibited qualities.

System Analysis - The guidance methodology must incorporate knowledge that would allow the designer to examine the set of features exhibited by the identified *system form*, determine the factors that have the greatest influence and elicit a strategy for changing the source system architecture.

The problem of system identification can be described as one aimed at deriving a parameter-based mathematical description of the system. This is an example of a *white-box* model. In other words, a model that is a completely transparent construct built using the well-known foundational assumptions regarding system features, which in this case refers to non-functional qualities. Such description can be as simple as a set of polynomial functions or a genetic representation which depends on presence of components or structures within the system.

This potential variety of forms gives rise to a more general system identification question:

> What is the representation of a *systems' design state* suitable for guiding the architectural optimisation of a system that is complex in its composition and interactions with its environment?

Developing the answer to this question requires a *model* that takes into account two major factors. Specifically, what a systems is and how it came into its current arrangement.

According to the definitions provided by Bunge (1979), a system can be represented as a set of non-propositional functions. Such form implies that the functions can not be evaluated to *true* or *false* given their domain alone and successful interpretation requires understanding the relationships that exist between the system and the design context. This, in turn, requires that the process of design itself must somehow be evident in the representation of the system created to answer the question stated above. The problem, however, is that system design is motivated by changes in the environment existing outside the boundaries of the system.

Seen from the optimisation guidance point of view, the design process is a series of decisions taken based on understanding the properties of all relevant artefacts, short-term and strategic goals and constraints regarding potentially applicable changes to the system. Consider a series of decisions such as this to be (Peterka, 1981, p. 9):

$$D_0, D_1, D_2, ..., D_n \tag{2}$$

Each element in the series (2) represents a designer's choice from a number of options. This choice is made under a degree of uncertainty regarding the true potential implications of the decision. At time 0, the very first decision is based on original goals and assumptions and introduces new information regarding system properties. This means that in its most general form a single decision can be decomposed into inputs u and outputs y:

$$D_t = \{u_t, y_t\} \tag{3}$$

In the above, u_t represents a set of values, for example specific choices of components or structure, imposed by the designer at time t onto the system. Whereas y_t represents the results of combining the inputs with the existing system context. It is the outputs set y_t that contains the resultant and emergent qualities of the system. These qualities, just like the outputs themselves, can only have their values influenced indirectly, through preceding design choices. However, the outputs are not restricted to qualities and can include other information such as measurements and forecasts that can serve an important role in future decisions.
Based on a combination of (2) and (3), the overall design process can be represented as a series of sets of inputs and outputs like so:

$$u_1, y_1, u_2, y_2, \ldots, u_{t-1}, y_{t-1}, u_t, y_t, \ldots \tag{4}$$

Given (4), the series of decisions between some two points in time i and j with $i < j$ can be shortened to the simpler form:

$$D_i^j = \{y_j, u_j, D_i^{j-1}\} \tag{5}$$

Within this new representation, if the value for index i is omitted then it can be assumed that D^j represents a set of all decisions from the first to the one taken at j.

$$D^t = \{D_1, D_2, \ldots, D_{t-1}, D_t\} \tag{6}$$

For a designer to make optimal decisions as part of the design process they need to decide in advance which of the available heuristics is optimal in the context of the design problem. To accomplish this, they must be able to forecast, before the input u_{t+1} is applied, what possible values the future elements in the input-output series (4) can take, for at least some of the distinct design choices available at time t. To determine this, the designer needs the conditional probability distribution:

$$p(D_{t+1}^{t+N} | D^t) \tag{7}$$

At this point the chain rule, shown below as (8), can be applied to produce the expanded equation shown in (9).

$$p(x_n, x_{n-1}, \ldots, x_1) = \prod_{k=2}^{n} p(x_k | x_{k-1}, \ldots, x_1) \cdot p(x_1) \tag{8}$$

$$p(D_{t+1}^{t+N} | D^t) = \prod_{i=t+1}^{t+N} p(D_i | D^{i-1}) \tag{9}$$

The equation (9) can be further expanded by making use of the basic relation stated as in (10) to obtain the following substitution for the product expression shown on the right side of (9):

$$p(a,b|c) = p(a|b,c)p(b|c) \tag{10}$$

$$p(D_i|D^{i-1}) = p(y_i,u_i|D^{i-1}) = p(y_i|u_i,D^{i-1}) \cdot p(u_i|D^{i-1}) \tag{11}$$

This, in turn, through substitution into (9), leads to the following expanded form:

$$p(D_{t+1}^{t+N}|D^t) = \prod_{i=t+1}^{t+N} p(y_i|u_i,D^{i-1}) \cdot p(u_i|D^{i-1}) \tag{12}$$

The factors in (12) have the following interpretation. The conditional probability distribution, as shown in (13), describes in general terms the transformation by which the input u_i is determined on the basis of the known past history of the decision process.

$$p(u_i|D^{i-1}) \tag{13}$$

The other factor, i.e. the set of conditional probability distributions, shown in (14), describe how the output y_i is a result of past history of the decision process D^{i-1} and the last choice of inputs u_i at each instance i.

$$p(y_i|u_i,D^{i-1}) \tag{14}$$

The set of conditional probability distributions shown in (14) is, in effect, the general description of the system as a compound result of the decisions which lead to its composition at point i. Hence, the requirement for the definition of a model for *systems' design state* posed earlier in this section can be fulfilled by *finding a set of conditional probability distributions (14) over a finite set of parameters for the time period required*. In effect, the *conditional probability distribution* links the elements of the proposed change to the fulfilment of design goals thus aiding the designer in devising an appropriate design strategy under uncertainty.

Importantly, the product of the conditional probability distribution (14) featured in (12) resembles closely the definition of the underlying model for Bayesian Belief Network presented in Equation 1. The formulae and nomenclature introduced in this section provide positive assessment as to the viability of the guidance approach based on use of BBNs for quality modelling. However, the analysis of existing BBN approaches discussed at the beginning of this section highlights the importance of the network construction process to the viability of BBNs as guidance tools supporting an architectural optimisation process. Consequently, the features of the BBNs relevant to their construction need to be examined.

As mentioned before, the key feature of BBNs is the *localisation* they afford, in other words, the fact that they provide a method for decomposing a probability distribution into a set of local distributions. The semantics of *independence* associated with the network topology specifies how to combine these local distributions to obtain the complete joint probability distribution (Equation 1) over all the input variables represented by the nodes in the network.

This has two important consequences:

1. When specifying a joint probability distribution as a table of values, the required a number of values increases exponentially with the number of variables. In systems in which interactions among the input variables are sparse, Bayesian networks drastically reduce the number of values required to be considered simultaneously.

2. The separation of qualitative representation of the influences between variables from the numeric quantification of the strengths of the influences has a significant advantage for knowledge engineering.

These structural and semantic features of BBNs are conducive to their construction and effectiveness, even in the presence of limited about of data. Specifically, the advantage mentioned in point 2 above is the greatest contributing factor to the simplicity of the BBN construction process. The process itself is broken down into two parts.
The first phase involves careful consideration of relevant concepts in the domain of the problem that is being modelled. In the domain of system optimisation this amounts to encoding of variations in inputs affected by the designer. These variations must also be accompanied by a recording of corresponding changes in the outputs that emerge as a result of the designer's decision. Upon completion of variable identification, the next step in BBN construction involves the assignment of dependencies between the variables. This assignment is accompanied by conditional probability information. After these two phases the resultant network is tested to verify that it matches the existing knowledge of the modelled domain.
Furthermore, the focus on domain-specific decisions, afforded by the use of Bayesian network technology, can, when applied within the context of system design optimisation, be interpreted in a dual way. The first one, as evident from work done by van Gurp, involves building one large network to represent the *system design state* in the most complete way possible. This is a complex and arduous task which incurs heavy validation and verification requirements for its outcomes. Even when this is completed, there is no guarantee that the resultant BBN structure would be able to withstand the test of time as the technological and business elements of the context evolve. For large systems, this is impracticable from an engineering standpoint.
Instead, the problem of BBN construction can be solved by adopting one of the available network discovery algorithms. These algorithms have been created to allow the structure of a Bayesian network to be learned from a sufficiently large set of observational data (Tsamardinos et al., 2003). One such approach is the Min-Max Hill-Climbing (MMHC) algorithm (Tsamardinos et al., 2006). The MMHC algorithm was chosen for implementation in the proposed guidance mechanism due to its effectiveness and simplicity of application.
The inclusion of the MMHC algorithm as part of the proposed guidance methodology introduces an additional problem concerning the availability of observational data. Within the context of a complex design problem such data may need to include specific values for variables representing quality measurements correlated with specific system and design states. Obtaining such data may not always be possible even if is the designer has access to variety of sources such as interviews, historical records, design documentation and system telemetry. To address this, the proposed methodology includes multi-paradigm simulation described in greater detail in Section 4.2.1.
However, in complex systems it is possible that different unknown dynamic relationships can lead to the same topographic configuration of the resultant Bayesian Belief Network. Hence the application of the proposed method takes place in a context where two distinct elements are present: the reality of the situation being analysed and a set of possible candidate models which can be produced from the available data set. The problem of ensuring that the identified model is the actual, or a very close, depiction of the qualitative relationships present in the system can only be handled by submitting it to the evaluation by the system's stakeholders and the designer.

4.2 Guidance Components and Model

The proposed guidance methodology relies on two supporting concepts: multi-paradigm, or hybrid simulation (Section 4.2.1), and automatic discovery of causal relationships between a system's characteristics and its qualities (Section 4.2.2). The role of the former is to provide the designer with a facility capable of modelling the current, as well as possible, structural and behavioural characteristics of the system. The latter of the two is necessary to ensure that data collected during the successive runs of the modelling and exploration phase are aggregated into a meaningful representation of causalities. In this case a BBN is used to understand the causal relationships and to identify the drivers behind the specific non-functional qualities of the system.

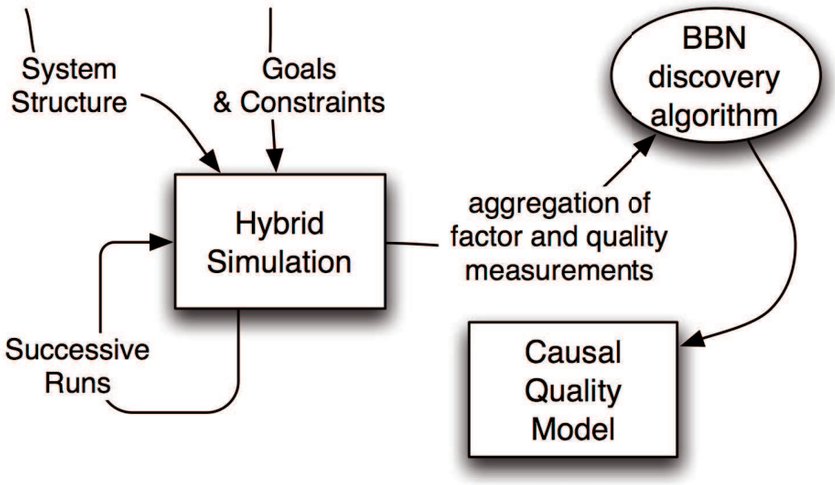


Fig. 4. The formulation process resulting in the production of guidance data.

The interaction between elements described above is depicted in Figure 4. Two elements (*Hybrid Simulation* and *BBN Discovery Algorithm*) are employed to produce a, best-effort, comprehensive understanding of the system with respect to the requirements and constraints imposed upon the designer. This interaction of composing elements is proposed as one that would help to ensure that no factor is omitted from exploration and analysis of system's design state. The exception being where the designer explicitly wishes to ignore certain factors. Within such an approach it is necessary to ensure that the simulation of the system is flexible enough to represent the ramifications of possibly very large and complex changes both on system structure and on the outside elements such as users or external processes.

4.2.1 Hybrid Simulation

The practice of exploring quality characteristics of Computer-Based Systems through construction and execution of various type of simulations is well established, as was shown in the overview of research presented in Section 2. However, the diversity of specific features that are combined to attain individual qualities in any given system demands creation of a portfolio of simulations, each targeting a single or a group of related qualities. The process of creating and maintaining such a portfolio becomes a task in itself. A task that is made difficult by issues of maintaining consistency throughout the various simulation models comprising the portfolio and creating ways to combine their outputs to construct a context describing the

non-functional qualities of the system and, as a result, provide effective analysis and decision support.
Therefore, in the context of a design optimisation guidance that aims to improve a variety of possible interrelated qualities exhibited by a complex CBS, it is desirable for the designer to be able to construct a single simulation model. Doing so would address the problems of maintaining consistency across multiple models as well as difficulties in combining their outputs. However, to be successful, this single, all-encompassing model must be rigorous enough to provide a faithful representation of a complex system while remaining flexible enough to facilitate exploration of relationships between *emergent* system qualities and the effects of possible design changes.
In order to achieve this flexibility the simulation must be able to deal with concepts from three major paradigms which currently dominate the field of simulation modelling (Borshchev & Filippov, 2004):

- System Dynamics (SD) - Required to be able to represent the effects of policy introduction or modification at the highest levels of abstraction, as well as the analysis of trends and other system *properties-of-the-whole* (Forrester, 1991, p. 22).
- Discrete Events (DE) - Required to understand the issues associated with utilisation of various resources available to the system, as well as the effects of various scheduling decisions.
- Agent-Based (AB) - This paradigm allows the simulation of elements which can only be meaningfully represented as active objects with individual, purposeful behaviour (Borshchev, 2005, p. 8).

Borshchev et al. (Borshchev, 2005; Borshchev & Filippov, 2004) has proposed combining the modelling paradigms mentioned above. The proposed combination places a special emphasis on the use of Agent-Based modelling to accentuate both its flexibility and pragmatism in situations where complete information about the system may be unavailable. Additionally, the AB approach facilitates the exploration of the *emergence* of global system properties. It achieves this by examining the interactions of various system elements over time, a characteristic which can be successfully employed to support the process of causal discovery described in Section 4.2.2.
The multi-paradigm, hybrid simulation, approach (Borshchev & Filippov, 2004) possesses both abstract characteristics (system dynamics, discrete events) and pragmatic behavioural properties (agents). Combined together these approaches provide a modelling paradigm that has been shown to be flexible and powerful enough to address the simulation and optimisation needs of problems ranging from supply chain management (Almeder & Preusser, 2007) to enterprise IT cost analysis (Popkov et al., 2006).
The overall aim of the proposed optimisation guidance method is to provide a recommendation on a list of changes considered by the designer. To achieve this, it relies on a simulation model of the original system. This model has to be robust enough to allow exploration of pre-defined usage scenarios targeted, for example, at exploring the *scalability* of the system.
It is possible that multiple runs of the simulation will be required in order to obtain a better understanding of the causal relationships between various observed system factors and its overall qualities. Eventually, the framework will be able to output a large set of aggregated data combining information on the effects of variations in system factors on the system qualities measurements. This resultant set of data will serve as an input for the automatic casual discovery algorithm.

4.2.2 Bayesian Belief Network Discovery

In Figure 4 the causal discovery mechanism of the proposed design optimisation guidance method is represented by the "BBN discovery algorithm" entity. This refers to an automatic way of constructing the topology of a Bayesian Belief Network from a large set of observational data.

The BBN elucidated from the simulation output data is used as a Causal Quality Model of the system being subjected to design optimisation. In this capacity, the function of the resultant BBN with respect to the designer is somewhat similar to that of a Decision Support System. Its aim is to show the designer which factors within the system have the greatest measure of effect on the system qualities specified as acquisition concerns. Further, it aims to show how these, and other, qualities may be affected by possible design changes.

To achieve this, the BBN encapsulates the main inference relationships present in the system design with respect to the emergent qualities. These qualities, as previously discussed, may serve as the goals or the constraints of the overall optimisation process. Semantically, the BBN used to provide optimisation guidance is built with the same underlying principal node types as used by van Gurp (2003) (Section 4), which, in turn, stem from the Factor-Criteria-Metric structures defined by McCall (1994). Specifically, there are three distinct groups of nodes:

simple characteristics These nodes describe the observed characteristics of the system known to be relevant to the goals of the system optimisation.

complex characteristics Members of this group represent the knowledge of how the simple characteristics combine into more complex ones which in turn produce cumulative effect onto the system qualities.

system qualities Final layer of nodes which is created to compose the inference structure about the non-functional system characteristics pursued by the optimisation process.

It is the explicit responsibility of the designer to define which specific features of the simulation model should be processed and included as part of the data set that will be used as an input to the BBN learning algorithm. However, faced with such a task the designer may find it difficult to decide which of the features are relevant. Nevertheless, some direction can be provided in this difficult task. Specifically, the BBN that is likely to produce the best guidance should include nodes from two major groups: *principal* and *auxiliary*.

The *principal* network nodes are necessary to represent the core inference structure required to control the direction of the optimisation process. This group of nodes should include characteristics of the system that can change based on the available selection of design modification heuristics and the system qualities stated as goals of the optimisation. For example, if the designer is required to optimise scalability of a given system then the network should contain as many nodes as possible describing the issues affecting system scalability in that context.

A typical *auxiliary* network node should be determined based on the principles of a narrow quality focus. The *auxiliary* nodes are necessary primarily to ensure that the constraints of the optimisation are properly addressed within the process. Depending on the conditions imposed by the problem these constraints may be classified as *hard* or *soft*. Hard constraints ensure that the changes to the system design during optimisation do not dramatically reduce a known quality of the system. Whereas soft constraints are there to show what effects a certain course of action will have on qualities not expressly sought after as optimisation goals.

Since each node in a BBN represents a variable, which may assume a value according to the associated conditional probability distribution, it is not possible to express general relationships within the system without enumerating all the potential states every node may assume

in advance. This may make the task of creating *auxiliary* nodes difficult as it will most likely be based upon static domain expertise. However, the BBN discovery process employes a construct called a Markov Blanket (MB) (Tsamardinos et al., 2003), which helps to remove attributes that do not affect the nodes of interest and, therefore, are unnecessary.
Discovering the structure of a Bayesian network from a set of data has emerged as a major focus for research. It has been of particular interest since when source data can be represented by a time series, the edges in the graph of a Bayesian network can be used to infer likely causal relations (Spirtes et al., 2000). The actual algorithm used to *discover* the presence and orientation of links between the variables featured in the BBN at the core of the guidance framework is the Max-Min Hill-Climbing (MMHC) algorithm developed by Tsamardinos et al. (2006).
This MMHC algorithm draws upon a variety of ideas from search-and-score and local learning techniques, such as Markov Blanket discovery, to construct the skeleton of a Bayesian network corresponding to the source data and then perform a Bayesian-scoring greedy hill-climbing search to orient the edges within the graph. One of the most attractive features of the MMHC algorithm is that it has been proven to work well with the highly dimensional data sets showing that it can deal well with situations where non-functional qualities are a result of interactions between a large number of factors. Furthermore, research has shown (Tsamardinos et al., 2006, p. 30) this algorithm exhibits good scalability and accuracy when applied to learn converging sparse networks.
However, the Direct Acyclical Graph (DAG) of the BBN produced as a result of MMHC application does not represent the complete Causal Quality Model. Additional calculations must be performed to establish the probability distributions in the network. Specifically, to determine for each variable X that has a set of states S_x the probability of X being in the state $s \in S_x$ for each combination of states of its parents P_x. In order to perform this calculation an additional algorithm was developed to combine the source data from the simulation with the structural information obtained from the application of MMHC algorithm. The Section 4.3 provides a description of an example showing application of the proposed methodology to a small-scale decision scenario.

4.2.3 Guidance Methodology Synthesis

The ultimate goal of combining the elements depicted in Figure 4 is to obtain the Causal Quality Model (CQM). What the CQM effectively represents is a new, analytical view of the system that should be constructed and used in tandem with the traditionally accepted views representing static structure, processes, data and physical deployment.
Hence, the synthesis of this view should rely on the available architectural, environmental and contextual information. Specifically, to construct this view the following information is needed:

1. A set of goals and constraints: obtained from functional and non-functional requirements, business drivers, future plans and budgeting both in terms of time and money.
2. Information about system structure combined with elements deemed relevant in view of goals and constrains, these elements may include back-office workflows and customer behaviour. Simulation time can also be a very important factor and should be included as a special consideration based on the goals of optimisation.
3. Feature and metric selection must be made to identify nodes for the BBN (based on the feature and goal information), the actual structure of the DAG will be elicited using an algorithm.

4. Possible variation in characteristics of the system being optimised can also be added as part of the simulation. In this case the successive runs of simulation can be used to explore how these degrees of freedom affect the goals specified in 2.

The next section (Section 4.3) explores the issues of implementation in the context of a simple problem in order to focus on the specific of the guidance methodology and not the complexities of the optimisation problem.

4.3 Example

One of the major critiques that can be made about the examples of the BBN applications described in Section 4 is that, in both cases, the assignment of probability values for nodes affected by multiple parents is based on conglomeration of a priori knowledge held by domain experts. Although networks constructed using such information have been shown to be useful analytical tools when applied to system architectures (Trendowicz & Punter, 2003; van Gurp & Bosch, 2000), they are nonetheless exposed to the possibility of errors that can be introduced when domain experts from multiple fields are asked to quantify the combined influences effecting a specific node. The example presented in this section aims to explore the way by which the proposed architectural optimisation guidance methodology aims to address this issue in its use of Bayesian networks.

This example itself focuses on how specific knowledge about system qualities such as *Reliability* and *Performance* (shown in Table 2) can be used in combination with known details of user behaviour to reason about possible avenues for design optimisation aiming to increase the overall *Usability* of the system.

quality	**P(good)**	**P(bad)**
reliability	0.99	0.01
performance	0.86	0.14

Table 2. Likelihood values for Reliability and Performance

To this end, the process of model construction was started by creating a state machine describing an individual system user as an Agent[1]. The resultant User Agent is shown in Figure 5. This simplified version is based on data accumulated in system logs that track some aspects of behaviour exhibited by individual system users. As such this data represents only the most basic, meaningful, use-case scenario for the services offered by the system. In short, the agent represents a typical user of the search and reservation services.

Several things should be noted about the described state transitions. Firstly, transitions leading to the “Leave" state on the diagram represent instances of agents discontinuing the use of service due to poor reliability. The execution of these transitions leading to this node is controlled by a set of functions which combine the probability value associated with *Reliability* (Table 2) and the current state of the agent. Specifically, the “Leave” state can be reached from three instances of a choice function, or *branch element* node, marked by the letter ‘B’. When reached, the choice functions uses probability information to determine whether the specific instance of User Agent experiences unreliable system behaviour and if, in fact, such an occurrence is observed the function immediately activates the transition to the “Leave” state. At

[1] All hybrid models used in this and other chapters of this thesis were done using AnyLogic™ modelling tool (`http://www.xjtek.com/`).

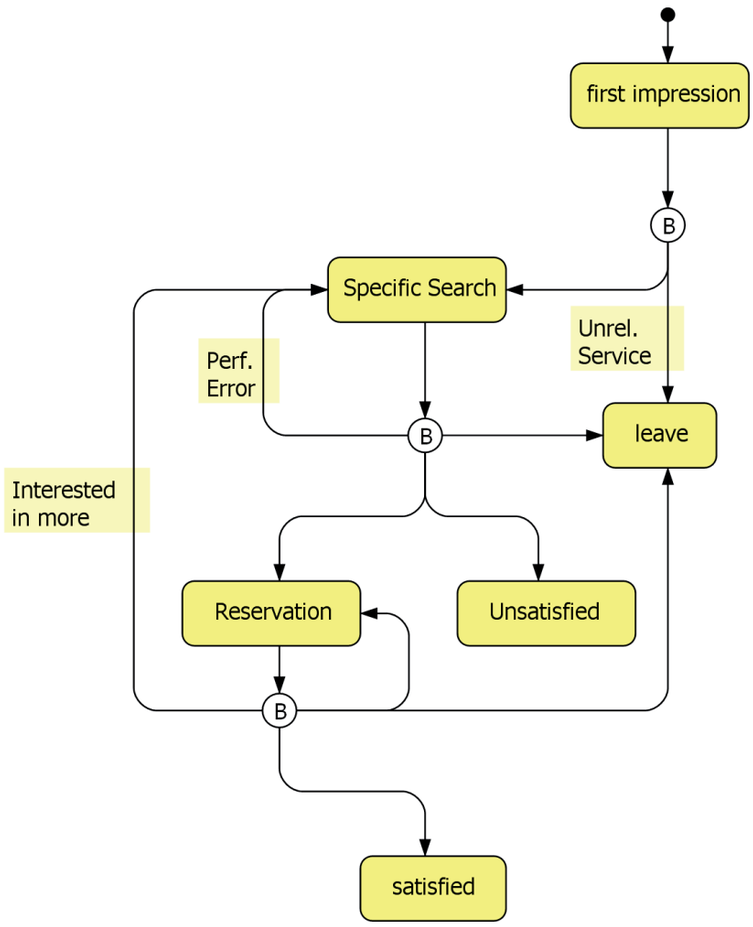


Fig. 5. Behavior Description for Useability Agent

this point the instance of the User Agent terminates operation and the system can be said to *have been unusable due to 'bad' Reliability*.

Likewise, the same choice functions also control the activation of arcs leading back to the originating node. These arcs represent retries due to poor performance. In the case of observing 'bad' *Performance* the User Agent will attempt to retry the current operation, the number of times that this will be attempted depends on the previous state of the agent. This is done with the intention of reflecting the observation that users attempting to make a reservation usually attempt more retries than the users performing searches. However, even if a given instance of the User Agent observes both 'good' *Performance* and *Reliability* it may still end up in the "Unsatisfied" state, which also exists to represent the group of system users who do not progress towards reservation.

In the context of this simple model, the measure of system *Usability* was determined by the likelihood of a User Agent reaching "Satisfied" state. To obtain observations for this measure a simulation of the behaviour exhibited by a population of 1000 User Agents was created and executed until all agents reach one of the terminating states: "Leave", "Unsatisfied" or "Satisfied". After several runs of the simulation were completed, the following numeric results were produced:

- R_t number of repeat requests due to timeouts = 222
- R_s number successful requests = 1332
- R_c number total *countable* requests = 1387
- **useability at good performance** $(1 - R_t/R_c)$ **= 0.84**
- U_f failures due to the unreliable behaviour of the system = 41
- U_t number of User agents = 1000
- **useability at good reliability** $(1 - U_f/U_t)$ **= 0.959**

A small clarification must be made to the figures above concerning *countable requests*. Due to the nature of the User agent behaviour some requests made by a specific User, even if successful, may still lead to the User abandoning the system and terminating operation by entering the "Leave" state.
The values listed above can be used to calculate the combined likelihood values, which describe the effect various states of *Reliability* and *Performance* have on *Usability* (Table 3). The values in Table 3 have been calculated by multiplying the values shown in the list of *Usability* values shown in the list above. This conditional probability information is shown in its marginalised form in the BBN depicting the structural relationships between qualities shown in Figure 6. Naturally, the specific values of 'good' for each of the qualities in question corresponds to a range of actual values for the metrics contributing to the evaluation of the corresponding quality.

reliability	good		bad	
performance	good	bad	good	bad
useable	0.80556	0.15344	0.03444	0.00659
not	0.19444	0.84656	0.96556	0.99341

Table 3. Likelihood values for Usability

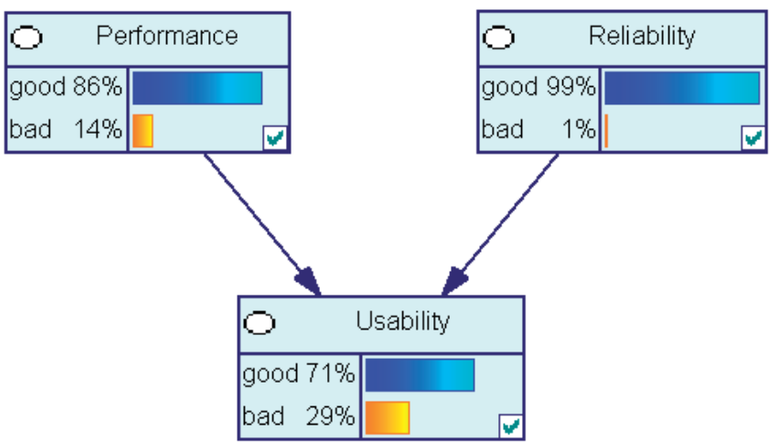


Fig. 6. The example BBN composed of *Performance*, *Reliability* and *Usability* nodes.

In this example only the Agent Based simulation paradigm was used to ensure simplicity and verifiability of simulation results by inspection. However, even at this level, the example

shows that the knowledge of the specific system and context characteristics can be used to obtain, via simulation, numerical representation of the influences exerted by these characteristics onto the emergent system qualities. Specifically, the information about user behaviour, when combined with known system behaviour, reliability and performance, contributed towards the creation of the model for the emergent quality of usablility.

4.4 Guidance Data Interpretation and Analysis

The resultant Causal Quality Model can be used for analysis in three distinct ways:

1. The structure it exhibits can be analysed.
2. The conditional probabilities at each node can be used to explore the strength of interrelationships between system qualities.
3. The underlying joint probability function can be used to propagate some *observational evidence* through the network.

The uses of these techniques individually, or in combination, can be classified into two groups: *diagnostic* and *predictive*. Although the belief network examined in the example above (Section 4.3) is simple in composition, this is merely a result of the limited amount of input information used to create the hybrid simulation that provided the learning data for the MMHC algorithm. It is therefore possible that, in the context of an optimisation problem applied to a complex system, the designer will have to draw upon a number of disparate information sources describing both the technical characteristics and the operational qualities of the system.
Given this potential breadth of information, a Causal Quality Model, like the one depicted in Figure 6, could be used as a *diagnostic* tool to uncover the presence and strength of relationships between qualities and their contributing factors. Knowledge regarding the existence and effects of these relationships can then be exploited to guide the choice of design change heuristics to either remove or strengthen some relationships or increase the likelihood of higher measurement for some qualities of interest.

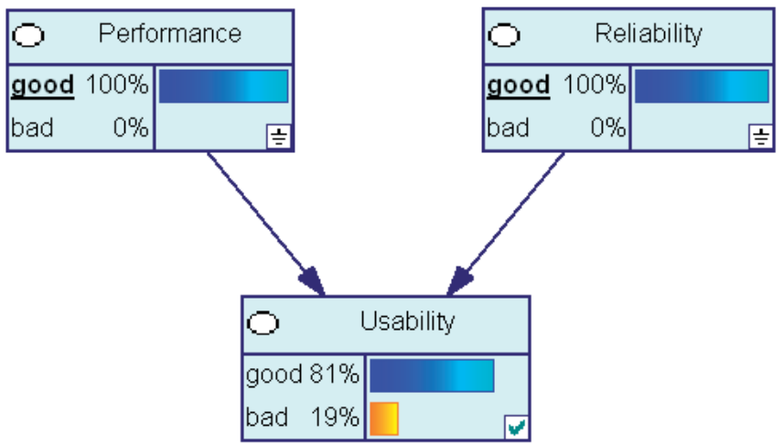


Fig. 7. The example BBN composed of *Performance*, *Reliability* and *Usability* nodes with evidence set for *Performance* and *Reliability* nodes.

As mentioned above, a Causal Quality Model can be used in a *predictive* capacity. In order to do this, the designer must provide *evidence* for one or more nodes of the BNN. In other

words, set the likelihood value for a specific state to 100%. This can be done in two ways: by manipulating either the parent or the child nodes in the causal interaction. A version of the former, as applied to the example BBN (Figure 6), is shown in Figure 7, while the latter is depicted in Figure 8.
In a case when the evidence regarding the states of parent nodes in a causal interaction is provided the Causal Quality Model can be used to determine the *likely* observed state of the child nodes. The Figure 7 shows the likelihood of 81% for the "Usability" variable to be in a state marked as 'good' when the variables for "Performance" and "Reliability" are observed to be within the bounds of what is considered 'good' for both of those qualities. This information can serve as a starting point for predicting changes that can advance the qualitative boundaries of the system.

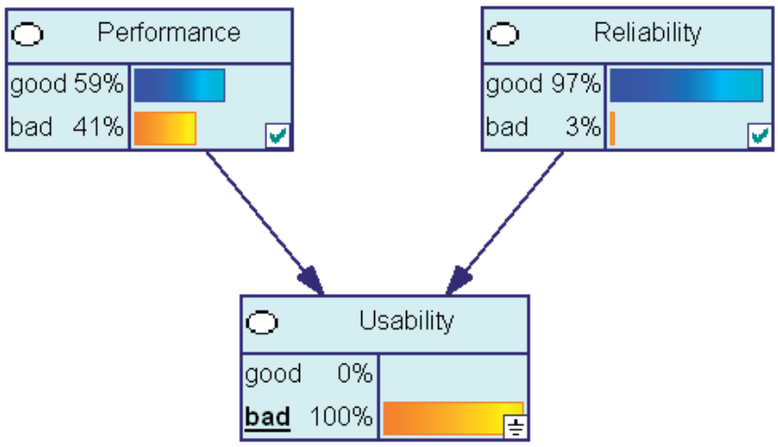


Fig. 8. The example BBN composed of *Performance*, *Reliability* and *Usability* nodes with evidence set for *Usability* node and calculated *posterior* probabilities.

However, when it comes to consideration of system qualities it may be more interesting to consider the negative case. Figure 8 depicts the posterior probabilities of observations for system's "Performance" and "Reliability" given evidence that "Usability" is observed to be 'bad'. It can be seen that, for low values of "Usability", while measure for "Performance" drops considerably, the measure for "Reliability" shows little change: only 2%. Based on these results the designer can conclude that under the current configuration, and based on known user behaviour patterns, the "Usability" of the system is highly sensitive to the "Reliability" of the system. The combined information of the original BBN and the results of setting evidence shown in Figures 7 and 8 give the designer a new perspective on the possible direction of potential design modifications.
Overall, the *diagnostic* and *predictive* uses for the Causal Quality Model can provide a valuable insight into the aspects of the problem and their relationship to the design change options made available to the designer. As a result, it is possible for the designer to understand clearly which system features contribute to the overall qualities of the system. Furthermore, the designer is able to identify which changes to those system features are most likely to render beneficial improvements in the system qualities, the goals of the optimisation process. This clarity also provides the designer with a new from of system representation, one which may prove useful both in decision making and as a communication tool capable of relating technical concepts to the non-technical stakeholders of the system.

5. Conclusion and Future work

The design optimisation guidance methodology aims to aid the designer in directing the overall system optimisation process. One of the major difficulties of providing such guidance is the nature by which this optimisation process is advanced. Specifically, the designer is essentially incapable of affecting the qualities directly. Instead, he or she is forced to consider a set of choices targeting the specific features of the design contributing towards achievement of desirable system qualities. As a result, since a single choice could affect multiple qualities, this introduces a requirement for guidance to provide the designer with understanding of the causal relationships existing in the system.

Achieving this involves the study of assumptions held by the designer and other stakeholders, the relevance of existing knowledge and the accuracy of possible predictions. The fusion of simulation modelling and the BBNs can serve as tool of such study as its aim is to provide a tangible link between the way in which the system is structured and its observed levels of quality. Additionally, by combining the hybrid simulation model with BBN discovery algorithm we managed to obtain a much more repeatable output that is validated against encoded assumptions and is less prone to human error.

However, the method's success relies greatly on validity of the model and clarity of the BBN representation. To this end we have found that the simulation model should be built in an incremental manner using a variety of information sources and explicit encoding of assumptions help by the participants. Consequently, the extracted BBN plays a dual role both as a guidance tool and a model verification tool as the conditional probabilities it displays can quickly highlight inconsistencies within the model.

The results presented herein warrant further investigation along four major axis:

- further research is needed to help the designer with choice of quality factors and criteria that contribute to the nodes of the CQM;
- a taxonomy of simulation primitives needs to be developed to aid the designer with construction of hybrid simulation models;
- additional research is needed to examine how various BBN discovery algorithms perform on the types of simulation output produced by models of systems from different domains;
- studies should be conducted into the various stochastic methods of optimisation such as Cross-Entropy (Caserata & Nodar, 2005) that could be implemented based on the outcomes of BBN use for qualitative applied over a succession of system development cycles.

Finally, the development of this approach to guidance should be used to construct a fully fledged decision support and optimisation framework described in Section 3.

6. References

Almeder, C. & Preusser, M. (2007). A hybrid simulation optimization approach for supply chains, *Proceedings EUROSIM 2007*, Ljubljana, Slovenia.

Borshchev, A. (2005). System dynamics and applied agent based modeling, *In Proceedings of International System Dynamics Conference*, Boston, MA.

Borshchev, A. & Filippov, A. (2004). From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools, *In Proceedings of The 22nd International Conference of the System Dynamics Society*, Oxford, England.

Bosch, J. & Bengtsson, P.-O. (2001). Assessing optimal software architecture maintainability, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering*, IEEE Computer Society, p. 168.

Bucci, G. & Maio, D. (Sep 1982). Merging performance and cost-benefit analysis in computer system evaluation, *IEEE Computer* **15**(9): 23–31.

Bucci, G. & Streeter, D. N. (1979). A methodology for the design of distributed information systems, *Commun. ACM* **22**(4): 233–245.

Bunge, M. (1979). *Treatise on Basic Philosophy: Volume 4: Ontology II: A World of Systems*, 1st ed. edn, Springer.

Caserata, M. & Nodar, M. C. (2005). A Cross-Entropy Based Algorithm for Combinatorial Optimization Problems, *European Journal of Operational Research* .

Chen, H.-M., Kazman, R. & Garg, A. (2005). Bitam: An engineering-principled method for managing misalignments between business and it architectures, *Science of Computer Programming* **57**(1): 5–26.

Coit, D. & Konak, A. (2006). Multiple weighted objectives heuristic for the redundancy allocation problem, *Reliability, IEEE Transactions on* **55**(3): 551–558.

Coit, D. W. (2001). Cold-standby redundancy optimization for nonrepairable systems, *IIE Transactions* **33**(6): 471–478.
URL: *http://www.springerlink.com/content/q96wkpuqme2fe3ju*

Coit, D. W. & Smith, A. E. (1996). Reliability optimization of series-parallel systems using a genetic algorithm, *IEEE Transactions on Reliability* **45**: 254–260.

Denford, M., Leaney, J. & O'Neill, T. (2004). Non-functional refinement of computer based systems architecture, *Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the*.

Diaconescu, A. & Murphy, J. (2005). Automating the performance management of component-based enterprise systems through the use of redundancy, *ASE*, pp. 44–53.

Folmer, E. & Bosch, J. (2005). Case studies on analyzing software architectures for usability, *EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE Computer Society, Washington, DC, USA, pp. 206–213.

Forrester, J. W. (1991). *The Systemic Basis of Policy Making in the 1990s*, Sloan School of Management, MIT, Boston, MA, chapter System Dynamics and the Lessons of 35 Years.

Gokhale, S. S. (2004). Cost constrained reliabilty maximization of software systems, *Reliability and Maintainability, 2004 Annual Symposium - RAMS* pp. 195–200.

Grunske, L. (2003). Transformational patterns for the improvement of safety properties in architectural specification, *Proceedings of The 2nd Nordic Conference on Pattern Languages of Programs (VikingPLoP 03)*.

Grunske, L. (2006). Identifying "good" architectural design alternatives with multi-objective optimization strategies, *ICSE '06: Proceedings of the 28th international conference on Software engineering*, ACM, New York, NY, USA, pp. 849–852.

Grunske, L., Geiger, L., Zündorf, A., Van Eetvelde, N., Gorp, P. V. & Varro, D. (2005). *Model-driven Software Development - Volume II of Research and Practice in Software Engineering*, Springer, chapter Using Graph Transformation for Practical Model Driven Software Engineering.

Haddawy, P. (1999). An overview of some recent developments in bayesian problem solving techniques, *AI Magazine Special Issue on Uncertainty in AI*.

Kazman, R., Barbacci, M., Klein, M., Carrière, S. J. & Woods, S. G. (1999). Experience with performing architecture tradeoff analysis, *ICSE '99: Proceedings of the 21st international conference on Software engineering*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 54–63.

Leaney, J., Denford, M. & O'Neill, T. (2004). Enabling optimisation in the design of complex computer based systems, *ECBS '04: Proceedings of the 11th IEEE International Conference and Workshop on Engineering of Computer-Based Systems*, IEEE Computer Society, Washington, DC, USA, p. 69.

Maxwell, C. (2007). *Representing Heuristics for Architectural Optimisation*, PhD thesis, University of Technology, Sydney, Faculty of Information Technology.

Maxwell, C., O'Neill, T. & Leaney, J. (2006). A framework for understanding heuristics in architectural optimisation, *13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2006)*, Potsdam, Germany.

Maxwell, C., Parakhine, A., Denford, M., Leaney, J. & O'Neill, T. (2005). Heuristic-based architecture generation for complex computer systems, *12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2005)*.

McCall, J. A. (1994). *Encyclopedia of Software Engineering*, Vol. 2 O-Z, John Wiley & Sons, New York, chapter Quality Factors, pp. 1085–1089.

MDA Guide (2003). omg/2003-06-01. Model-Driven Architecture Guide, Version 1.0.1.

Parakhine, A. (2009). *Architectural Optimisation Guidance of Complex Computer-Based Systems*, PhD thesis, University of Technology, Sydney, Faculty of Information Technology.

Pearl, J. (1986). A constraint-propagation approach to probabilistic reasoning, *Uncertainty in Artificial Intelligence*, Elsevier Science, North-Holland, Amsterdam, pp. 357–369.

Peterka, V. (1981). *Trends and Progress in System Identification*, Pergamon Press, chapter Bayesian Approach to System Identification, pp. 239–304.

Popkov, T., Karpov, Y. & Garifullin, M. (2006). Using Simulation Modeling for IT Cost Analysis, *Technical report*, Distributed Computing and Network Department, St.Petersburg State Technical University, St.Petersburg, Russia.

Rowe, D., Leaney, J. & Lowe, D. (1998). Defining systems evolvability - a taxonomy of change, *International Conference and Workshop: Engineering of Computer-Based Systems (ECBS '98)*.

Sharma, V. S. & Trivedi, K. S. (2005). Architecture based analysis of performance, reliability and security of software systems, *WOSP '05: Proceedings of the 5th international workshop on Software and performance*, ACM Press, New York, NY, USA, pp. 217–227.

Spirtes, P., Glymour, C. & Scheines, R. (2000). *Causation, Prediction, and Search*, The MIT Press.

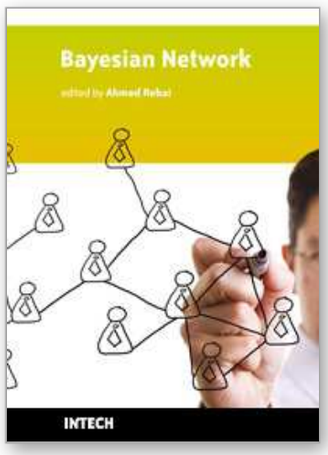
Trendowicz, A. & Punter, T. (2003). Applying bayesian belief networks for early software quality modeling, *IESE-Report No. 117.03/E* . Fraunhofer IESE.

Tsamardinos, I., Aliferis, C. F. & Statnikov, A. (2003). Algorithms for large scale markov blanket discovery, *The 16th International FLAIRS Conference*, St. Augustine, Florida, USA.

Tsamardinos, I., Brown, L. E. & Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm, *Machine Learning* **65**(1): 31–78.

van Gurp, J. (2003). *On The Design & Preservation of Software Systems*, PhD thesis, Rijksuniversiteit Groningen.

van Gurp, J. & Bosch, J. (2000). Automating software architecture assessment, *In Proceedings of Nordic Workshop on Programming and Software Development Environment Research 2000 (NWPER 2000)*, Lillehammer, Norway.



Bayesian Network
Edited by Ahmed Rebai

ISBN 978-953-307-124-4
Hard cover, 432 pages
Publisher Sciyo
Published online 18, August, 2010
Published in print edition August, 2010

Bayesian networks are a very general and powerful tool that can be used for a large number of problems involving uncertainty: reasoning, learning, planning and perception. They provide a language that supports efficient algorithms for the automatic construction of expert systems in several different contexts. The range of applications of Bayesian networks currently extends over almost all fields including engineering, biology and medicine, information and communication technologies and finance. This book is a collection of original contributions to the methodology and applications of Bayesian networks. It contains recent developments in the field and illustrates, on a sample of applications, the power of Bayesian networks in dealing the modeling of complex systems. Readers that are not familiar with this tool, but have some technical background, will find in this book all necessary theoretical and practical information on how to use and implement Bayesian networks in their own work. There is no doubt that this book constitutes a valuable resource for engineers, researchers, students and all those who are interested in discovering and experiencing the potential of this major tool of the century.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen