Design Guidance Using Simulation-Based Bayesian Belief Networks

Artem Parakhine[1,2], John Leaney[2] and Tim O'Neill[2]
1. Faculty of Information Technology
2. Institute for Information and Communication Technologies
University of Technology, Sydney
aparakhi@it.uts.edu.au
John.Leaney@uts.edu.au
tim.oneill@avolution.com.au

Abstract

In our work, the task of complex computer-based system design optimization involves exploration of a number of possible candidate designs matching the optimisation criteria. However, the process by which the possible candidate designs are generated and rated is fundamental to an optimal outcome. It is dependent upon the set of system characteristics deemed relevant by the designer given the systems requirements. We propose a method which is aimed at providing the designer with guidance based upon description of the possible causal relationships between various system characteristics and qualities. This guidance information is obtained by employing principles of multiparadigm simulation to generate a set of data which is then processed by an algorithm to generate a Bayesian Belief Network representation of causalities present in the source system. Furthermore, we address the issues and tools associated with application of the proposed method by presenting a detailed simulation and network generation effort undertaken as part of a significant industrial case study.

1. Introduction

The contemporary state of art for system design and construction incorporates a variety of technologies and methods which are aimed at ensuring that specific non-functional qualities such as scalability and modifiability are observed when large sets of functional elements are integrated to form a single complex system. However, there are no definitive guidelines concerning which specific factors within the design may be affecting the qualities most favoured for acquisition. Furthermore, as the design space is explored over time and new functionality is added the system may start to exhibit unexpected characteristics and new, previously omitted, factors may start to yield influence over qualities of interest, or perhaps a conflict between a number of qualities may arise.

The unexpected characteristics of the target system may have their roots in its structural properties, the unforeseen shortcomings of its comprising elements or in assumptions made by the designer, the latter being a possible reason for decisions made which lead to the former. In some cases the assumptions made by the designer may focus on a limited range of factors affecting the system qualities and thus the set of candidate solutions created based on these assumptions might not address all changes necessary to ensure non-functional qualities of the target system. In a context of a given system the designer defines the factors of interest based on empirical observation on the said system, various modelling approaches or previous personal experiences.

Consequently, it is imperative to ensure that the process of system design possesses faculties to determine possible causal relationships between specific properties of the system and its non-functional qualities. It should be equipped with a way to manage possible conflicts and prevent any single non-functional quality from taking overwhelming precedence over others.

Previously [8] we proposed a *heuristic*-based system design and optimisation framework. A brief overview of the framework for the proposed process is shown in Figure 1. As shown on the diagram the proposed framework contains a feedback loop introduced to allow iterative application of *heuristic*-encoded design decisions to find a solution which represents an optimal compromise on competing system qualities.

To achieve a favourable outcome the process first requires a baseline design as well as a list of goals and constraints. The original design is then evaluated to establish a base line understanding of its capabilities and shortcomings. Consequently this information is used by the "Optimisation Guidance" component to generates inputs which can be

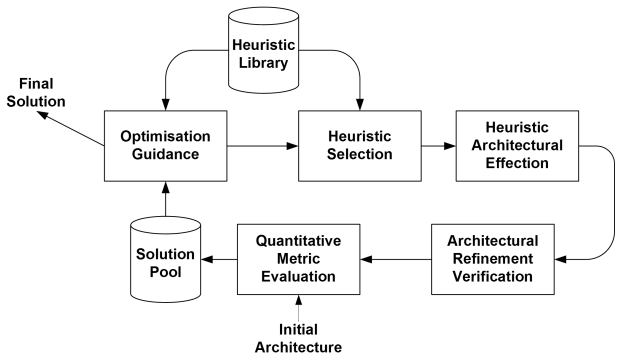

Figure 1. Overview of heuristic-based optimisation framework

used by the "Heuristic Selection" module to determine which heuristics should be considered as candidates for future application. The format and nature of information pertaining to system changes is stored as *heuristics* in the "Heuristic Library" is described by Maxwell [7]. Once a candidate solution has been generated, the "Architectural Refinement Verification" [5] module validates that the outcome functionally corresponds to the original design.

The modular structure evident in the proposed framework was chosen to ensure that various areas of it can 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 quality evaluators.

In the context of the framework the "Optimisation Guidance" component is charged with tying together the design goals, candidate solutions, metrics and heuristics. To achieve that the component must be able to determine which structural or parametric properties of the architecture have to be addressed using available *heuristics* to ensure progress towards the stated goals under the specified constraints. Previously [10] we proposed Bayesian Belief Networks (BBNs) supported by Multi-Paradigm Simulation (MPS) as a possible avenue of obtaining a method by which the guidance component will be able to create specific design suggestions that fulfil the aforementioned criteria.

The approach used to realise the proposed methodology employing BBNs and MPS as the main decision aid is the focus of this paper. The content is structured as follows: Section 2 provides background information on the concepts related to BBN generation. Following that, Section 3 contains details of an example where the proposed method has been applied with success and Sections 4 and 5 provides a discussion of conclusions and future work.

1.1. The Need for Guidance

Having being given or specified the non-functional requirements such as *Scalability* or *Modifiability* the designer attempting to achieve them in the context of some complex computer-based system faces the problem of having to define the set of variables which affect the target goal. Consider the diagram in Figure 2. It depicts possible causality arcs which show the specific factors affecting Scalability and Modifiability properties in the context of a hypothetical system.

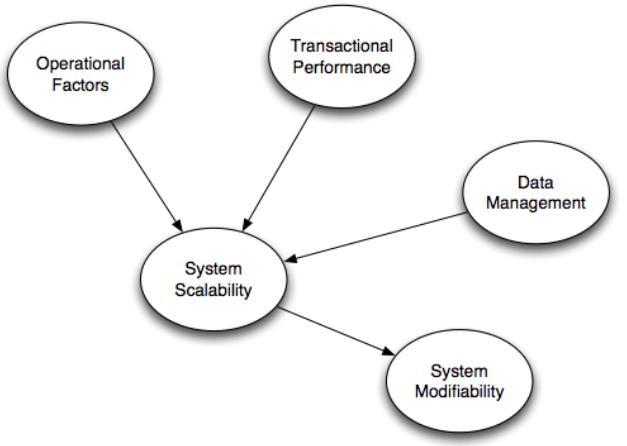

Figure 2. Possible causalities in a hypothetical system

On the diagram the nodes represent the following measurable system characteristics:

- Operational Factors - How does an increase in system size consume operational resources? If the system is to be split geographically or, perhaps, its structure to be modified to create multiple nodes performing the same type of task in parallel then how will it affect the support staff size, costs incurred due to changed hardware and network requirement, ease of feature deployment and roll-back? All these factors are omitted in Figure 2 and would appear as causal parents of the *Operational Factors* node.
- Transactional Performance - What is the transactional capacity of the system? The metric for this quality may be a simple *number of successful transactions* per *unit of time*, however in real systems not all transactions are created equal and therefore the metric should reflect that in some way.

- Data Management - Is the system capable of efficient data management? In this case "efficiency" of management refers to how a data access policy affects the scalability of the system. The view of the data modularised based on multiple levels of importance is considered more efficient then a strictly monolithic view [4].
- Scalability - Overall scalability of the system expressed as a direct result of the combined observations of its causal primitives.
- Modifiability - The ease with which new features can be added to this system. In this hypothetical case a good overall measure of *Scalability* means that each new feature can be developed without spending too much time and resources optimising the existing architecture to allow for increase in transactions or resources consumption.

The problem is that the architect may not be completely aware of the relationships represented in Figure 2. The architect may have access to information about the structure of the system modules (conceptual/source/deployment), may possess resource consumption telemetry obtained during system operation, information about typical user behaviour, knowledge of operational and development procedures, etc. However in order to combine all of these pieces of knowledge into a coherent network of interrelated variables the architect will have to employ ad hoc processes based on domain knowledge and previous experience. As a result it is possible that a dependency between *Data Management* properties of the system and *Scalability* may be missed, leading the architect to concentrate on *Transactional* and *Operational* aspects of the system design. Given time and resources, there is no doubt that the architect will be able to succeed at vastly improving these aspects and, by implication, the *Scalability* of the system. However, eventually the *Data Management* problems would surface and negate some or, in the worst case scenario, all advantages gained in other areas.

Hence the aim of the guidance approach is to provide the architect with an analysis tool that could be used to explore the effects particular changes would have on the interrelated properties of the system.

We envisage that the process would require the architect significant historical understanding the nature of the system. From the available information the architect should be able to create one or more models which describe the system operation, behaviour and structure. Following that, these models are tested with various input data sets which are mapped to possible event scenarios considered by the architect. A typical scenario may be based on external factors (e.g. "if Users really love a certain feature") or internal drivers such as "if the product range increases by a factor of 10 or if the company is to expand to Europe". During and after simulating the system under a variety of conditions the guidance component should collect statistics on the state of various variables of interest. Once a sufficiently big set of data is obtained the guidance component can employ an automatic structure discovery algorithm discussed further to build a BBN which can be used as a decision tool to further the design activities.

2. Supporting Elements for Design Guidance

The efficiency and overall success of the proposed design guidance methodology are based on two supporting concepts: multi-paradigm or hybrid simulation and automatic discovery of causality relationships between systems characteristics and its qualities. The former is required to provide the designer with a facility capable of modelling the current as well as possible structural and behavioural characteristic of the system. The latter of the two is necessary to ensure that data collected during the successive runs of modelling and exploration phase is aggregated into a meaningful graph of causalities which can be used to understand the drivers behind the specific non-functional qualities of the system.

The interaction between elements described above is depicted in Figure 3. Two elements interact to produce the most comprehensive picture of goings-on in the system with respect to the requirements and constraints imposed onto the designer. The aim is to ensure that no factor is omitted from exploration and consequent analysis unless the designer wishes it so. 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 plausible outside elements such as user base or external processes.

2.1. Hybrid Simulation

In order to achieve such flexibility the simulation must be able to deal with concepts from three major paradigms which currently dominate the field of simulation modelling [3]:

- System Dynamics - required to be able to represent effects of policy introduction or modification at the highest levels of abstraction as well as analysis of trends and other system *properties-of-the-whole*.
- Discrete Events - required to understand issues associated with utilisation of various resources available to the system as well as effects of various scheduling decisions.

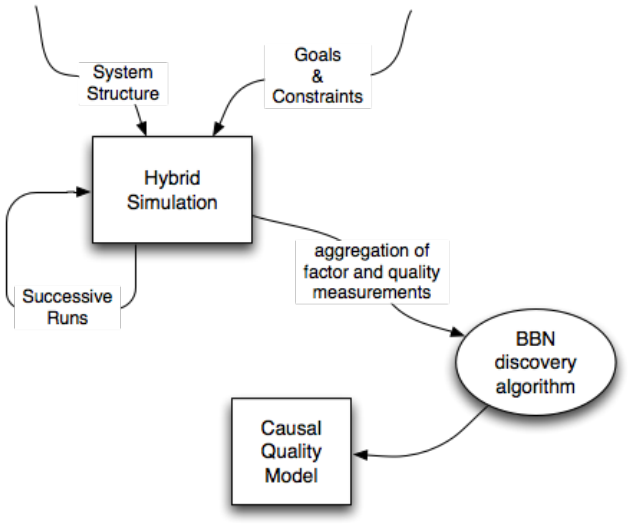

Figure 3. Interaction of guidance support elements

- Agents - this paradigm allows to simulate elements which can only be meaningfully represented as active objects with individual, purposeful behaviour.

The combination of the modelling techniques is proposed by Borshchev [3], [2] with special emphasis made on Agent-based modelling to accentuate both its flexibility and pragmatism. Contrariwise to other approaches from the list above, Agent-based modelling is, at its core, based on principles of decentralisation. In order to use this paradigm, the designer is not required to have the precise knowledge of the way system operates. Instead, this approach allows to achieve *emergence* of global system properties by examining the interactions of various system elements over time. Therefore, this approach may be more pragmatic in situations where the complete information about the system is not available.

The multi-paradigm or hybrid simulation approach which has been proposed by Borshchev [3], possesses both the 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 optimisation needs of problems ranging from supply chain management [1] to enterprise IT cost analysis [11].

The overall aim of the method is to provide a recommendation on a list of changes considered by the designer. To this end the simulation approach should is necessary to allow testing various quality scenarios similar to the ones mentioned in Section 1.1. Once the source model of the original system has been built it becomes possible to translate partial use case scenarios targeted, for example, at exploring the *scalability* of the system into a set of valid inputs for the simulation of the model.

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. Furthermore, it is up to the designer to determine which factors should be included into the observation set. Additionally, the designer is required to provide some form of measurement methodology for the system qualities which will be applied by the framework after each successive run of the simulation. 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.

2.2. Causality Discovery and Representation

In Figure 3 the causal discovery part of the method is represented by "BBN discovery algorithm" entity. In this context we are referring to an automatic way of learning the topology of a Bayesian Belief Network from a large set of observational data.

A Bayesian Belief Network (BBN) is a directed acyclic graph, a mathematical construct that compactly represents a joint probability distribution for a set variables [6]. However, this approach to representation allows one to show how the system-wide probability distribution is broken down into a set of local distributions in a subset of variables. This gives the BBNs the ability to dramatically reduce the amount effort required to specify a joint probability distribution in a system with sparse interaction between the observed variables. Furthermore, the localisation feature of BBNs allows one to examine partial probability distributions associated with some specific chain of causality which may be of particular interest to the designer. Due to the aforementioned characteristics, BBNs are frequently employed for modelling domain knowledge in Decision Support Systems [14].

In the context of our methodology a BBN elucidated from the simulation output data is used as a Causal Quality Model of the system being examined. In this capacity, its function 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.

There have been few examples when BBNs were used to represent the causal relationships between various at-

tributes of a system and its qualities. Gurp [15] proposed a BBN which was inspired by McCall's Factor-Criteria-Metric model [9]. Trendowicz [13] on the other hand proposed an iterative process by which the final structure of the BBN representing the Causal Quality Model is built over time based on heavy involvement of domain experts and an extensive data collection effort. However, the result of both Gurp's and Trendowicz's work is a BBN with a static structure targeted at re-use for a software product line.

In a large system with multiple observed variables which influence multiple system qualities, the effort involved in building a single static BBN that encompasses all possible causal relationships manually is very costly from an engineering standpoint. Additionally, the resulting static BBN structure has to be validated every time the system is changed or a new variable is introduced. Therefore it is desirable, for systems that need to address a broad range of problems, to be able to dynamically establish the causality links between factors and qualities.

The problem of learning the structure of a Bayesian network from a set of data has emerged as a major focus for research, since under the conditions when source data can be represented by a time series the edges in the graph of a Bayesian network can be used to infer possible causal relations [12]. In order to build the Causal Quality Model in our framework we propose to use the Max-Min Hill-Climbing (MMHC) algorithm developed by Tsamardinos et al. [14]. This 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 from the biomedical domain of research [14].

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 have to be performed. 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 a new algorithm was developed to combine the source data from the simulation with the structural information obtained from the application of MMHC algorithm. In the sections to follow we provide a detailed description of an example application of the proposed methodology.

3. Example Application

The problem described in this section is based on an actual situation faced by a medium sized company operation in the domain of on-line travel servicing mostly the USA market. The company has between 50 and 100 employees in spread over two offices situated on the opposite sides of the Pacific Ocean. Due to the fact that the core business of the company is travel, the level of demand for its services steadily increases through the year until it peaks in summer (June - August). It is during the middle stages of this gradual increase that the production system started to sporadically experience a failures that can only be described as catastrophic. The company relies wholly on Open Source products to deliver its services.

The service provided by the company is supported by the system is comprised of 3 types of nodes:

- Service type processes the users request and generates one or more requests to the Database node. There are several nodes of this type in the system. The system also remembers which node accepted the first request from a new user and routes all following requests from that user to the same *Service* node. This node uses a Java application server.
- Database type there is only one node of this type in the system. It acts as a centralised repository of data accessed by service nodes.
- Routing type accepts users request and forwards them to the appropriate *Service* node. There is only one node of this type in the production system. This node employs a simple round-robin policy to distribute requests only to *Service* nodes it determines to be operational.

It was known that the system was failing when one of the *Service* nodes would stop operating due to memory shortage. Consequently the router would detect that the node has failed and would stop sending it both requests from new users and new requests from users already registered with the failed node. This, in turn, would increase the pressure on the *Service* nodes that were still operational. Sometimes a sharp increase in user requests would cause another *Service* node to fail. At this point the overall system would become unstable and, unless the failed nodes were promptly restarted, would degrade to the point at which quick recovery was impossible and a complete restart was required.

Additionally, the following items of information were known about the system operation generally and at the time of failure:

1. The Database node never suffered any degradation of service .
2. From testing and empirical observation it wasps known that a *Service* node can process up to 50 requests concurrently without experiencing any memory consumption or CPU utilisation problems or serious performance degradation.

3. It has been observed that the *Service* node fails once it occupies all of the available memory at which point the specifics of JVM implementation used by the *Service* node cause the CPU usage to increase to the available maximum. It was also known that immediately before the failure the CPU utilisation is between 10% and 20% of available capacity.
4. The only time memory usage was observed to increase significantly (up to 30% of available capacity) in a short period of time (less then 15 minutes) was immediately after the *Service* node has been restarted. After that the memory increase was gradual until allocated capacity has been exhausted. Furthermore, memory consumption monitoring has shown that *Service* nodes reached 95-99% memory consumption after about 4 hours and continued to operate successfully for up to 24 hours.
5. The main non-functional requirement posed by the business owners of the system has been to ensure fast user experience. To fulfil this requirement the technical owners of the system introduced aggressive caching at various levels of the system. However only one of those caches was based in memory of the *Service* node.

The problematic situation described above was causing massive reduction in the availability of the system. The technical owners of the system have posed a number of theories to explain the recurring failure. The theories proposed ranged from unexpected user behaviour to performance problems to a banal memory leak due to programming error. These theories were proposed based on the way all available data was interpreted by various technical owners of the system. Obviously, each of the owners had a bias towards his personal experience and parts of the system he knew.

At this point the only method by which a positive correlation between system properties and its availability characteristics could be found was by long and costly process of trial an error performed on a scaled down copy of the production environment. In the next section we will describe the method, and explain its application to this case study.

3.1. Model

The proposed methodology consists of five distinct steps: creation and simulation of the model, conversion of the simulation output into format appropriate for causal discovery algorithm, application of the algorithm to obtain a DAG structure of the BBN, calculation of the probability distribution values from the source data based on DAG structure, and finally, creation of a visual representation of the network which can be used for analysis.

The first step in the proposed method is to build a model of the system which can be used to explore the scenario of interest. In this case we were interested in simulating the failure scenario of a single server and exploring the factors that may contribute towards the server's eventual demise. To build the model we chose to use AnyLogic tool (http://www.xjtek.com/) which provides facilities for creation of hybrid simulations.

The Figure 4 shows the discrete event model of the *Service* node. In the diagram *enter* and *Exit* elements are used to communicate with agent entities from the collection denoted by label *user*. Periodically, an event labelled *varyUserAmount* is fired at which point the function *userInFlux* is used to determine how many new agents of type User should attempt to use the service. The values returned by the *userInFlux* function are based on the survey of real time usage information collected in the production environment.

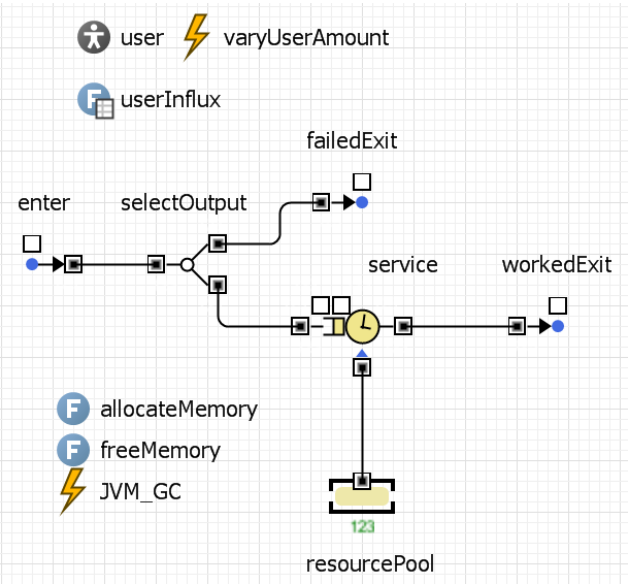

Figure 4. AnyLogic screen shot of the *Service* node model

Every time an agent tries to communicate with the service the type of the sent message is used in the *allocateMemory* function to determine how much memory is required to service it and attempt to reserve this memory for the service. If the latter operation fails the *JVM_GC* event is fired to clear out unused memory as per the standard behaviour of the Java Garbage Collector. Should the Garbage Collect fail to free enough memory the message is marked and the *selectOutput* entity routes it to *failedExit* at which point it is translated back into agent semantics. However, if memory has been successfully reserved then the message

is added to the queue of the *service*. The *service* entity allocates a processing resource from the *resourcePool* for each message which is then processed over time *t*. Once *t* has expired the resource is returned to the pool and the message is communicated back to its originating agent via *workedExit* entity.

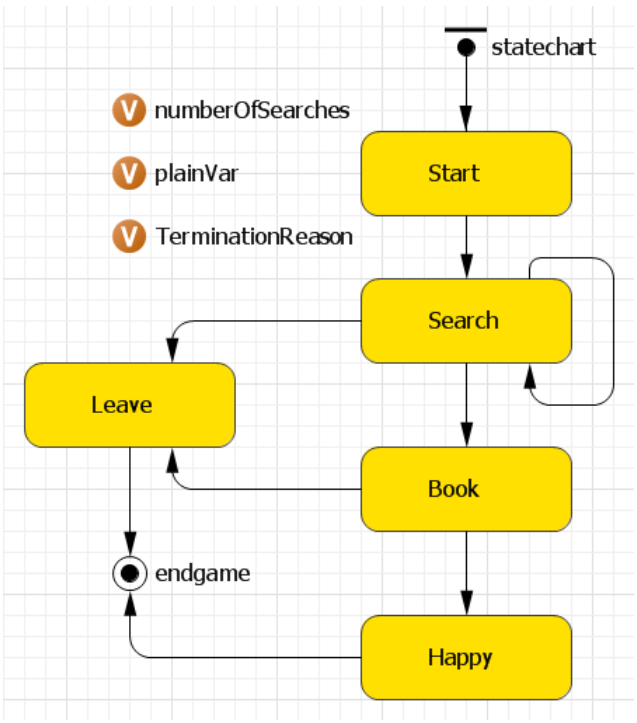

Figure 5. AnyLogic screen shot of the User agent model

The state diagram which defines the behaviour of the *User* agent attempting to use the *Service* node is depicted in Figure 5. The Users submit two types of messages: Search and Book. It was known from the survey of user behaviour that only 1.5% of all requests are of type Book. To reflect that the *User* agent was built to repeat the *Search* request numerous times before proceeding to send the *Book* message to the *Service* node. Should the agent receive an error message issued due to the fact that the service was unable to obtain enough memory or should the request time out the agent will leave.

The simulation was performed a number of times and it was decided that the following metrics should be collected:

- Users: number of happy, number of unhappy due to search, number of unhappy due to booking problems, total number;
- Memory: JVM Tenured Memory (long-living objects) left and JVM YoungGen Memory (short-lived objects) left;
- Resources: CPU, the size of the queue to access the *service* and the number of active threads (occupied resources from the *resourcePool*;

The measurements were collected periodically during runs and stored into an external file. Once the simulation has been stopped, the aggregated data has been converted from scalar values into state values based on the range of values observed for each variable. For example, CPU has been broken down into 10 state values each corresponding to a 10% range. Following that the state data has been used as an input for causal discovery algorithm.

3.2. Causal Discovery

The implementation of the Max-Min Hill-Climbing algorithm we used is distributed as part of the Causal Explorer tool kit written for MATLAB software package and provided by its authors [14] (http://discover1.mc.vanderbilt.edu/discover/public/). The results of MMHC application to the data collected from simulation were represented by an adjacency matrix where the value of 1 in the *i*th row and *j*th column showed that *i* is a parent of *j*. Based on the adjacency matrix and the specific probability distribution values collected from the simulation output we were able to generate an XML document in a format suitable for use in the GeNIe environment (http://genie.sis.pitt.edu/) which provides a good quality interface for manipulation and analysis of Bayesian networks. The resultant BBN is shown in Figure 6.

From the network shown in Figure 6 it can be clearly visible that the greatest influences on memory consumption (*jvm tenured left*) come from *queued* and *happy users* nodes. The first arc of influence is supported by the knowledge that memory consumption increased during short-lived spikes in the user activity. However, the arc leading from the *happy users* node to the *jvm tenured left* node required more investigation. Since the main way a *User* agent can affect the memory consumption by the *Service* is by initiating search requests, our attention naturally turned towards the caching facilities of the system. After some investigation it was discovered that a number of individual decisions made by different developers over the lifetime of the system have lead to a very unstructured caching policy which resulted in a large number of aggregated results duplicated under certain conditions of user behaviour that emerged during the seasonal increase in load. Once a new, better structured, caching policy has been implemented the system was able to maintain stability for the desired period of time.

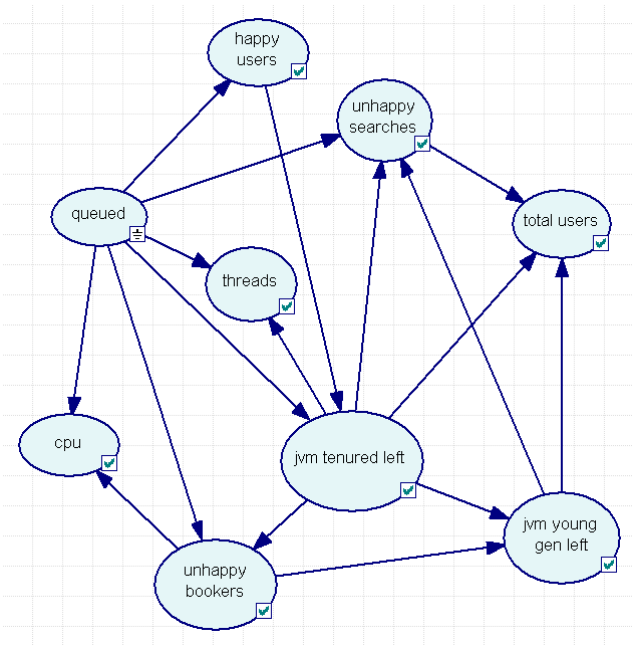

Figure 6. GeNIe screen shot of the BBN produced from the simulation results

4. Conclusions and Future Work

In this paper we have explored a design guidance methodology based on a combination of hybrid simulation of the system coupled with Bayesian structure learning aimed to discovering possible causal relationships that may link system attributes and non-functional qualities. We have also shown a practical application of the proposed methodology and the possible benefits that it can render.

Following the discussion of in this paper, we believe the next step is to further develop the modelling and Bayesian structure learning aspects of the framework with the aim of broadening the range of possible applications. Additional study to be undertaken into the effects which variations of the MMHC algorithm parameters have on the emitted structure of metric-criteria-quality dependencies in various system.

Also, it is necessary to further develop a software platform that can support the simulation, causal discovery and provision of visualisation required for successful operation of the guidance framework. We also aim to incorporate empirically collected heuristics that can be used to guide variation of various system factors during successive simulation runs. Ultimately, the aim is to create a comprehensive environment which can be used by the system designer to perform architectural optimisation.

References

[1] C. Almeder and M. Preusser. A hybrid simulation optimization approach for supply chains. In *Proc. EUROSIM 2007*, Ljubljana, Slovenia, 9th-13th September 2007.

[2] A. Borshchev. System dynamics and applied agent based modeling. In *In Proceedings of International System Dynamics Conference*, Boston, MA, July 2005.

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

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

[5] M. Denford. *Practical Architecture-Based Refinement of Computer Based Systems*. PhD thesis, Faculty of Engineering, University of Technology, Sydney, 2005.

[6] P. Haddawy. An overview of some recent developments in bayesian problem solving techniques. In *AI Magazine Special Issue on Uncertainty in AI*, July 1999.

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

[8] C. Maxwell, A. Parakhine, M. Denford, J. Leaney, and T. O'Neill. Heuristic-based architecture generation for complex computer systems. In *12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2005)*, 4th-7th April 2005.

[9] J. A. McCall. *Encyclopedia of Software Engineering*, volume 2 O-Z, chapter Quality Factors, pages 958–969. John Wiley & Sons New York, 1994.

[10] A. Parakhine, J. Leaney, and T. O'Neill. Application of bayesian networks to architectural optimisation. In *14th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2007)*, 2007.

[11] T. Popkov, Y. Karpov, and M. Garifullin. Using simulation modeling for it cost analysis. Technical report, Distributed Computing and Network Department, St.Petersburg State Technical University, St.Petersburg, Russia, 2006.

[12] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2000.

[13] A. Trendowicz, J. Heidrich, J. Münch, Y. Ishigai, K. Yokoyama, and N. Kikuchi. Development of a hybrid cost estimation model in an iterative manner. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 331–340, New York, NY, USA, 2006. ACM.

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

[15] J. van Gurp. *On The Design & Preservation of Software Systems*. PhD thesis, Rijksuniversiteit Groningen, February 2003.