Efficient Design of SCADA Systems Using Minimum Spanning Trees and the NFR Framework

Haranath Yakkali
Dept. of Computer Science
University of Texas at Tyler
Tyler, TX 75799
hyakkali@patriots.uttyler.edu

Nary Subramanian
Dept. of Computer Science
University of Texas at Tyler
Tyler, TX 75799
nsubramanian@uttyler.edu

Abstract

Supervisory Control and Data Acquisition (SCADA) systems are being increasingly used to monitor and control critical infrastructures ranging from computer networks to manufacturing, and proper design of SCADA systems is an important issue for developers of SCADA systems and their users. Current practice seems to center around purchasing a best-of-breed solution and configuring it to meet the expected system goals. In this paper we propose an efficient design approach based on minimum spanning trees and the NFR Framework wherein SCADA designs are captured in spanning trees and minimum spanning trees indicate the optimally weighted designs, and the most suitable among these weighted designs is chosen by employing the NFR Framework, which is a goal-oriented framework for analyzing competing alternatives and choosing the best one based on the goals for the SCADA system. The design algorithm is verified by applying it to a case study.

Keywords: SCADA, design, architecture, efficient, spanning tree, NFR Framework

1. Introduction

SCADA stands for Supervisory Control and Data Acquisition. SCADA systems monitor and control from remote – for example, the state of all facilities in an airport can be monitored and controlled from a central control room [1, 2]. The software/hardware in the central control room are referred to as the Master of the SCADA system, while the remotely monitored and controlled components such as door sensors, door actuators, door alarms, water monitors/controllers, HVAC monitors/controllers, etc. are all referred to as the remote terminal units (RTU's). Typically an RTU is a superset of each sensor/controller/actuator in the sense that an RTU can monitor and control a set of sensors/controllers/actuators – therefore, the air-conditioning plant at the airport may be connected to the Master by one RTU, while an airport passenger terminal may be connected to the Master by another RTU. The Master is connected to the RTU's by means of high speed datacomm links such as twisted cables, coaxial cables, optical fibers, or wirelessly. SCADA gives the business owner a centralized viewpoint of the status of the complete system so that any vulnerabilities or bottlenecks are detected much before they become an issue. SCADA may control facilities in a small geographical area or may be dispersed over a state, country, or internationally; for example, SCADA that monitors a water-treatment plant may cover only a small area, while the SCADA that monitors the water pipelines may be distributed over a much larger geographical area. In the case where SCADA covers huge geographical areas, the datacomm links connecting the Master and the RTU's may include cellular, satellite, and dedicated circuit technologies.

In several systems which employ SCADA, the control system architecture need not necessarily be the same as the system under control: for example, nervous system of the human body compared to the circulatory system; road network versus the traffic signaling system; and air network in relationship to the air traffic control. However, there are systems wherein the physical architecture of the control system and the system being controlled could be alike: for example, the Internet routing system and the Internet, or home electrical wiring and appliances controlled using X10 protocol [7]. But in the case of oil pipeline systems, the SCADA architecture is typically different both logically and physically from the oil pipeline layout. The most important concerns in deploying SCADA for oil pipelines seem to be reliability, integrity, security, and performance. While typically a best-of-breed solutions could be used for procuring SCADA systems for a given set of requirements, it would be useful for both SCADA system designers and users if a systematic method could be employed for selecting the appropriate control network design and thereby the most satisfactory SCADA system. In this paper we propose a technique for control network design that uses the concept of spanning trees – more specifically, minimum spanning trees [5], to develop potential design candidates that are then evaluated for suitability for system goals using the NFR Framework [6]. Since most of the systems being controlled (electrical wiring, network routing, and others in practice) themselves form a spanning tree or sparse graphs [4], the number of different spanning trees for control networks are usually limited – therefore, minimum spanning trees give a good approximation to the SCADA architecture and the NFR Framework provides an ideal platform for tradeoff analysis and suitability evaluation. The algorithm proposed in this paper is evaluated by applying it to a case study.

This paper is organized as follows: section 2 briefly discusses minimum spanning trees, section 3 discusses the NFR Framework, section 4 presents the algorithm for designing SCADA systems employing minimum spanning trees and the NFR Framework, section 5 applies the algorithm to a case study, and section 6 concludes the paper.

2. Minimum Spanning Trees

Consider the physical pipeline layout shown on top of Figure 1. This pipeline (could be water, natural gas, oil, or any other fluid) has a pump for increasing the fluid pressure, a valve controller for directing the fluid along the correct outlet, and three pressure sensors for monitoring the pressure at the outlet. One possible control architecture for this system is shown in the middle of Figure 1 (architecture A) wherein the control links follow the actual system – this could mean that the medium of the pipe is used for transmitting electrical signals or the electrical cables follow the path of the pipe (for example, telephone signals that parallel rail lines or X10 signaling that uses the wires for transmitting control signals as well). Another possible control architecture for this system is shown in the bottom of Figure 1 (architecture B) wherein all the control links connect directly to the control center. Similarly, other control architectures are possible that connect the controlled entities (RTU's) with their control center.

However, an important point to note is that all control architectures form a spanning tree – more importantly, a minimum spanning tree, where the minimization is performed according to a dominant criteria. Given a connected, undirected graph $G = (V,E)$ where V is the set of all vertices (RTU's) in the network, and E the set of all possible interconnections between pairs of nodes, then the acyclic subset $T \subseteq E$ that connects all of the vertices is called the spanning tree [5]. If for each edge $(u, v) \in E$, we assign a weight $\omega(u, v)$ specifying the cost (in terms of units of length or money or any other) to connect u and v, then the acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight $\omega(T) = \Sigma_{(u, v) \in T} \omega(u, v)$, is minimized is referred to as the minimum spanning tree. For example, if cost of links (connecting components of the control network) were the most important criterion, the minimum spanning tree could be different from that obtained if the length of the links were the most important criterion. For example, in Figure 1, we could say that control architecture A, which is a spanning tree, is minimized from the perspective of cost of securing the installation while control architecture B, another spanning tree, is minimized from the perspective of the length of the links. For a given graph there can be different minimum spanning trees based on the minimization criterion and therefore it becomes important to choose from competing spanning trees or go for a hybrid of two or more spanning trees. This choice is guided by technical factors such as RTU technologies, communication technologies, data gathering techniques, telemetry and controls, and protocols [2]. For this purpose the NFR Framework provides an excellent tool to choose among alternatives and this Framework is discussed next.

3. The NFR Framework

The NFR Framework [6] is briefly described in this section. The NFR Framework, where NFR stands for Non-Functional Requirements, requires the following interleaving tasks, which are iterative:

1. Develop the NFR softgoals and their decomposition.
2. Develop operationalizing softgoals and their decomposition.
3. Determine contributions between operationalizing softgoals and NFR softgoals.
4. Develop goal criticalities.
5. Evaluation and analysis.

The graph that results from the application of above steps is called the Softgoal Interdependency Graph (SIG). The NFR Framework uses the concept of goal satisficing. The notion of goal satisficing of the NFR Framework assumes that decisions taken during the development process usually contribute only partially (or against) a particular goal, rarely "accomplishing" or "satisfying" goals in a clear-cut sense. Consequently any model is expected to satisfy NFRs within acceptable limits, rather than absolutely. There are different degrees of satisficing and the degrees

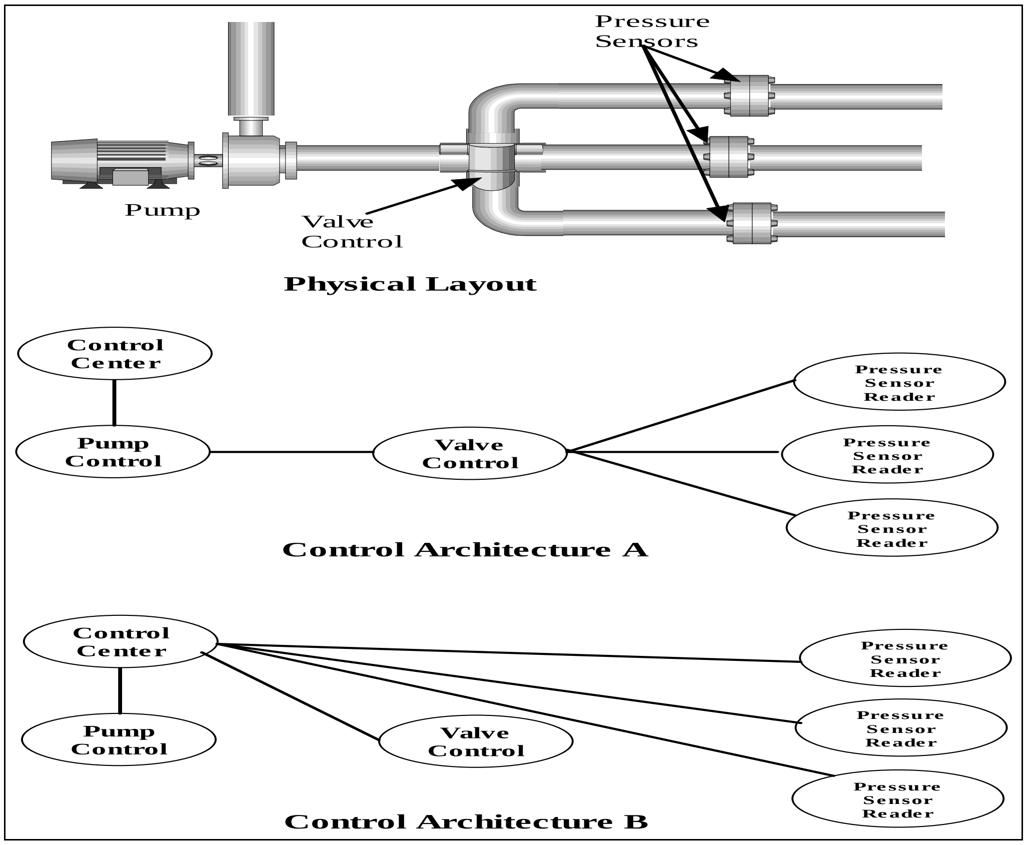

Figure 1. Control Architectures for a SCADA System

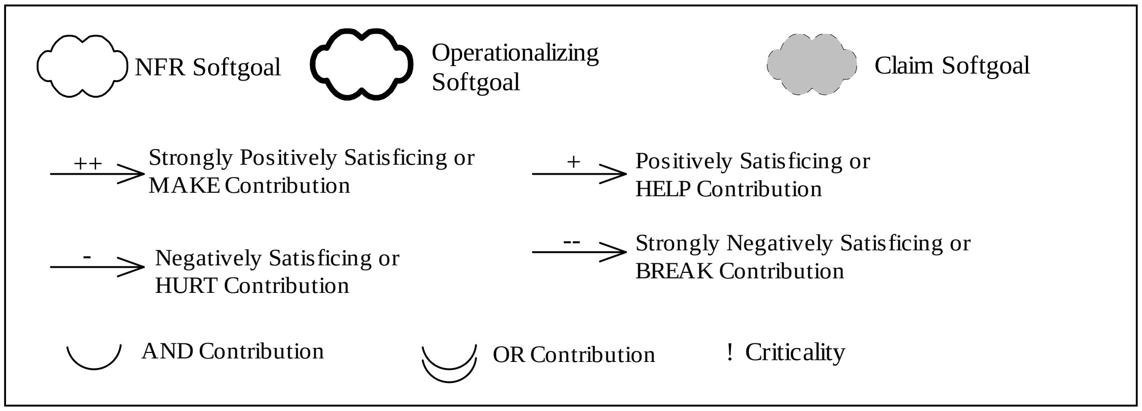
NFR Softgoal
Operationalizing Softgoal
Claim Softgoal
++ Strongly Positively Satisficing or MAKE Contribution
+ Positively Satisficing or HELP Contribution
- Negatively Satisficing or HURT Contribution
-- Strongly Negatively Satisficing or BREAK Contribution
AND Contribution
OR Contribution
! Criticality

Figure 2. Partial Ontology of the NFR Framework

are indicated by arrows annotated with + or – symbols. In the NFR Framework, each requirement (either system requirement or software requirement) is called an NFR softgoal (depicted by a cloud), while each potential configuration of a SCADA system is called an operationalizing softgoal (depicted by a dark cloud). The rationale for various decisions is captured by yet another softgoal – the claim softgoal (depicted by a cloud with dashed border). The partial ontology of the NFR Framework is given in Figure 2 and the steps to use the NFR Framework are described below.

During the NFR goal decomposition process, the NFR softgoals are decomposed into their constituent softgoals based on the domain – in this paper we treat the requirements for the SCADA system as NFR softgoals. This decomposition is not unique and depends on what the people performing the decomposition consider important for the domain at that point in time. The softgoals are named using the following convention:

Type[Topic1, Topic2, …]

where *Type* is an NFR and *Topic* is the system or domain to which the *Type* applies. The NFR softgoals can be related to each other by three contributions: AND-contribution (indicated by single arc), OR-decomposition (indicated by double arc), and a refinement (only one child NFR softgoal). The various architectures for a SCADA system and their elements are considered as operationalizing softgoals in the NFR Framework. Some NFR softgoals may be declared critical or priority softgoals by annotating them with '!' marks and this is done during step 3 of the NFR Framework application process. During NFR softgoal satisficing determination we determine the contributions made by the operationalizing softgoals to the various NFR softgoals – these contributions can be of four types: MAKE, HELP, HURT, and BREAK. These contributions have the following ranking:

MAKE > HELP > HURT > BREAK.

During the final step – the evaluation and analysis step – we apply the propagation rules of the NFR Framework to determine to what extent the models satisfice the NFR softgoals. While detailed propagation rules may be seen in [6], in this paper we will be using the following simplified propagation rules:

R1. If most of the contributions received by a leaf NFR softgoal are positive (MAKE or HELP) then that leaf NFR softgoal is considered satisficed.
R2. If most of the contributions received by a leaf NFR softgoal are negative (BREAK or HURT) then that leaf NFR softgoal is considered denied or not satisficed.
R3. In the case of priority softgoals, or when there is a tie between positive and negative contributions, the system architect or the developer can take the decision based on or a variation of R1 and R2
R4. In the case of AND-contribution, if all the child softgoals are satisficed then the parent NFR softgoal is satisficed; else the parent softgoal is denied.
R5. In the case of OR-contribution, if at least one child softgoal is satisficed then the parent NFR softgoal is satisficed; else the parent softgoal is denied.
R6. In the case of refinement (only one child) the parent is satisficed if the child is satisficed; and the parent is denied if the child is denied.

Upon applying these propagation rules, if the root NFR softgoals are satisficed then the goals of that SCADA system have been met to a large extent. Throughout the SIG development, the rationales for the various contributions are captured by claim softgoals.

4. The Algorithm

The algorithm for optimal design of SCADA systems is given below:

Step 1. Draw the graph G = (V, E) of the required SCADA system – this graph will have the control centers and the RTU's as nodes and the connections between the nodes as edges.
Step 2. Associate weights for edges in the graph of Step 1 – the weights may relate to real cost, perceived cost, length, or another measure; here perceived cost relates to cost of intangibles: for example, reliability of links may be related to cost of products used or to the strength of the signal used or to the amount of energy used, each of which may be interpreted in terms of a dollar value.
Step 3. For each weighted graph G, determine the minimum spanning tree.
Step 4. Determine the suitability or otherwise of each minimum spanning tree for the goals of the SCADA system using the NFR Framework.
Step 5. Implement the most appropriate minimum spanning tree determined from the NFR Framework as the optimal SCADA design implementation; if more than one minimum spanning tree is determined to be appropriate then a hybrid design may be required.

In Step 1 we draw the graph of the expected SCADA system – here the RTU's on the physical layout are treated as nodes and any control centers to which the RTU's need to communicate are added to the graph as new nodes. Then add edges between nodes using wired and wireless technologies – that is one graph has wired edges and the other has wireless edges (both of these follow the physical topology), the third has wired connections from each RTU direct to a control center, and the fourth has wireless connection from each RTU direct to that control center. All the four types of graphs are weighted according to length and cost and minimum spanning trees are determined for each graph. Each of the minimum spanning trees is evaluated for suitability using the NFR Framework and the most suitable set of minimum spanning trees are considered for implementation; if the set has more than one element then a hybrid design may be the most optimal.

5. Case Study

Our algorithm was applied to the SCADA requirements shown in Figure 1. For Step 1, we created the graph as shown in the left at the bottom of Figure 3. For Step 2, this graph was weighted with the cost of the links, first for wired and then for wireless connections. For Step 3, we developed a Java language

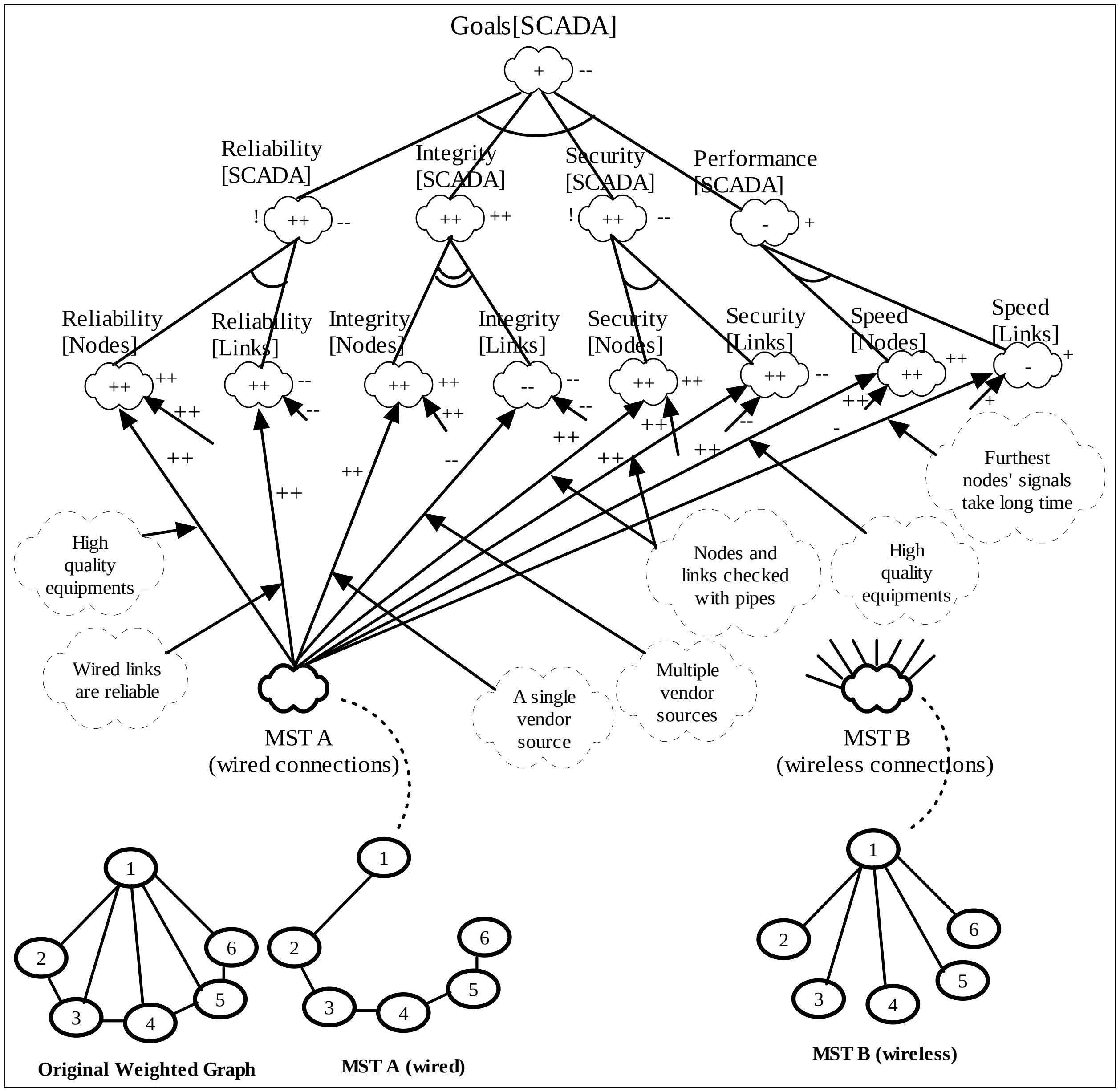

Figure 3. MST's and SIG for Optimality Analysis

program that used the Prim's method [5] to determine minimum spanning trees (MST's) for the graph and the two MST's obtained are shown at the bottom of Figure 3 as MST A (for wired connections) and MST B (for wireless connections). For Step 4, we developed the

Softgoal Interdependency Graph (SIG) shown at the top of Figure 3. In this SIG we decomposed the goals for the SCADA system (represented by the NFR softgoal Goals[SCADA]) into four child NFR softgoals: Reliability[SCADA], Integrity[SCADA], Security[SCADA] and Performance[SCADA]; this is an AND-decomposition as indicated by a single arc which means all these goals need to be satisfied. We also indicated two of these softgoals (Reliability and Security as high priority by '!' symbols). Each of these NFR softgoals was decomposed into corresponding child softgoals for nodes and links (as indicated by the topics in the softgoal names). The contributions made the two MST's were then determined – in Figure 3, justifications (claim softgoals) for the contributions made by MST A are only shown. Then the propagation rules of the NFR Framework were applied and the values propagated up the SIG are shown within the NFR softgoals as ++ for MAKE, + for HELP, - for HURT, and - - for BREAK. The root NFR softgoal (Goals[SCADA]) is HELP-satisficed by MST A (indicated by the + within the NFR softgoal) and BREAK-satisficed by MST B (indicated by the - - next to the NFR softgoal), which means that the design corresponding to MST A satisfies the goals of the SCADA system while that corresponding to MST B does not satisfy the goals. In Step 5 we implement MST A since it is the optimal design.

6. Conclusions

Current approach to designing SCADA systems typically focus around purchasing best-of-breed solutions – however, it will be useful to both developers of SCADA systems and their users if a systematic approach to designing such systems were available. In this paper we propose an efficient technique for designing such SCADA systems based on minimum spanning trees and the NFR Framework. Typical SCADA solutions conform to a spanning tree structure [5] simply because the systems monitored and controlled are typically sparse networks such as electrical wiring, oil pipelines, railway lines and the like; therefore, minimum spanning trees that form when spanning trees are minimized based on a criterion such as cost or length are natural candidates for SCADA control architectures. However, how do we ensure optimal solutions? For this we use the NFR Framework [6], a qualitative framework that allows selection among competing designs using a goal-oriented approach to evaluate amongst tradeoffs. We verified our algorithm by applying to a case study.

There are several directions for further research. Most importantly we need to define the parameters for creating the graph in Step 1 of the algorithm and for weighting the graph in Step 2 so that too many possibilities are not created for the MST's. We need to apply this technique to more practical situations and understand its pros and cons better. Finally, we need to consider hierarchical approaches when the number of nodes is a large number. However, we believe our approach provides an efficient means for designing SCADA systems.

References

1. N. Subramanian, "Improving Security of Oil Pipeline SCADA Systems Using Service-Oriented Architectures", *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, Vol. 5333, pp. 344-353, November, 2008.
2. G. Bezerédi, "Crude Oil Pipeline SCADA system review", downloaded from http://www.pipeline-automation.com/pac-paper-bezeredi.pdf on December 5, 2009.
3. "Keystone Pipeline Project – Mainline Pipeline Route Alternatives", Document No. 10623-004, January 2007, downloaded from http://www.entrix.com/keystone/project/supplemental/alternatives.pdf on December 8, 2009.
4. A. Kershenbaum and R. V. Slyke, "Computing Minimum Spanning Tree Efficiently", *Proceedings of the ACM Annual Conference,* Vol. 1, pp. 518-527, 1972.
5. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1997.
6. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, Boston, 2000.
7. http://www.x10.com/support/basicx10.htm