Simulating Tyco’s iSTAR Door Security Monitoring Systems

Design Team
Faisal AlRashed, Christopher Coury, Grant Eastman, Sean Naegeli

Design Advisor
Professor Sagar Kamarthi

Abstract
Tyco’s Software House division has identified a need to simulate and mimic the behavior of their iSTAR and CCure product lines. Currently these products are tested by creating a mock building with a security monitoring system. This mock building is made up of physical products and is known as a “forest”. In collaboration with a team of students from Shanghai Jiao Tong University, the Northeastern iSTAR Capstone team was tasked with creating a proof of concept solution to mimic the activity of these “forests”. This proof of concept will allow Tyco to test future server products with this simulation solution, instead of continuing to invest in physical prototypes. The solution utilizes object oriented programming to create and communicate input and output messages to and from the host server.
The Need for Project

This project will replace the current Software House testing system with a solution that will eliminate the use of physical prototypes to test the reliability of Tyco’s server software. Software House, a subsidiary of Tyco, manufactures a family of products called iSTARs. These products serve as commercial security access control monitors for small and large security systems. Currently, tests are run on these controllers through physically setting up mock systems to replicate what a system will act like in a real building. Stacks of these systems fill up rooms within the Tyco Facility in Westford, MA. These rooms of systems have been named forests due to the look of these densely packed systems. Throughout the capstone project, the team has worked on replacing these “forests” with a simulation tool capable of testing and mimicking multiple controller systems for quality assurance purposes.

The Design Project Objectives and Requirements

The objective of the project is to create a simulation tool that allows the user to input a number of parameters. These parameters will determine the messages created and communicated to the host. The project will also eliminate the need for a physical setup of iSTAR controllers.

**Design Objectives**

The objective of this project is to create a simulation tool that enables a user to input multiple parameters through a Graphical User Interface (GUI). According to those parameters, messages will be created and communicated to a host to test the reliability of the host in handling messages. It will also eliminate the need for physically using iSTAR controllers to test the host. The new “simulator” will be a proof of concept and will only simulate two types of messages. The first indicates that the iSTAR is being tampered with. The second message will show that the iSTAR is cleared and active for use. These messages will be shown in the monitoring station on the SiteServer, the host that is used to configure and monitor iSTAR devices within the system.

**Design Requirements**

The simulation tool must allow the team to create a GUI that will enable Software House to specify arrival rates, number of iSTAR controllers to be simulated, and message type. The tool must offer a great deal of flexibility and allow for future expansion of this proof of concept. The software must support Object Oriented Programming (OOP) and allow for third party function customization.

**Design Concepts Considered**

Several design concepts were considered, mostly involving the
The use of simulation software that was a core competency of the group. It soon became apparent that the team needed to develop the software from the ground-up in C# in conjunction with SJTU.

A language the team was unfamiliar with, or using pre-existing simulation software and contouring it to fit the needs of this project. The team considered Arena, AnyLogic and Process Model for potential simulation software solutions. Although this would reduce the amount of lead time required for the team to complete a solution, the use of a simulation package was rejected due to lack of functionality.

For the coding language, three major language approaches were considered; C++, C# and a combination of JavaScript with XHTML and PHP. In the early documents from Tyco, the initial suggestion was to use C#, but a familiarity with C++ and suggestion from a faculty resource to use JavaScript justified the consideration of both of these options.

The two concepts that were seriously considered were Arena and C#. A major reason to consider these options was that both solutions could interact with a front end user interface. The other component is the ability to talk with a back end code that communicates with the server that would be developed by the team from SJTU. While Arena has the capability to communicate with other applications, it does not have the functionality to run live event handling.

Alternatively, C# has the capability to integrate user input, the GUI, in a “Windows” style while running live event handling. Using C# allows for the output of error messages and event simulation at user specified instances. This code would require the use of loops, classes and other C# architecture that the team did not have experience developing. However, the partner team from SJTU had coding experience with C# and was an asset while the NU team was learning the language.

**Recommended Design Concept**

C# will be used to complete the simulation design. This program will involve a GUI, Controller Generation, Event Handling and Communication with the host SiteServer.

Both the Northeastern and SJTU teams agreed that creating the simulation in the C# programming language was the best fit for this needs of this project. A GUI interfacing with the C# program would generate a user-specified number of iSTAR controllers, each with a unique IP and MAC address. This is the group of iSTARs that would be used to send activity input messages to the host server at different rates. The rate of activity message transactions would also be defined by the user input. These messages will be communicated with the host
SiteServer to mimic the existence of real iSTAR units. The same C# program that generates the controllers will be used to interact with the host client.

**Design Description**

The design of this project is divided into two phases of development. Responsibilities between the SJTU and Northeastern groups to develop the Simulation were divided into two large sections based on the functions of the iStar control units and the CCure site server software. The final application can be viewed as a two part application. One part acts as a means of simulating and mimicking the behavior of the iStar units and card readers. The other part takes information from the iStar units and communicates it to the site server software.

Phase I of the Northeastern responsibilities involves seven milestones: Problem Formulation, Develop Pseudo-Code, Simulation Model Development and Verifying and Validating the Model. This would require the most work of either phase in this project. Phase II of the project will focus on integrating the simulation application developed by the Northeastern team with the other parts of the final solution. These include the SJTU application to communicate with the site server and a GUI for users to interact with the application. This phase will begin when the Northeastern team begins the development of the GUI. Verification and validation of the final result would conclude this phase prior to implementation.

**Key Advantages of Recommended Concepts**

The chosen design for development of the simulation, and the workload to complete it, adhered to each team’s strengths. The advantage of this concept was that SJTU would focus on the majority C# programming. Meanwhile, the Northeastern team could focus on development of data to feed into the Remote Server as well as creation of the GUI. The core competency for the Northeastern Team has been a combination of project management and professional communication, in addition to mining for statistical data trends. The experience that SJTU had with C# was advantageous when undertaking some of the more in depth coding necessary for communication with the server.

C# has high flexibility and additionally offers numerous libraries to
add functionality, such as implementation of distributions. Due to this level of language versatility, C# can be used to build more complex future phases of this proof of concept. Finally, C# is a standard language used throughout industry which will allow for Tyco, or future project teams, to easily expand this solution.

Financial Issues

No large capital investments will be made as a result of this project.

Although Tyco will have the chance to further develop this solution, labor will be the only subsequent investment.

The software solution created in this project will require no immediate capital costs for Tyco. The implemented solution used C# scripted files for the back end solution and front end GUI. The use and implementation of these files will come at no cost to Tyco. The company already has the software necessary to view, run and edit these programs. The only costs potentially associated with implementing this project stem from additional labor Tyco will have to put into this project to iterate the solution for additional iSTAR functionality. This cost may be avoided if Tyco decides to involve other student project teams to continue on the next phase of this project.

Each iSTAR unit produced by Tyco is sold for between $5,000 and $10,000. Forests are built at an enterprise level with over 2,500 of these controllers at a minimal cost of $18,750,000 to test for bottlenecks and glitches in the CCure server environment. This project will be used by Tyco to eventually replicate these test environments in a short amount of time. Preforming these tests on new CCure servers translates to millions of dollars in cost savings by avoiding sunken costs for the installation of a commercial system and a recognition of potential field errors that was previously impossible.

Recommended Improvements

It is recommended that Tyco use this project as a model for future iterations of iSTAR system functionalities.

This project was a proof of concept of one type of message passed between an iSTAR unit and a CCure Server. Tyco can use this as a model for other message types. The current message format is scalable to include these messages. It is recommended that future phases of this project follow a similar format for creating and sending instances of messages.