Submitted for publication.

Architecture-based Simulation for Security and Performance

Bradley Schmerl, Shawn Butler, and David Garlan
School of Computer Science, Carnegie Mellon University, Pittsburgh, USA
{bradley.schmerl,shawn.butler,david.garlan}@cs.cmu.edu

Abstract

Architectural analysis is key to producing high quality architectures. In this demonstration we present two extensions to AcmeStudio, and domain-neutral Architecture Development Environment, to add Performance and Security Simulation. Using AcmeStudio as the integration platform for these analyses allows comparisons and trade-offs between these different quality attributes.

1. Introduction

Developing a software architectural model now recognized as a crucial step in producing high quality software. In addition to allowing designers to understand a design in terms of its high-level abstractions such as computational components and their interactions, an architectural model is often suitable for analyses that can prevent errors from propagating to later phases of development. Such analyses include performance analysis [5], simulation [3], and protocol analysis [1].

One of the major difficulties in providing tool support for architectural design and analysis is the need to tailor those capabilities to the application domain. While some analyses may be generally applicable across many domains, typically the more significant forms of analysis take advantage of the particular kind of system built within an organization. For example, representational and analytic needs of a web services domain, which may be concerned with performance and throughput, will be quite different than those for an embedded controller domain, which may be concerned with schedulability and resource allocation.

One possible solution is to create many specialized environments – one for each domain. Indeed, during the first decade of interest in architecture description languages and tools we saw the introduction of dozens of notations and analytical methods, each specialized for some particular family of systems. For example, C2 [6] was restricted to layered event-based systems, and MetaH [7] focused on architectures for embedded avionics control systems.

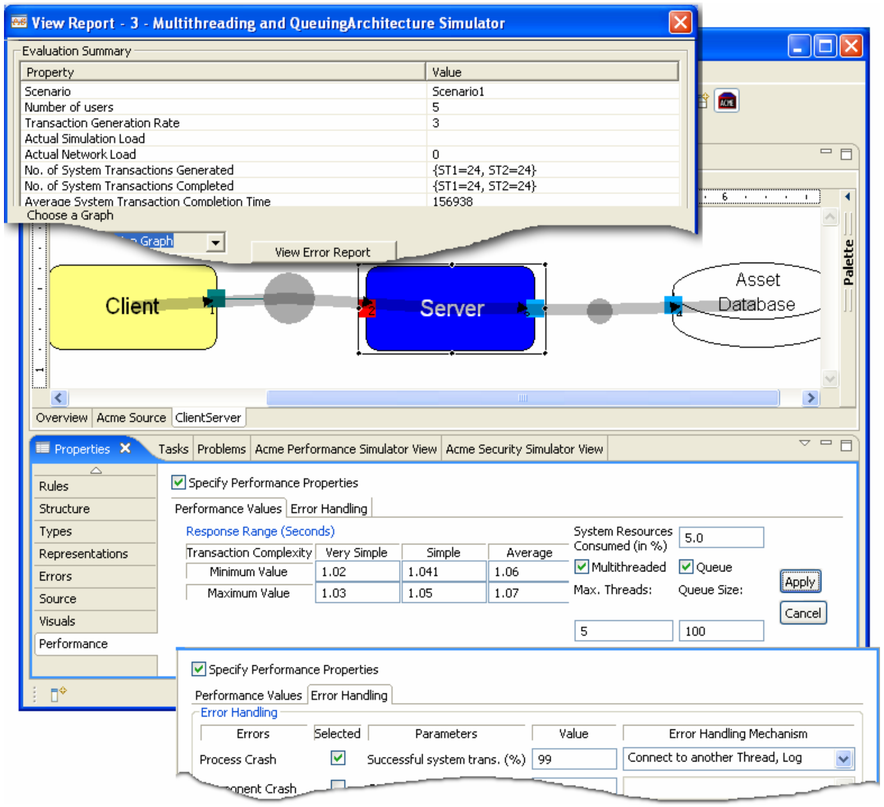
Unfortunately, constructing a new tool from scratch for each domain and form of analysis incurs a prohibitive cost. On the other hand, it is not desirable to require architects to use tools that are not suitable to their domains. Furthermore, integrating various analytic tools to take advantage of their respective benefits is also extremely difficult.

In previous work, we have developed a domain-neutral architecture development environment, called AcmeStudio, that can be tailored to specific domains at relatively low cost. We demonstrated this tool at ICSE 2004 [4], and showed how it could be tailored to the domains of automotive design and space systems engineering. The type of analysis that was provided for these domains used the built-in first-order predicate language of Acme to analyze the general topologies of those architectures.

While the kinds of analysis built in to AcmeStudio are useful, more sophisticated analysis is needed to analyze such quality attributes as security and performance. In this research demonstration we present a tool for analyzing the security and performance of architectural models and for making trade-offs across these two dimensions. The tool is an extension of AcmeStudio, leveraging its existing features for defining architectural models, providing specific architectural styles to specify the properties and topology relevant to the kinds of analysis, and using AcmeStudio's plug-in framework to provide security and performance analysis using Monte Carlo simulation to evaluate these properties under certain assumptions about their stochastic behavior.

2. Performance Simulation

Performance simulation allows an architect to conduct simulations based on performance properties of individual components and connections in the architecture, and the paths of interaction between them. In addition to performance properties, such as the average process-


View Report - 3 - Multithreading and QueuingArchitecture Simulator

Evaluation Summary

Property	Value
Scenario	Scenario1
Number of users	5
Transaction Generation Rate	3
Actual Simulation Load	
Actual Network Load	0
No. of System Transactions Generated	{ST1=24, ST2=24}
No. of System Transactions Completed	{ST1=24, ST2=24}
Average System Transaction Completion Time	156938

Choose a Graph

Specify Performance Properties

Performance Values | Error Handling

Response Range (Seconds)

Transaction Complexity	Very Simple	Simple	Average
Minimum Value	1.02	1.041	1.06
Maximum Value	1.03	1.05	1.07

System Resources Consumed (in %): 5.0

[✓] Multithreaded [✓] Queue

Max. Threads: 5 Queue Size: 100

Specify Performance Properties

Performance Values | Error Handling

Error Handling

Errors	Selected	Parameters	Value	Error Handling Mechanism
Process Crash	✓	Successful system trans. (%)	99	Connect to another Thread, Log
…onent Crash				

Figure 1. Performance property specifications and results for a simple architecture.

ing time for components and the average transmission rate on connectors, the architect may also specify:

a. Whether a component is *multithreaded*, and how many threads it is allowed to produce. This allows simultaneous processing of requests.

b. Whether a component has *queuing* enabled, and the size of the queue. For requests that cannot be processed because the component it busy, it is possible to specify how many requests may be queued before requests are discarded.

The above properties are standard properties required to conduct performance analysis; we used similar properties in a queuing-theory based approach to performance analysis in [5]. Furthermore, the architect is able to specify the probability of component error and how to recover from these errors. For example, an architect may specify the probability that a component or thread will crash, and whether to connect to another component/thread, report an error and stop, or report an error and continue.

Once component and connector properties are specified, the architect may specify paths through the architecture that can be taken to exercise the components and connectors. Out of these, scenarios are constructed that specify the simulation time, how many users are connected, the rate at which these users generate requests that will follow the paths in the architecture, the probability that particular paths will be taken, and the network load. The Performance Simulator then constructs a thread (or multiple threads) for each component and user, and then performs a Monte Carlo style simulation, using the probabilities of errors, calculating random service times based on the architect's inputs. The final result is a report indicating how many requests were generated, and how many of those requests were processed or failed.

Figure 1 illustrates how the Performance Simulator is implemented on top of AcmeStudio. To specify the performance properties required, the system must satisfy the Acme PerformanceSimulation style, and each component must satisfy the PerformanceComp component type; this type is used in two ways in AcmeStudio: (1) the type specifies the properties that need to be defined for performance simulation to work; and (2) the particular types trigger AcmeStudio to look for specific user interface embellishments that can be used to enter properties for that type. (In Figure 1, the Properties view has been extended by the Performance Simulator to add a Performance tab, allowing users to enter in response rates, threading and queuing characteristics, and error handling properties for the Server.)

In addition to integration into the AcmeStudio UI, the Performance Simulator adds its own view to AcmeStudio for defining the paths through the architecture that should be taken by the simulation (a path is shown as the thick grey line of Figure 1), and actions for running the simulation. An example report showing the result of the simulation of a scenario with 5 users and a generation rate of 3 requests per second is shown at the top of Figure 1. It indicates that all transactions on the shown path were able to be completed; the architect has specified satisfactory requirements for the performance of each of the components in the path.

3. Security Simulation

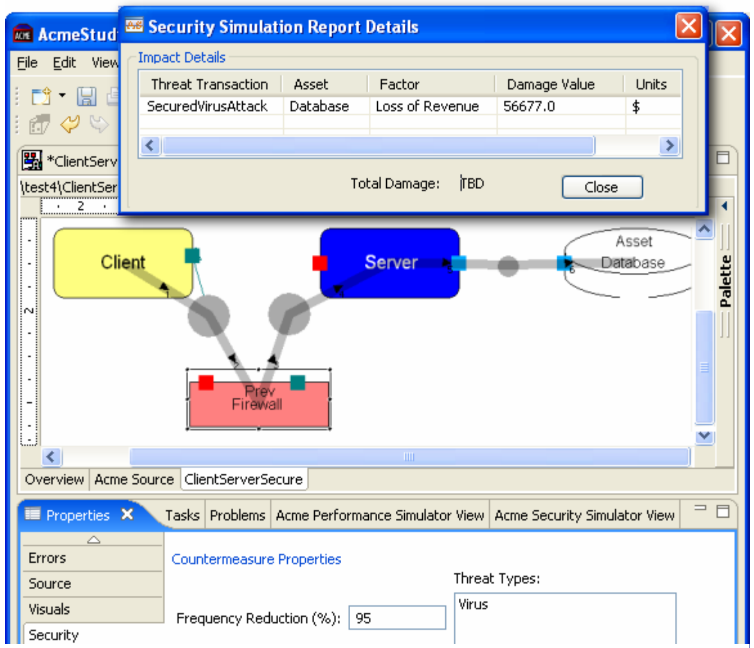
The objective of the Security Simulator is to enable an architect to perform simulations based on threat scenarios that affect the architecture. The main concepts in the security analysis are threat types, assets, and countermeasures; the simulation is based on the approach outlined in [2].

Threat types specify the possible threats that can affect the system, e.g., a virus , denial of service. Because different systems may be subject to different types of threats, the architect must specify each of the threat types that may be posed to the system.

Assets are components that may be damaged by particular threats. Assets are assigned a monetary value, and the particular threat types that may affect the asset are specified. For example, a database component may not be susceptible to password sniffing attacks, but may be vulnerable to data corruption as the result of a virus.

Three different types of *countermeasures* can be defined: *Preventative components* affect the frequency at which threats occur; *Monitoring components* and *recovery components* reduce the effect of a threat. The architect specifies each of the countermeasure's target threat types, and the effectiveness or reduction that the countermeasure has on the target threat.

Once the particular properties are specified, the architect must then define paths (consisting of components and connectors) through the architecture that particular threats may take. Such a path is called a *Threat Transaction* in the Security Simulator. The threat type that affects that path and the frequency (as a stochastic function) of the threat type are specified.



Threat Transaction	Asset	Factor	Damage Value	Units
SecuredVirusAttack	Database	Loss of Revenue	56677.0	$

Total Damage: TBD

Frequency Reduction (%): 95

Threat Types: Virus

Figure 2. A simple client-server architecture showing the cost of a virus attack on a database protected by a firewall that is 95% effective against it.

After the threat transaction is created, the assets in the path can be given outcome values. The outcome can be in terms of dollars, loss of life, loss of productivity, etc. A weight is assigned to each outcome factor.

Threat scenarios are composed of one or more transactions. A scenario is used as the main entry-point for the simulation, and specifies the amount of time that will be used in performing the simulation. The simulation takes into account the threat entering the transaction path, the frequency of the threat type and the countermeasures in the path. Monte Carlo simulation is performed to determine the most probable damage value to each of the assets in the threat transaction. The value obtained is multiplied by the frequency of threat transaction and the simulation time. This gives the total damage for the particular threat outcome factor. The end result of the simulation is a report that details the threat scenario, threat transaction and total damage to the assets in the threat transaction path.

In a manner similar to the Performance Simulator, the Security Simulator adds UI components to specify the security properties, keyed on the type of the particular component (Asset, Preventative, etc.), and the ability to define paths and scenarios for the simulation.

Consider the simple architecture illustrated in Figure 1, where we additionally define the database as an asset (giving the asset value of $100K), run a security simulation on the same path for a simulated virus attack, and define the scenario so that (1) the simulation time is two virtual months; and (2) a virus attack happens on average 5 times per day, with a maximum of 20 attacks per day. The report generated indicates a $1,112,409 loss of revenue for this scenario.

Figure 2 shows the Security Simulator where we have added a firewall between the client and the server, and changed the path to run through the firewall. If the firewall is 95% effective against virus threats then running the same scenario indicates that the damage has been reduced to $56, 677.

Such a simulation allows the architect to evaluate different scenarios, and to evaluate the effectiveness of different countermeasures against different attacks.

4. Quality Trade-off

Generally, design decisions affecting one quality attribute will interact with decisions affecting other attributes. For example, increasing the security of a design may decrease the performance (e.g., a firewall will slow the performance of the system). Therefore, the simulation results enable architects to gain insights into performance and security tradeoffs in their architectures, and evaluate the merits of including different countermeasures.

Table 1. Comparing tradeoffs between 3 different firewalls.

Design d			Simulation Results x	
FW	FW Perform (sec)	Viruses (% stopped)	User Event Latency (sec)	Exposure ($K)
A	.3	98	1.2	50
B	.22	90	0.9	63
C	.17	87	0.4	75

The architect can use Acme Simulator to compare the performance and security risk exposure values for various architectures. For example, suppose the architect needs to choose among different firewalls, each with different effectiveness and performance characteristics. The simulations for performance and security of each design can be run, and a table like the one in Table 1 can be constructed. From the table, the design with Firewall (FW) A appears to have the slowest performance, but the least amount of financial exposure. In contrast, the design that includes Firewall C has the fastest performance, but greatest exposure. Currently, comparisons need to be constructed manually; in future work, the tool will support this comparison directly.

Although this is a particularly simple example, with more complex designs understanding the trade-offs between performance and countermeasures to a variety of threats is highly non-trivial.

5. Implementation

Acme Simulator is written in the AcmeStudio framework. AcmeStudio is written on the Eclipse framework, which allows flexible extensions. Therefore, AcmeStudio acts as an integration framework for adding additional domain-specific architectural analyses. To support additional analysis, three things must be provided to AcmeStudio:

a. An architectural style on which to base the analysis. In the case of the security and performance analyses, the SecuritySimulation style specifies that that an Asset should provide a value property and a set of threat types to which it is vulnerable.
b. The code that performs the simulation. In the case of the performance simulation, each component, and the associated performance properties, are mapped into a thread that models the performance of that particular property.
c. User interface embellishments that allow for tight user interface integration with AcmeStudio. For example, the Security tab in the Properties view at the bottom of Figure 2 is provided by the Security Simulator plugin.

6. Conclusions

We propose to demonstrate a tool illustrating the potential of integrated simulation-based tools, centered on an architectural design environment that permits an architect to make engineering trade-offs when architectural decisions affect multiple quality dimensions. The use of an architecture based tool provides benefit through (1) a consistent interface for creating and viewing the architectural model and the impact of architectural decisions on dimensions such as performance and security; (2) a flexible integration framework for Monte Carlo simulations as plug-in analyses that become available when certain styles are used; and (3) the ability to support trade-offs between different analyses, possibly written by different parties.

AcmeStudio is available for free from http://acmestudio.org. The Acme Simulator extensions are also available from this site.

Acknowledgements
This work was supported by a grant from CMU's Cylab. We also appreciate the work of various student teams in implementing the simulations described here.

References

[1] Allen, R., Garlan, D., and Ivers, J. “Formal Modeling an Analysis of the HLA Component Integration Standard.” Proc. 6th International Symposium on the Foundations of Software Engineering (FSE-6), 1998.

[2] Butler, S. Security Attribute Evaluation Method: A Cost-Benefit Approach. *Proc. ICSE 2002*, pp. 232-240.

[3] Luckham, D.C., Kenney, J.J., Augustin, L.M., Vera, J., Bryan, D., and Mann, W. “Specification an Analysis of System Architecture Using Rapide.” *IEEE Transactions on Software Engineering*, 21(4):336-355, 1995.

[4] Schmerl, B., and Garlan, D. “AcmeStudio: Supporting Style-Centered Architecture Development (Research Demonstration).” Proc. 26th ICSE Edinburgh, 2004.

[5] Spitznagel, B., and Garlan, D. “Architecture-Based Performance Analysis.” Proc. the 1998 Conference on Software Engineering and Knowledge Engineering (SEKE'98), 1998.

[6] Taylor, R.N., Medvidovic, N., Anderson, K.M., Whitehead, E.J., Robbins, J.E., Nies, K.A., Oriezy, P., and Dubrow, D.I.. “A Component- and Message-Based Architectural Style for GUI Software.” *IEEE Transactions on Software Engineering*, 22(6):390-406, 1996.

[7] Vestel, S. “MetaH Programmer's Manual, Version 1.09.” Technical Report, Honeywell Technology Center, 1996.

Appendix A: Demonstration

The demonstration that we plan to do at ICSE will be divided into three phases, that follows the outline described in the body of the paper.

Phase 1: Performance Simulation

During this phase we will demonstrate the performance simulation tool, starting with an architecture in an architectural style that does not have performance attributes defined. We will then:

1. *Show how to specify the performance attributes for the architecture.* This is done by assigning the PerformanceSimulation architectural style to the family, and then choosing the components and connects to be involved the performance simulation, by assigning the appropriate types. Assigning the types will make available the performance simulation UI embellishments, and we will show how to use these to specify simple response and transmission times for the component and connectors.
2. *Define the scenario for the simulation.* This is done by first defining paths through the system that the scenario will exercise. We will then define the scenario which specifies the paths to exercise, the number of users in the simulation, how frequently the generate transactions, and the total simulation time.
3. *Run the simulation and view the results.* Once the scenario is simulated, we will show the reporting results. In this simple case, many of the requests will be lost because the system will be overloaded. We will show how to tell this from the reports.
4. *Improve the performance attributes.* We will then show how adding multi-threading and queuing to the components will improve the performance of the system.

After this phase, it will be clear to the audience how to use the simulation to determine the performance properties of the architecture that will need to be satisfied by an implementation to ensure correct performance of the implemented system.

Phase 2: Security Simulation

During this phase, we will demonstrate the security simulation tool, continuing with the architecture that we completed in Phase 1. This phase will contain the following steps:

1. *Specify the values for the assets and the paths through the system.* The paths will be based on the paths defined in the performance simulator.
2. *Execute the security simulation.* On this system, because it does not have any countermeasures yet defined, the security simulation will show that a threat can cause extreme damage to the system.
3. *Make the system more secure.* By adding a countermeasure to the architecture in the path taken by the threat, we will show how to make the system more secure, and how this reduces the damage that a threat along the path will have. The architecture that will result will be similar to Figure 2.
4. *Show that security degrades performance.* We will now rerun the performance simulation, after assigning performance values to the countermeasure, and will show that now the performance of the system is impacted negatively by the addition of the security measures.

After this phase, it will be clear to the audience how to specify security attributes, how to run the security simulation, and how it is possible to now reason about the impact of security on performance, and vice versa.

Phase 3: Performance/Security Trade-offs

In this phase, we will introduce different versions of the architecture used in the previous phases that contain different security countermeasures to ward off various threats. We will run the security and performance simulations on these architectures to construct a table for comparing and choosing the most suitable version of the architecture that finds a balance between security and performance for the stakeholders of the architecture.