An Environment for Automated Performance Evaluation of J2EE and ASP.NET Thin-client Architectures

John Grundy[1, 2], Zhong Wei[1], Radu Nicolescu[1] and Yuhong Cai[1]
Department of Computer Science[1] and Department of Electrical and Computer Engineering[2], University of Auckland
Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract

Assessing the likely run-time performance of applications using thin-client architectures during their design is very difficult. We describe SoftArch/Thin, a thin-client test-bed generator that synthesises performance test-bed thin-client and server code from high-level software architecture models. This generated code is performance tested using a third-party tool and the results summarised. Architecture models can be evolved and tests repeated during application development to inform software engineers of realistic performance characteristics of their designs. Our environment currently supports J2EE and ASP.NET-based thin-client code generation and performance testing.

1. Introduction

Thin-client software architectures have become extremely common in enterprise system implementations. A user accesses the system via a web browser, which obtains content to render from web server pages. These web components access databases, application server objects, legacy systems and so on. Two common technologies for implementing such systems are Java 2 Enterprise Edition and Microsoft™ .NET.

Predicting the likely performance of thin-client systems during their design is very challenging, as it is with other distributed systems approaches [5, 10, 12]. Various tools and techniques have been developed to predict distributed system performance, notably via simulation, benchmarking and rapid prototyping [1, 9, 3, 15]. Various tools have been developed to assess the performance of implemented distributed and thin-client applications [7, 12, 17]. It remains difficult to assess the likely impact on performance characteristics of different web architectural design and implementation technology decisions. None of these current approaches provide architects with high-level support for modelling their architectures, easily exploring performance impacts of different design and implementation alternatives and obtaining realistic likely performance indicators.

We describe SoftArch/Thin, an environment for high-level modelling of thin-client application architectures with an associated performance test-bed generator. Architects sketch a high-level design of their intended system architecture, including major client, web server component, application server component and database abstractions. Runnable J2EE and ASP.NET code is generated fully automatically from this model along with configuration scripts for the Microsoft™ Application Centre Test, a third-party thin-client performance testing tool. These generated servers are performance tested and the results presented for analysis and architecture and technology evolution.

We firstly motivate this research and describe related approaches to thin-client architecture performance analysis. We give an overview of the SoftArch/Thin approach and illustrate some case study distributed system architecture models. We describe how performance test-bed code and testing tool scripts are generated and illustrate examples of performance testing different design models and implementation technology performance. We discuss results of evaluations of our prototype tool, summarise the contributions of our work and outline areas for future research.

2. Motivation

Thin-client software architectures follow more or less the structure outlined in Figure 1. Users access enterprise system functionality via web browsers, which render marked-up text, images and other media obtained remotely from web servers.

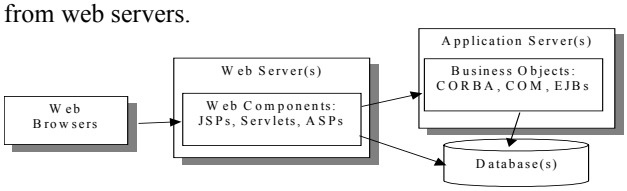

Figure 1. Thin-client software architectures.

A set of "web components" are hosted by these web services, typically grouped into multiple implementations of "pages" to display in the client browsers. Typical implementation technologies used to build these web components include Java Server Pages (JSPs), Java Servlets, Active Server Pages (ASPs), PHP scripts, CGI scripts, and Perl scripts. Web components may access databases directly (in relatively simple systems), producing a "three-tier" architecture. Alternately, they may access (possibly distributed) enterprise application server components like Enterprise Java Beans (EJBs), COM and CORBA remote objects, and legacy systems, producing a "multi-tier" architecture.

When designing a thin-client application developers will normally have some desired or required performance criteria from the system's non-functional requirements that different web components should meet. A variety of different architectural arrangements are possible: splitting or combining pages, using three-tier or multi-tier architectures, replicating web server and application server components, using multiple databases. They typically specify various performance-impacting characteristics e.g. number of server threads used, hardware and network configurations and so on. They may also realise the design with different web component, application server component and database server technologies. It is very hard to predict how well a thin-client architecture design will meet its performance requirements under this great range of different alternatives [1, 5].

A variety of approaches have been developed to estimate or assess performance of software applications. Simulation-based approaches build models of software application architectures and use these models to estimate application performance based on architecture [1, 14] or middleware [10, 15] of the target application. A variety of modelling and simulation approaches have been tried. As these approaches simulate application performance, their accuracy is prone to large variation and it is very difficult to obtain performance models for 3rd party support applications such as databases. Benchmarking approaches [3, 5] provide reference application architectures and implementations and compare relative performance when different technologies are used to implement the reference application. These provide accurate performance measures for the benchmark application, but are only a very rough performance guide for any related application. Rapid prototyping approaches [9] focus on rapid development of a partial application, usually focusing on implementing partial versions of performance-critical parts (e.g. networking and database loading). A large amount of development effort must typically be expended even for simple prototypes. If the architecture evolves the rapid prototyping must be repeated to get updated performance estimates for the final target application.

A number of tools have been developed to support software architecture performance analysis [7, 1, 14], application performance measurement [12, 8, 16, 18], and network performance analysis [17], as well as software architecture modelling and analysis in general [4, 11].In general these either do not support performance modelling of architectures or provide simulation-based approaches. Those supporting analysis of architecture performance via testing of real code either don't support thin-client abstractions or require a hand-implemented, completed thin-client application to test.

3. Our Approach

We have extended our earlier performance test-bed analysis tool work [7] to investigate support for thin-client architecture modelling and performance analysis. This has included developing code generators for both J2EE and C#, ASP.NET thin client application implementation technologies. The aim is to provide architects with a tool to quickly model their architectures and to gain accurate performance estimates of these models by generated and performance testing real web server component and related code. Such a tool could be used iteratively throughout development of the application, from initial simple architecture analysis to detailed performance testing of a complex architecture used for the actual system design.

Figure 3 provides an outline of how SoftArch/Thin is used by architects to assess the likely performance of their thin-client architecture designs. An architect first models a candidate software architecture for their web-based system (1).

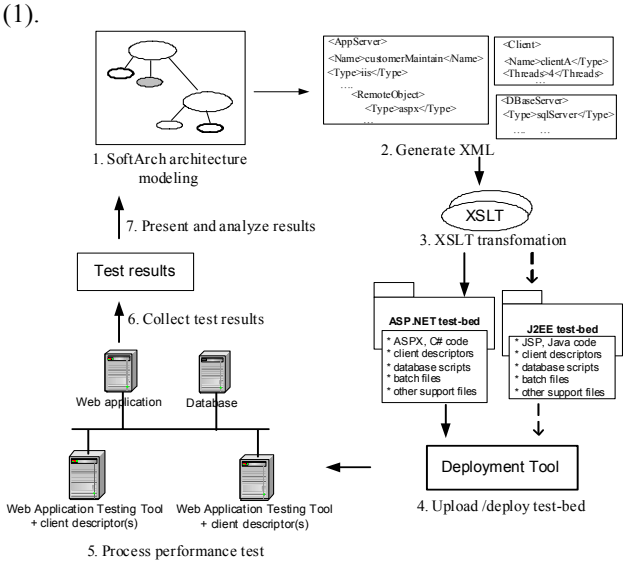

Figure 2. Outline of our approach.

Abstractions used include clients, client requests, servers, server components, server operations, databases, database tables, and various relationships and properties. From this architecture model an XML encoding of the design is generated by SoftArch/Thin (2). A set of XSLT transformation scripts is run over this XML to generate

various client and server test bed code and support files, including JSP and .java code, ASP and C# code, compilation, database configuration and component installation scripts, and thin-client testing tool configuration scripts (3). A deployment tool is used to upload this generated code and scripts to multiple networked client and server host machines (4). Performance tests are run which involves the client browsers being instructed to make large numbers of requests to the server components (5). Performance test results are collected (6) and analysis and summaries displayed (7). The architect may then refine their architecture and/or choose different implementation technology options in the SoftArch/Thin modelling tool, repeating the performance test generation and runs.

4. Modelling Thin-client Architectures

Our SoftArch/Thin performance test generation tool provides a graphical modelling environment for specifying abstractions making up a thin-client application architecture design. Each design element has a set of properties, some related to its structural architectural characteristics and others used by the code and script generation process to formulate performance test beds. The modelling tool is based on our earlier SoftArch architecture design environment work [6], with meta-model extensions for supporting thin-client and test bed code generation facilities.

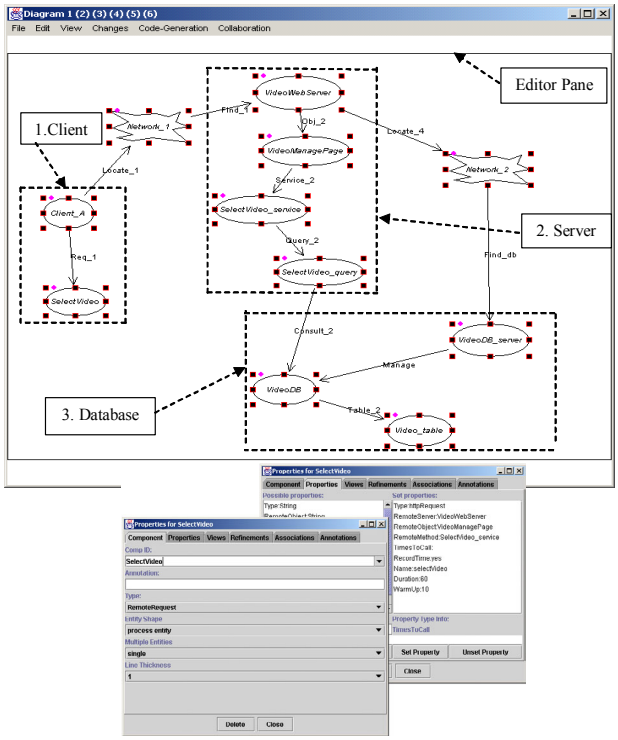

Figure 3. SoftArch/Thin model of a simple architecture.

A simple example of a software architecture model from SoftArch/Thin is shown in Figure 3. A single client, Client_A, has one request defined for it, selectVideo. A VideoWebServer has one web component, VideoManagePage, which in turn has a selectVideo_service. The selectVideo_service makes one database request as part of its operation. A database server, VideoDB_server, has a videoDB database, with one table defined, video_table. Each of these abstractions in the architecture design is related to others by a variety of relationships. These include ownership e.g, selectVideo belongs to Client_A; hosting e.g. videoDB_server Manage videoDB; and message passing e.g. selectVideo_query Consult2 videoDB. Other relationships include refinement, allowing complex architecture elements to be composed of hierarchical compositions.

Each architectural element and relationship has a range of properties, accessed by dialogues as shown in Figure 3 for selectVideo. Some are structural, such as Type and RemoteServer, specifying architectural element and relationship characteristics. Others are used by the test bed code generation, such as TimesToCall, RecordTime, Duration and Warmup. These are used to formulate appropriate testing parameters and control code.

Architecture models in SoftArch/Thin may be very abstract, capturing very few basic abstractions. Test beds generated from such a model can give a broad indication of performance of the modelled abstractions in the target application but will necessary be over-simplified. The architect can refine a high level architecture model to much more detailed levels, specifying many more client and client request mixes, more detailed web server pages and services, more detailed application server objects, and more database tables and table properties. Such a more detailed model will enable much more realistic performance measures to be determined.

Figure 4 shows such a more detailed architectural design, in this example part of the “PetStore” J2EE reference application [13]. In this example several clients are identified, each with different requests they will make to the server. Each client has different numbers of instances specified, and similarly each client request has different numbers of requests to run and wait times between each request. When test bed code is generated, these clients will be concurrently run. The architect can vary the number and mix of client requests to gauge their impact on application performance. Some of the web server pages do single operations e.g. ViewCategory_service, while others do multiple e.g. PlaceOrder_service. Multiple diagrams can be used to model subsets of the architecture to help manage complexity. Alternative models for the same architecture can be versioned within the tool, allowing multiple design decisions to be modelled and their relative performance compared.

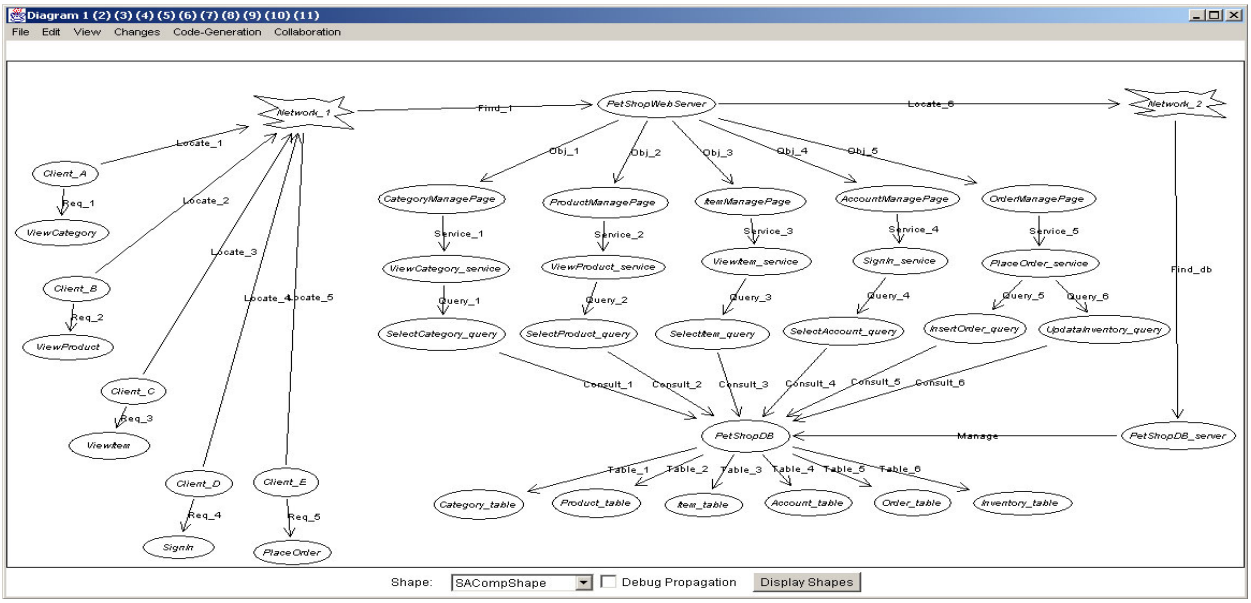

Figure 4. Example of modelling parts of the J2EE Reference PetShop application in SoftArch/Thin.

5. Generating and Running Test-beds

Our SoftArch/Thin environment models are used to fully-automatically synthesise performance test bed code, configure a 3rd party web application performance testing tool, and to co-ordinate the running of the tests and analysis and visualisation of test results. Figure 5 shows an outline of this process. The SoftArch/Thin model is converted to XML, and XSLT transformation scripts run over this XML to generate JSP, ASP, Java and C# code, along with build script, deployment script and testing tool script files (1). A deployment tool is used to upload this generated information to multiple hosts, each running a deployment tool server (2). The code is compiled and configuration scripts run to initialise the web server(s), application server(s) and database server(s) required by the architecture model under test. The thin-client testing tool, Microsoft™ Application Centre Test, is told to run the performance test (3). This behaves as one or more concurrent user client browsers making requests to the generated web server components. An analysis of the performance tests is shown to the architect (4).

The XML encoding of a part of a SoftArch/Thin architecture model is shown in Figure 6 (1). This example illustrates how the synthesis of a JSP web component to be performance tested is achieved. The XML file containing the software architecture model encodes various abstractions as XML tags and property values. In this example, an AppServer "j2ee_videoWebApp" has a RemoteObj (web component) "videoSearch", which is of Type "jsp" and defines a number of StatsReturned. A JSP page generator XSLT script (2) is run on this XML by SoftArch/Thin, transforming the XML-encoded architecture properties into a .jsp code file (3). Part of this particular transformation script is shown, a template that matches a StatesReturned/State record in the input XML model and transforms this into JSP page tags to display object state information. Usually such information has been returned from a database and stored in a "JavaBean" object.

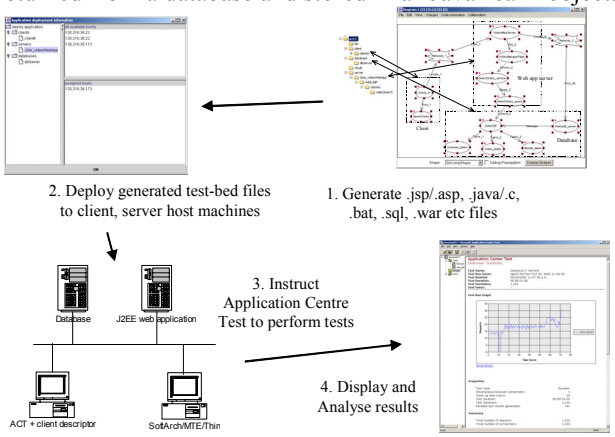

Figure 5. Generating thin-client performance test-beds.

The generated .jsp file (3) code has tags which instruct the JSP when accessed by a web browser to generate mark-up to display state variable values of a JavaBean object in HTML text fields. The JavaBean field names and text field names are those from the source .xml file StatesReturned/State records. Some of the XSLT scripts get quite complex and can generate large amounts of code from the model XML. Part of an Application Centre Test configuration script is shown (4), which is the "client" code generated by another XSLT script to test the performance of the generated web page.

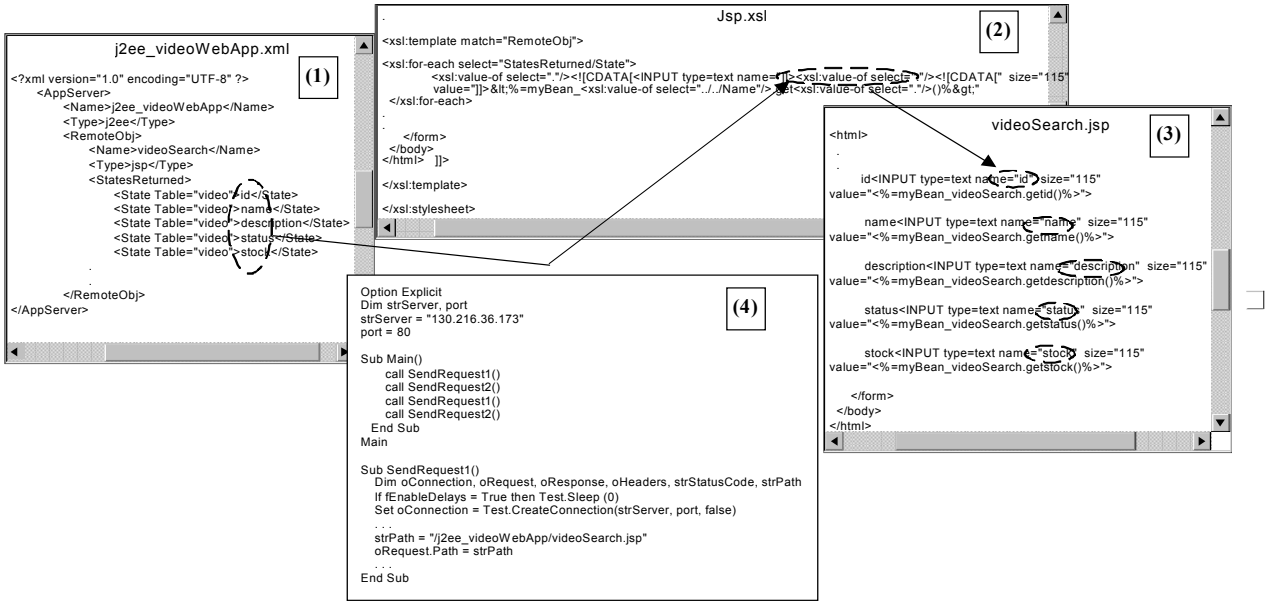

Figure 6. Examples of .jsp code and ACT script generation in SoftArch/Thin.

We chose to use the Microsoft™ Application Centre Test (ACT) tool to carry out the performance tests and basic analysis. We did also prototype a simple "psuedo-web browser", and a web client test application that used COM to instruct an IE5 browser, which can both perform the specified client requests. However we found the ACT tool provided the same capability to run concurrent performance tests and also offered useful results capture and visualisation facilities. SoftArch/Thin generates configuration scripts for ACT and also uses the deployment tool servers to invoke multiple instances of the tool running on different clients for large-scale performance tests.

Figure 7 shows two examples of ACT testing tool runs for generated architecture models for the PetShop application. The only difference between the models was that one specified "JSP" web components be generated and the other "ASP". The ACT testing tool displays in this example are showing number of requests performed per second over time along with summary information.

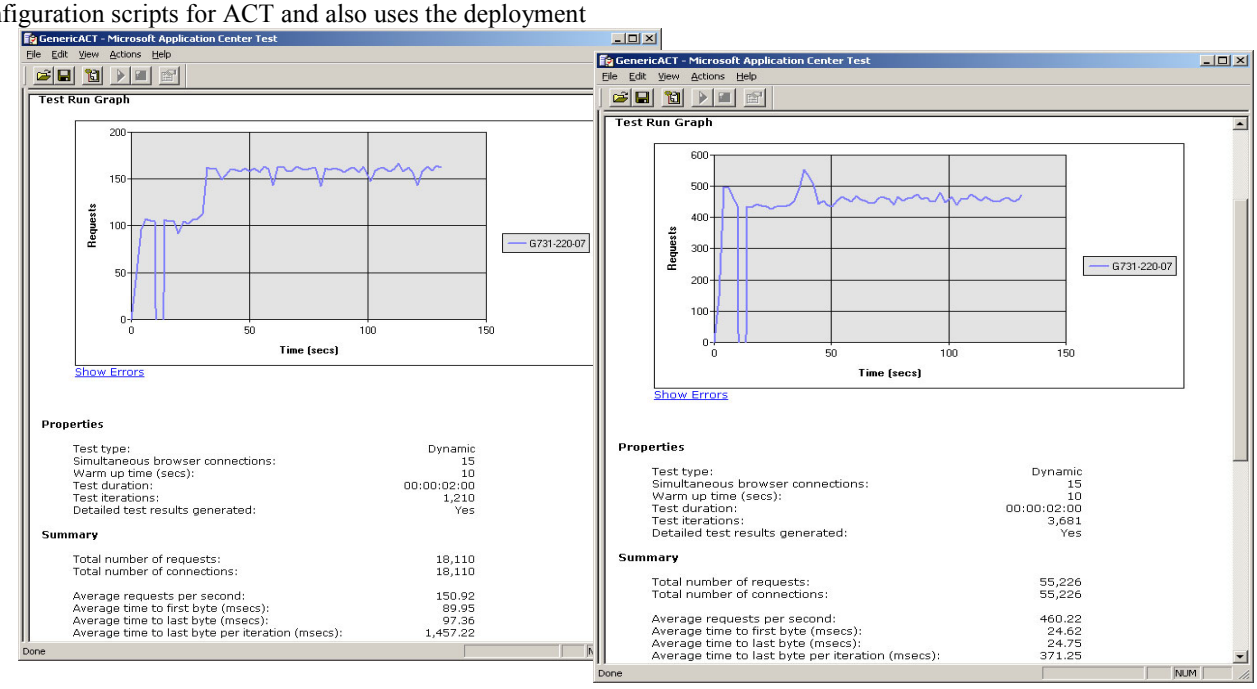

Figure 7. Example of using the ACT to analyse performance of generated systems.

Figure 8 shows a graph of average performance results in terms of milliseconds to complete for several generated JSP and C#/ASP.NET web components in the PetShop application example. These tests were run with three networked PCs, one each used to host the client (ACT tool), web server and components (JSPs/ASPs) and database (SQL Server 2000). The client requests and database server tables are identical, the middle-tier web components and servers being the difference. The C#/ASP.NET version performs much faster than the JSP version in this example. However, we used the Microsoft IIS web server, a commercial performance-optimised platform for the ASP.NET hosting, but used an un-optimised J2EE SDK application server to host the JSP web components.

The architect can modify various parameters in SoftArch/Thin to adjust their performance tests e.g. web component and database version to use, number of concurrent clients, number of times to perform a request, thread pool size, database table fields, and so on. They can also modify their architecture e.g. splitting or merging pages, adding or modifying web component database requests, or adding or modifying application server components and requests for multi-tier architectures. SoftArch/Thin regenerates the code and script files and re-runs the new performance tests automatically.

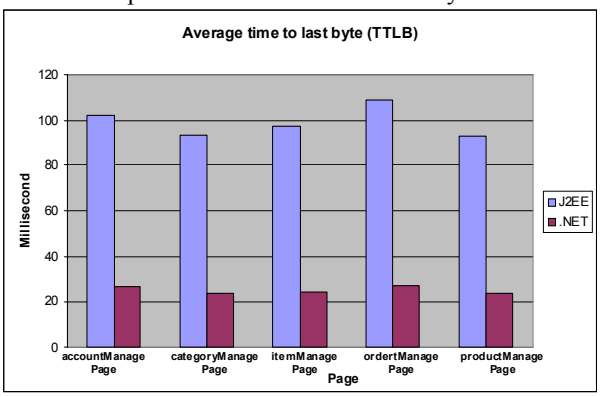

Figure 8. Comparison of J2EE vs ASP.NET PetShop application web page performance.

We compared the performance analysis results of our generated test-beds to some pre-existing, third-party code to illustrate the variation in performance between generated code and hand-written code. Figure 9 shows the performance test results of SoftArch/Thin-generated web components to that of a hand-implemented C#/ASP.NET-implemented version of the PetShop application, downloaded from the internet [13]. We ran the exact same ACT performance tests against the hand-implemented and generated web components. The hand-implemented application has additional application logic code which is not present in SoftArch/Thin, but most of the generated web component performance results are very close to the hand-implemented application ones. The main outlier is the orderManagePage, where the hand-implemented one has some complex transaction logic that is over-simplified in the generated version, hence the latter's performance is unrealistically fast.

Performance parameters	Real ASP.NET PetShop	SoftArch/Thin-generated PetShop
Overall average RPS (requests/second)	419.56	460.22
Overall average (ms)	28.67	24.75
accountManagePage	27.34	26.36
categoryManagePage	23.42	23.56
itemManagePage	23.74	24.34
orderManagePage	39.15	27.34
productManagePage	23.63	24.01

Figure 9. Real ASP.NET PetShop performance vs SoftArch/Thin-generated one.

6. Design and Implementation

The architecture of SoftArch/Thin is illustrated in Figure 10. We developed a set of meta-models which define thin-client architecture modelling abstractions available to architects. Architecture models developed with these abstractions are represented by an XML format, used as input to a set of XSLT transformation scripts. We used the Xalan XSLT engine to process these scripts to generate JSP, JavaBean, ASP, .NET component, batch scripts, configuration tool scripts and ACT scripts. These are uploaded to multiple remote client and server host machines and deployment scripts are run to compile, install and initialise web and application server code and database tables. The ACT clients deployed on client hosts are instructed to run by remote messages sent via the deployment tool servers. These run their specified performance tests, collecting and summarising results. If required, saved test results can be copied back to the SoftArch/Thin tool via the deployment servers for further analysis and aggregation.

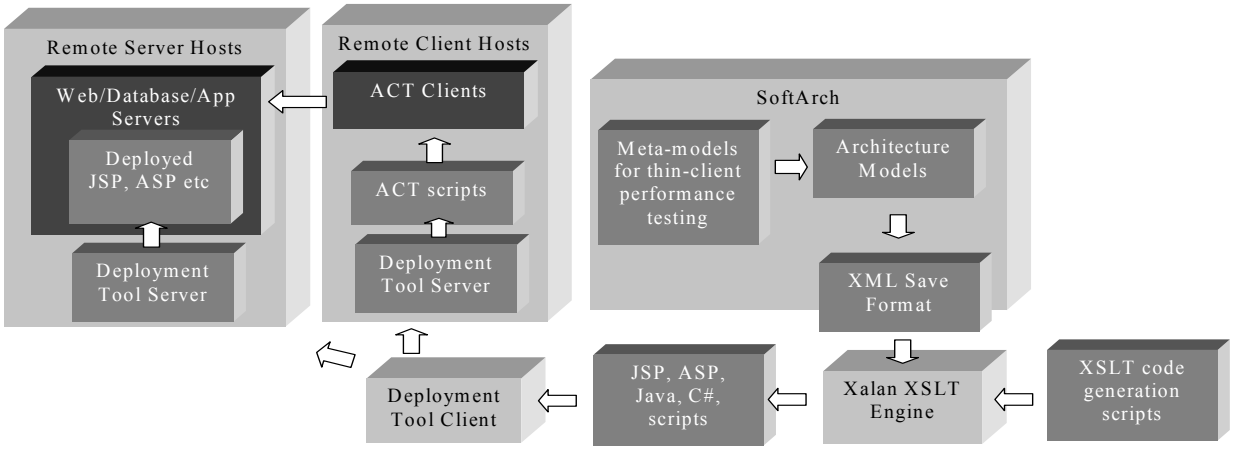

Figure 10. The architecture of SoftArch/Thin.

We implemented SoftArch/Thin using a Java application for the modelling tool, Xalan XSLT engine, a custom deployment tool, and a third-party thin-client application testing tool (Microsoft™ ACT). An existing software architecture modelling tool, SoftArch, was extended to support thin-client performance test generation by use of a set of meta-models providing domain-specific modelling abstractions and an XML save format for architecture models. We have found XSLT to be a good approach to code and script generation from XML source data. Some of our XSLT scripts are quite complex but scripts can be readily added and modified. This allows us to provide additional code generation support for new save file information without changing the modelling tool's implementation. The Application Centre Test thin-client performance testing tool proved to be a robust, flexible approach to carrying out the performance testing and provides basic results analysis to architects.

7. Discussion

7.1. Evaluation of SoftArch/Thin

We have used SoftArch/Thin to model several thin-client architecture applications and to provide performance testing of these models. These applications have included an on-line video search and rental library, the J2EE PetShop reference application, and a complex micro-payment system [2]. With the video application we experimented with several different architectural arrangements including 3-tier JSP/database, 4-tier JSP/CORBA/database and multi-tier solutions. We also ran performance tests using different database technologies and web browsers. With all three applications we generated both J2EE and C#/ASP.NET implementations of the web and application server components and compared their performance. We had pre-existing hand-implemented versions of each of these systems so we could compare the actual performance of to our generated web component performance measures.

With each of the SoftArch/Thin generated web component and application server component implementations we gathered performance test results. For most we used the Microsoft™ ACT tool, but to test performance with different browser clients we used a client proxy which drove the browser using COM. We then gathered performance test results for the real, hand-implemented versions of these applications and compared the actual performance of the applications to the results obtained by using SoftArch/Thin.

In general, the performance of the generated thin-client applications was close to that of the real, hand-implemented applications, for the most part within 15-30% of the actual application speed under the same ACT-managed tests. This result depended on the level of detail in the SoftArch/Thin models: very abstract, simplified architecture models produced correspondingly less accurate performance results. It also depended on the complexity of the application logic in the real system: the more complex the logic, the less accurate in general the generated web component performance test results. Some major performance differences were found in the micro-payment application due to the implementation approach used in the real application for CORBA remote object access and the parameter types passed between web component and application server components. Application components with complex transaction logic also tended to produce less accurate results as the generated components use only simple database and distributed object transaction models.

7.2. Advantages and Limitations

We have found that SoftArch/Thin provides a useful environment for sketching software architectures and generating performance test-beds to gain an understanding of the possible performance of a system using such an architecture. Software architects can sketch an architecture

design quickly and have significant amounts of code generated which test the performance of web server, application server and database performance under a wide variety of user-specified loading conditions. A range of architectural and implementation features can be modelled and appropriate code generated without any user intervention. The ACT suite provides a useful third-party tool for load-testing the generated systems and reporting performance test-bed results to the user.

However, our approach does have several limitations. While more detailed architecture designs produce more detailed generated test-bed code, as all of the code is generated from the high-level design the performance results obtained will never be exactly the same as those from a fully-implemented system based on the design. Hence the results will always only provide a guideline to the architect. Approaches that measure performance of hand-implemented code will always be able to provide more accurate results, if the users are willing to expend the effort involved in developing the prototype system.

SoftArch/Thin requires users to use its own proprietary software architecture design notation. We developed this notation as part of another project which investigated enrichening visual architecture description languages. Because this notation is non-standard, it may be useful to investigate a more common architecture representational approach, for example using an extended version of appropriate UML diagrams.

Currently SoftArch/Thin users are expected to specify all components of an architecture with the tool in order to generate performance test-bed code. In addition, its current support for results visualisation, storing and reusing previous test-run results and comparing multiple test-run results are very limited. The use of the Application Centre Test tool suite limits the range of performance statistical analysis results that can be presented to users. While we have applied the tool to a number of thin-client software architecture examples, it has limitations in terms of the size of the architectures that can be sensibly modelled in the tool and the results visualisation facilities for complex architectures.

8.3. Future Work

There are a number of enhancements that could be made to SoftArch/Thin to improve the facilities it provides and the usefulness of the performance measures that it produces. Automated visualisation of performance results within SoftArch/Thin architecture diagrams could be used to convey *in situ* summarises of different architecture element performance characteristics. Similarly, integrated graphing and results comparison would enable users to see multiple test run results for different architecture and implementation decisions within the tool. Generating more complex code, particularly to support complex transaction logic and caching of data would allow more realistic results to be obtained from generated server code. Allowing users to specify ranges of values for testing parameters e.g. 5-10 concurrent clients; findVideo invoked 10-15 times and returning between 3-10 rows, would allow multiple test configurations to be generated in one go. Ranges of averaged performance values could then be collected and presented rather than a single average performance measure as at present. We would like to add other measures than just transaction throughput in the future, such as average CPU/memory usage, disk performance and so on.

8. Summary

Determining the likely run-time performance of thin-client applications is very difficult. We have described a tool, SoftArch/Thin, which provides support for test-bed based thin-client application performance analysis. A high level software architecture model is used to generate web server components, application server components and database tables, along with compilation and configuration scripts. This generated code is uploaded to distributed client and server hosts and performance tests run on the web components using the Microsoft™ Application Centre Test tool. We have successfully used SoftArch/Thin to model several thin-client architectures and obtain realistic performance measures via generated test bed code.

References

1. Balsamo, S., Simeoni, M., Bernado, M. Combining Stochastic Process Algebras and Queueing Networks for Software Architecture analysis, In *Proceedings of the 3rd International Workshop on Software and Performance*, Rome, Italy, July 214-26 2002, ACM Press.
2. Dai, X. and Grundy, J.C. Customer perceptions of a thin-client micro-payment system: issues and experiences, *Journal of End User Computing*, Vol. 15, No. 4, Idea Publishing Group.
3. ECPerf Performance Benchmarks, August 2002, ecperf.theserverside.com/ecperf.
4. Gomaa, H., Menascé, D., and Kerschberg, L. A Software Architectural Design Method for Large-Scale Distributed Information Systems, *Distributed Systems Engineering Journal*, Sept. 1996, IEE/BCS.
5. Gorton, I. And Liu, A. Evaluating Enterprise Java Bean Technology, In *Proceedings of Software - Methods and Tools*, Wollongong, Australia, Nov 6-9 2000, IEEE.
6. *Grundy, J.C. and Hosking, J.G. SoftArch: Tool support for integrated software architecture development,* International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, No. 2, April 2003, World Scientific, pp. 125-152.*
7. Grundy, J.C., Cai, Y. and Liu, A. Generation of Distributed System Test-beds from High-level Software Architecture Descriptions, In Proceedings of the 2001 IEEE International Conference on Automated Software Engineering, San Diego, CA, Nov 26-29 2001.
8. Hansen, K. Loading testing your applications with Apache JMeter, http://javaboutique.internet.com/tutorials/JMeter/.

9. Hu L., Gorton, I. A performance prototyping approach to designing concurrent software architectures, In *Proceedings of the 2nd International Workshop on Software Engineering for Parallel and Distributed Systems*, IEEE, pp. 270 – 276.
10. Juiz, C., Puigjaner, R. Performance modelling of pools in soft real-time design architectures, *Simulation Practice & Theory,* vol.9, no.3-5, 15 April 2002, Elsevier, pp.215-40.
11. Kazman, R. Tool support for architecture analysis and design, In Proceedings of the Second International Workshop on Software Architectures, ACM Press, 94-97.
12. McCann, J.A., Manning, K.J. Tool to evaluate performance in distributed heterogeneous processing. In *Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing*, IEEE, 1998, pp.180-185.
13. MSDN, Using .NET to implement Sun Microsystem's Java Pet Store J2EE BluePrint application, October 2002, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/psimp.asp.
14. Nimmagadda, S., Liyanaarachchi, C., Gopinath, A., Niehaus, D. and Kaushal, A. Performance patterns: automated scenario based ORB performance evaluation, In Proceedings of the Fifth USENIX Conference on Object-Oriented Technologies and Systems, USENIX, 1999, pp.15-28.
15. Petriu, D., Amer, H., Majumdar, S., Abdull-Fatah, I. Using analytic models for predicting middleware performance. In *Proceedings of the Second International Workshop on Software and Performance*, ACM 2000, pp.189-94.
16. Subraya, B.M., Subrahmanya, S.V. Object driven performance testing of Web applications, In Proceedings of the First Asia-Pacific Conference on Quality Software, IEEE CS Press, pp.17-26.
17. Topol, B., Stasko, J. and Sunderam, V., PVaniM: A Tool for Visualization in Network Computing Environments, *Concurrency: Practice & Experience*, Vol. 10, No. 14, 1998, pp. 1197-1222.
18. Web Application Testing, WAPT Version 2.0, http://www.loadtestingtool.com/.