Revel8or: Model Driven Capacity Planning Tool Suite

Liming Zhu[1,2], Yan Liu[1,2], Ngoc Bao Bui[1,2],Ian Gorton[3]
[1]*Empirical Software Engineering Program, National ICT Australia Ltd.*
[2]*School of Computer Science and Engineering, University of New South Wales, Australia*
{ Liming.Zhu, Jenny.Liu, Betty.Bui}@nicta.com.au.au
[3]*Pacific Northwest National Laboratory*
ian.gorton@pnl.gov

Abstract

Designing complex multi-tier applications that must meet strict performance requirements is a challenging software engineering problem. Ideally, the application architect could derive accurate performance predictions early in the project life-cycle, leveraging initial application design-level models and a description of the target software and hardware platforms. To this end, we have developed a capacity planning tool suite for component-based applications, called Revel8tor. The tool adheres to the model driven development paradigm and supports benchmarking and performance prediction for J2EE, .Net and Web services platforms. The suite is composed of three different tools: MDAPerf, MDABench and DSLBench. MDAPerf allows annotation of design diagrams and derives performance analysis models. MDABench allows a customized benchmark application to be modeled in the UML 2.0 Testing Profile and automatically generates a deployable application, with measurement automatically conducted. DSLBench allows the same benchmark modeling and generation to be conducted using a simple performance engineering Domain Specific Language (DSL) in Microsoft Visual Studio. DSLBench integrates with Visual Studio and reuses its load testing infrastructure. Together, the tool suite can assist capacity planning across platforms in an automated fashion.

1 Introduction

Capacity planning is a challenging software engineering problem, especially in early life cycle when only design diagrams and prototypes are available. Various performance analysis techniques with prediction capabilities have been proposed to evaluate architecture designs [1]. However, they have not been widely adopted in industry for a number of reasons. Probably the biggest hurdle to adoption is the fact that strong performance modeling and mathematical skills are required for deriving analytical models from an application design. This creates an extra learning curve as software engineers must master the theories of performance modeling. Moreover, obtaining parameter values in order to populate an analytical model requires significant engineering effort. In addition, these activities are not integrated into day to day software development environments.

We believe that Model Driven Development (MDD) technologies and tight integration with development environment can solve these problems. The design of Revel8or addresses each of these problems as follows:

1) MDAPerf uses standards such as the UML performance and scheduling profile for performance annotations. This alleviates the need to learn a new annotation for each new performance analysis technique. Users work with their normal design diagrams directly. Most of the complexity for analytical performance modeling is hidden behind model to model transformations.
2) MDABench and DSLBench allow benchmark applications including test case data to be modeled and generated rather than be built manually. Performance data collection facilities are also generated automatically.
3) The Revel8or toolkit is tightly integrated with the most popular development environments. MDAPerf is an Eclipse plug-in. MDABench works with any UML modeling tool that can export XMI. DSLBench is essentially a Visual Studio Plug-in utilizing the Microsoft Domain Specific Language (DSL) toolkit. DSLBench also integrates with the load testing capability in Visual Studio by providing extra modeling capabilities for wizard and script based testing.

In addition to the above features, the Revel8or tool suite has a number of unique benefits. In practice, capacity planning is often done using industry standard benchmark results rather than application specific models and measurements. It is however difficult to use these industry standard benchmark results to sensibly infer useful performance characteristics about the application under design. In Revel8or, both the generated benchmark and the analysis model are based on a design that closely corresponds to the application of interest, and hence it captures the unique characteristics of the application. This leads to the capacity planning producing more representative measures and predictions of the eventual application.

Revel8or derives the benchmark application from the same application model that the performance analysis

model is derived from. This makes it possible to use the performance measurement data for analytical model validation and tuning model parameter values.

With MDAbench and DSLbench supporting design models in UML and DSL respectively, we provide alternatives for software architects and engineers in benchmark application modeling and performance test bed auto-generation. Notice that we do not favor either UML or DSL in our modeling capability. Instead, either can be used in the most appropriate context such as the preference of architects or constraints of platforms.

2 Related Work

Some researchers have applied model driven approaches in performance analysis [4, 6]and typically focus on deriving analytical models from UML design models through XSLT. This means the mapping and transformation information is ad-hoc and tangled within the XML query and transformation language, which has limitations for representing and validating mapping relationships. In Revel8or, the transformation from design artifacts to analytical models is supported by a meta-modeling framework. From an extensibility and portability point of view, the persistent format of models and its transformation follows the specification in [5], so that this tool can be integrated with the different analysis methods that support this specification.

Measurements in the form of benchmarking and prototyping [2]are used to obtain valuable information for populating analytical model parameters. Comprehensive prototyping can however be expensive. To further exacerbate the problem, multiple benchmarking and prototyping applications need to be constructed for different target platforms for comparison or deployment considerations. Our approach integrates customized benchmark generation and performance analysis model transformation into a single model driven capacity planning environment.

Some pioneering work has been done on generating benchmark and prototyping applications using models, as in [2]. However, these have several limitations:

- The code generators for the chosen technologies are built from scratch by the researchers. Any change to the chosen target technology or the introduction of a new technology requires significant extra work from the researchers. In contrast, Revel8or exploits existing code generation “cartridges” from AndroMDA and load testing infrastructure within Visual Studio.
- These methods do not usually follow standards or they favor one environment (e.g. Eclipse) over another (Visual Studio). The UML annotations in Revel8tor are based on standards. Revel8or uses two standard UML profiles on testing and performance/scheduling. Revel8or supports both UML and DSL modeling environments, including various UML 2.0 modeling tools and Visual Studio across J2EE, .Net and Web Services platforms.
- In these methods, the load testing part of the benchmark suite can not be comprehensively modeled. It usually relies on scripting or manual coding. Test case data is either embedded in code or test cases rather than specified using a data model. Revel8or distinguishes different testing elements (e.g. test context, test cases, data pool, data partition) by tailoring the UML 2.0 Testing Profile. In addition, a new simpler load testing DSL has been designed from scratch using Visual Studio DSL.

3 Tool Design and Implementation

As shown in Figure 1, the Revel8or tool suite consists of three individual tools: MDAPerf, MDABench and DSLBench. They work together through a common XML-based file exchange format. The demo will show features of the individual tools and also their integration.

3.1 MDAPerf

MDAPerf is essentially a tool to annotate design diagrams, along with a built-in engine responsible for deriving and solving a performance analysis model, currently a Queuing Network Model (QNM).

Software architects can annotate use case diagrams, sequence diagrams and deployment diagrams with the UML performance and scheduling profile. The tool then navigates through the diagrams, scans annotations and collects property values. These design specifications can be exported to text-based XMI files, which capture the semantics and tagged values in the UML diagrams. These XMI files are then manipulated by the prediction engine to derive a QNM and subsequently solve it.

MDAPerf is currently built as an Eclipse plug-in. It includes three major components:

- QNM generator: this derives a QNM that represents the structure of the application deployment based on the type of resources and workload. The transformation from UML diagrams to QNM utilizes the UML Performance and Scheduling Profile, so that this tool can be integrated with different methods.
- Service demand calculator: this calculates the service demand of each scenario on each resource
- QNM solver: this implements different algorithms to solve different type of QNMs. We have implemented exact and approximate algorithms for both open and closed QNMs with single class or multiple-classes workload.

3.2 MDABench

MDABench follows OMG’s Model Driven Architecture (MDA) to generate a deployable benchmark. application from design artifacts. The OMG’s MDA standard defines a way of transforming domain models into Platform Independent Models (PIM) and then

Platform Specific Models (PSM), and eventually to executable code. PSMs can be used to generate benchmark applications using specific technologies, such as J2EE/EJB or Web Service

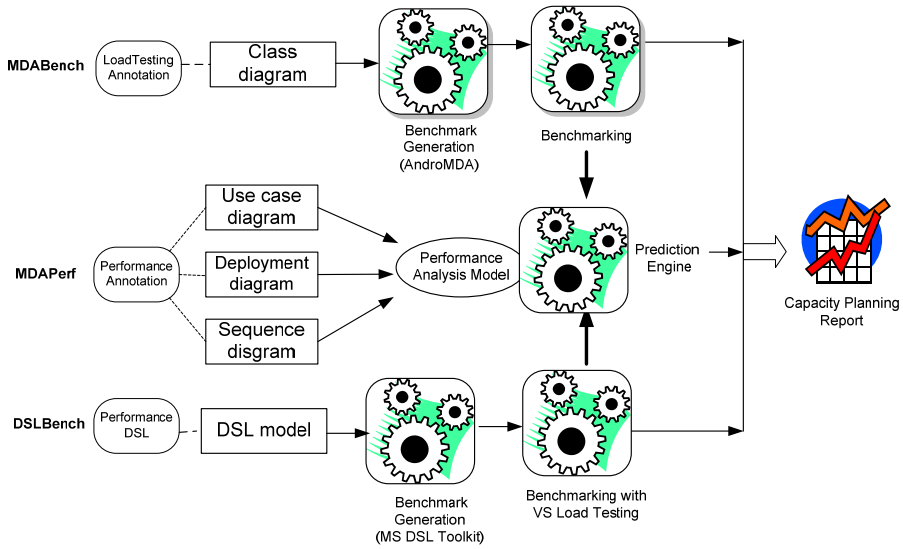

Figure 1. Revel8or Model Driven Capacity Planning Toolkit

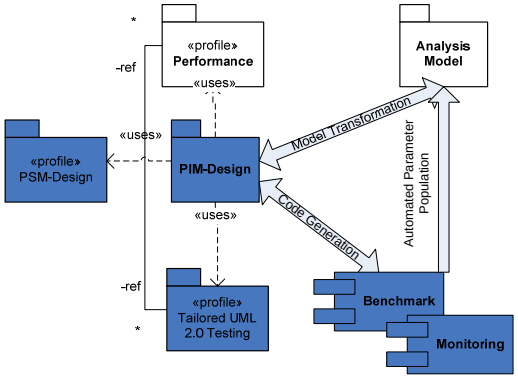

Figure 2 MDABench

As shown in Figure 2, the benchmark UML design model begins with a PIM which reflects the application logic. The benchmark UML design model is then annotated with UML profiles for code generation, and a load testing suite is modeled using the UML 2.0 Testing Profile. The resulting UML model is then exported using XMI and becomes an input to the open source AndroMDA code generation framework. We have extended the AndroMDA framework with a new cartridge to generate a load testing suite including default test cases and associated performance monitoring functionality. The same design model is also used for deriving the analysis model. Such integration enables better analysis model parameter population and consequently provides more accurate capacity planning results.

3.3 DSLBench

DSLBench is the counterpart of MDAbench implemented using Visual Studio DSL. A DSL language defines domain concepts, rules, and constraints that govern the use of the domain concepts and graphical notations associated with them. The conceptual part of the DSL for benchmarking has been abstracted from the performance testing domain through extensively surveying existing benchmarking frameworks, such as ECPerf, Grinder and Visual Studio load testing features. The common concepts and best practices from these are then grouped together to form the basis for our DSL, as shown in Figure 3. Though requiring a new syntax, the semantics of this DSL are designed to be compliant with the UML 2.0 Testing profile, and thus comparable with MDABench.

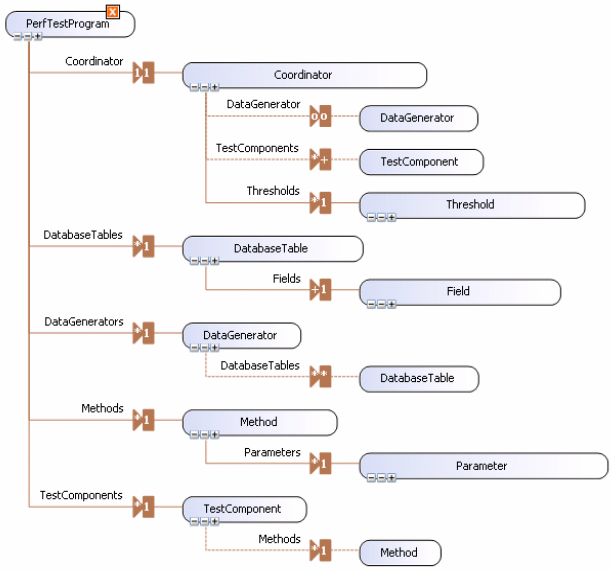

Figure 3: Meta-model for performance testing domain

DSLBench integrates with Visual Studio and supports the following activities:

- Modeling key scenarios and test cases
- Specifying the metrics to be collected
- Work load and load simulation
- Test case generation
- Specifying resource utilization threshold

The performance testing metamodel of DSLBench is generic and not particularly specific to Visual Studio environment. However, DSLBench only currently supports benchmark code generation in C# and deployment in .Net platform.

4 Evaluation

An evaluation of the Revel8or suite was done through a field trail of each individual tool, as described in the following.

MDAPerf

In MDAPerf, the accuracy of the performance analysis and prediction relies on several aspects, including

1. Design models that accurately capture the performance characteristics of the system to be implemented and deployed.
2. The transformation of design models to QNMs.
3. The solution of QNMs.

The first aspect is application specific and depends on the architect's expertise and knowledge of the overall architecture of the application and its performance characteristics. The next two aspects are generic and validated by our previous work in [3], from which the implementation of model transformation and solutions are reused. MDAPerf enhances [3] by providing an IDE for both architecture design and performance prediction, and automating the process.

MDABench

MDABench has been evaluated to generate deployable benchmark applications using J2EE and Web Services technologies on different application servers, including JBoss Application Server, BEA's WebLogic Server and Apache's Tomcat/Axis Web server.

We recently had the opportunity to test MDABench in a Web service-based e-Government project. The system allowed Australian tax payers to retrieve their medical costs for a given tax year directly from a Web service for lodging a tax return. Our aim was to assess the performance potential of the Web services involved. We were able to use the MDABench prototype in the measurement planning phase. We created test data models and specified the transaction mix, exception mix and measurement requirements. We then used the model to communicate the essential measurements to system's software engineers. This exercise has given us considerable insights into using such a tool in real world.

DSLBench

The same sets of benchmark applications have been implemented in DSLBench. The integration with Windows platform utilities such as performance counters through Visual Studio provides monitoring and reporting capability for DSLBench.

Both MDABench and DSLBench have been validated by deploying and running the generated benchmark applications. Performance measurements are obtained and they are consistent with results collected from manually implemented benchmark applications.

From our experience in evaluation, one advantage of Revel8or is that it significantly reduces the engineering effort in comparing an application's performance using different architectures, implementation technologies or platforms. This is due to the fact that such changes can be reflected by modifications to the design models, and the changes can be transformed into analytical models or benchmark code with minimal engineering activities.

5 Conclusion

In this paper we have described a model driven capacity planning tool suite, Revel8or for component and web service based applications. The core principle behind this research is to automatically transform design artifacts of an application into platform specific solutions for analyzing and predicting its performance. These solutions include performance models and benchmark applications. The unique feature of Revel8or is that it integrates performance analysis with automated benchmark generation. It provides architects with tool support to predict performance from a design in the early stage of software development, which reduces the engineering effort involved in capacity planning. Our future works includes applying Revel8or to more realistic case studies.

6 Reference

[1] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: a survey," *IEEE Transactions on Software Engineering*, vol. 30 (5), pp. 295-310, 2004.

[2] J. Grundy, Y. Cai, and A. Liu, "Generation of distributed system test-beds from high-level software architecture descriptions," in Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE), 2001.

[3] Y. Liu, A. Fekete, and I. Gorton, "Design-Level Performance Prediction of Component-Based Applications," *IEEE Transactions on Software Engineering*, vol. 31 (11), pp. 928-941, 2005.

[4] M. J. Rutherford and A. L. Wolf, "Integrating a Performance Analysis Kit into Model-Driven Development," in Proceedings of the the 5th GPCE Young Researchers Workshop 2003, Erfurt, Germany, 2003.

[5] C. U. Smith and L. G. Williams, "Performance Model Interchange Format (PMIF 2.0): XML Definition and Implementation," in Proceedings of the 1st Int. Conference on the Quantitative Evaluation of Systems, 2004.

[6] C. Yilmaz, A. M. Memon, A. A. Porter, A. S. Krishna, D. C. Schmidt, A. Gokhale, and B. Natarajan, "Preserving distributed systems critical properties: a model-driven approach," *Software, IEEE*, vol. 21 (6), pp. 32-40, 2004.

7 Appendix

Structure of the Demo

Introduction (Slides)

- Model driven capacity planning
- Model Driven approaches - DSL vs. UML
- Why do we need domain specific languages?

- Designing DSMLs using UML profiles.
- Designing DSMLs using Microsoft DSL

MDAPerf (Demonstration)

- Introduction of UML Performance and Scheduling Profile
- Introduction of EclipseUML
- Annotating use case, sequence and deployment diagrams
- Deriving Queuing Network Model (QNM)
- XML-based QNM
- Populating parameters
- Performance prediction results

MDABench (Demonstration)

- Introduction of UML 2.0 Testing Profile
- Introduction of AndroMDA
- Internal design of MDABench
- Modeling core benchmark logic
- Modeling load testing using UML 2.0 Testing profile
- Test data modelling
- Benchmark generation and data collection
- Benchmark report

DSLBench (Demonstration)

- Introduction of Microsoft DSL
- Meta modeling in DSL for performance testing domain
- Internal design of DSLBench
- Modeling performance testing using DSL
- Integration with VS load testing
- Configuring counter threshold for capacity planning
- Test data modeling
- Benchmark generation
- Running Benchmark within Visual Studio
- Benchmark Report

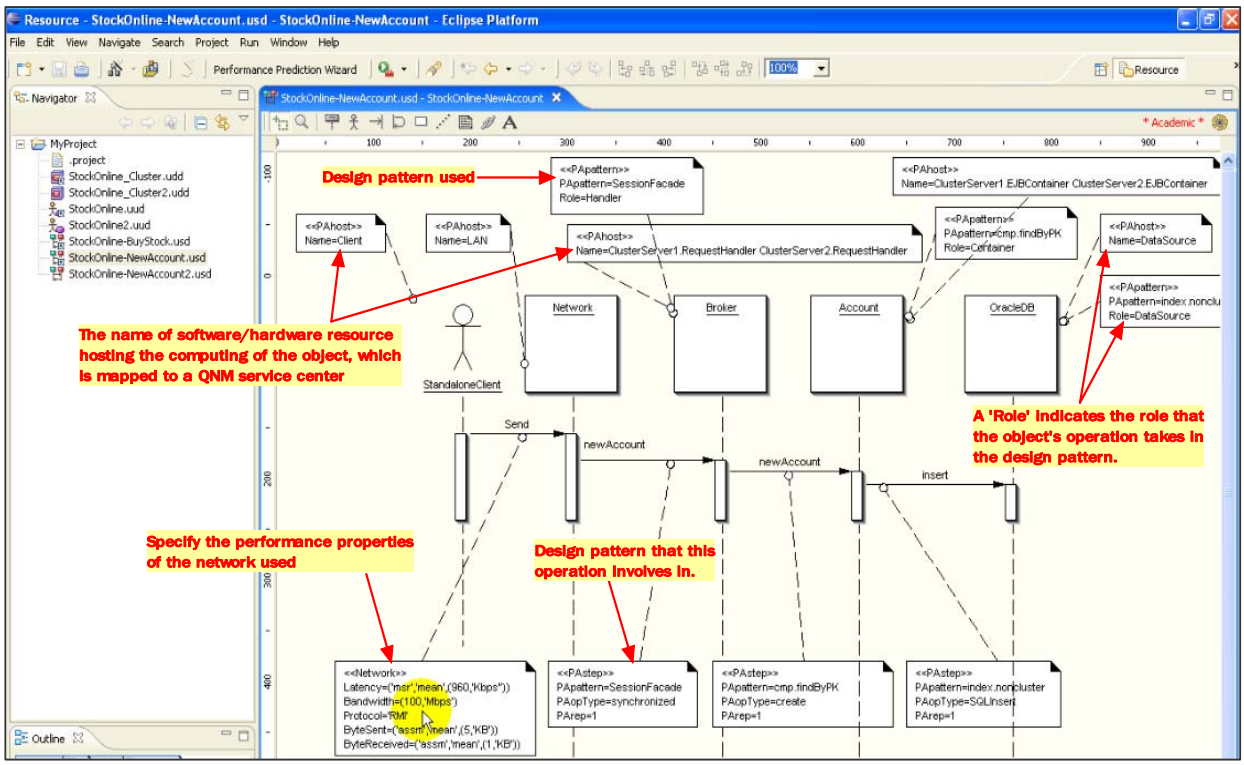

Figure 4. Annotating Sequence Diagram in MDAPerf

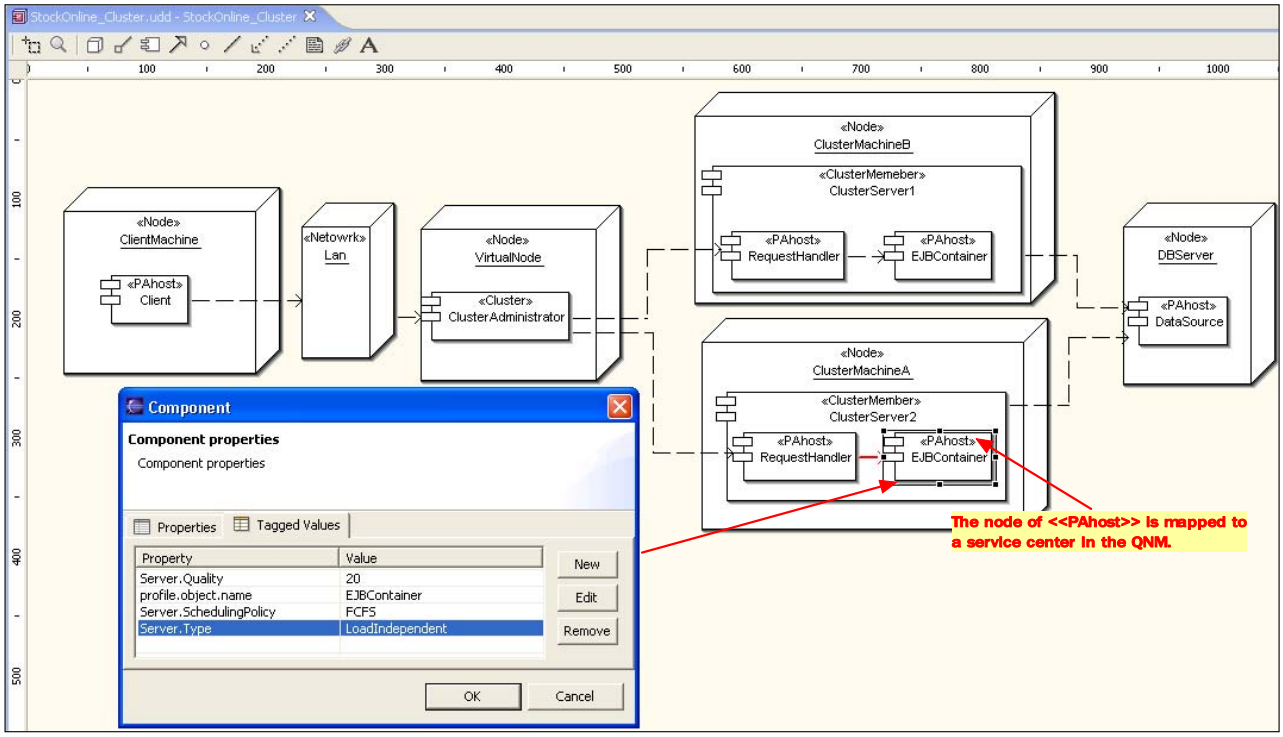

Figure 5. Annotating Deployment Diagram in MDAPerf

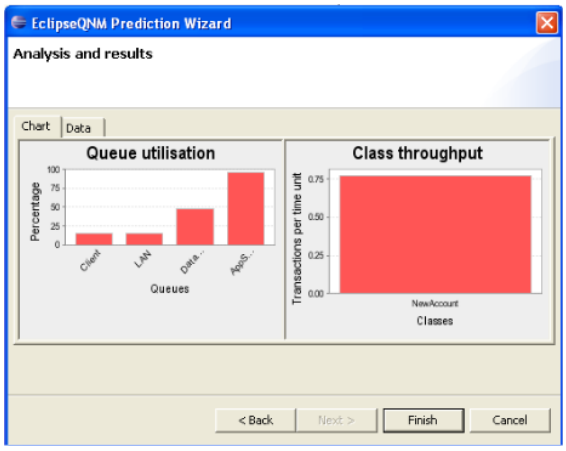

Figure 6. MDAPerf Analysis Results

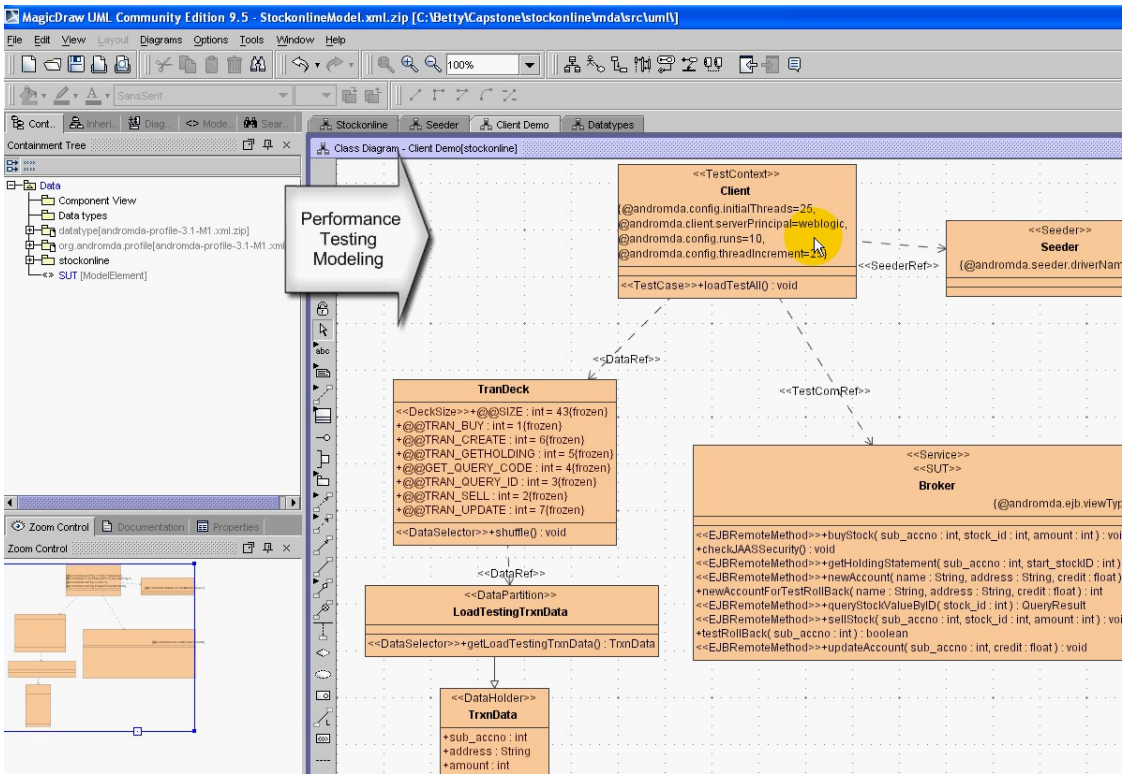

Figure 7. Performance Testing Modeling in MDABench

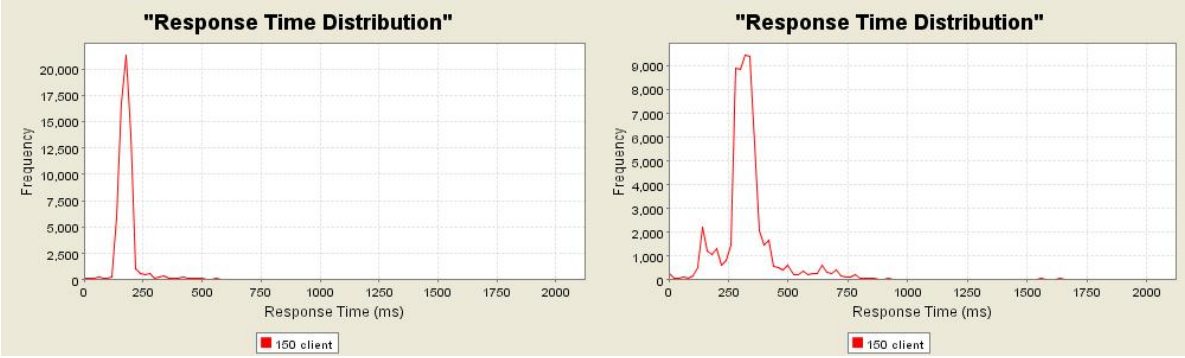

Figure 8 Benchmarking Results in MDABench

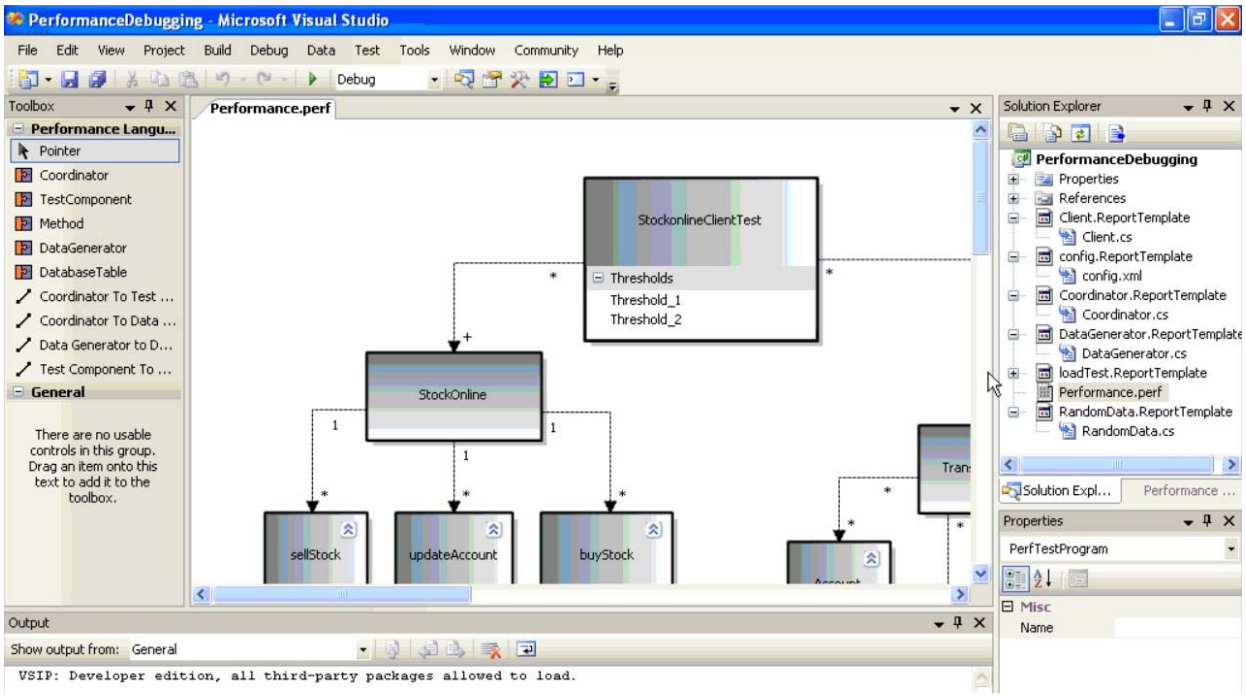

Figure 9 Performance Modeling in DSLBench

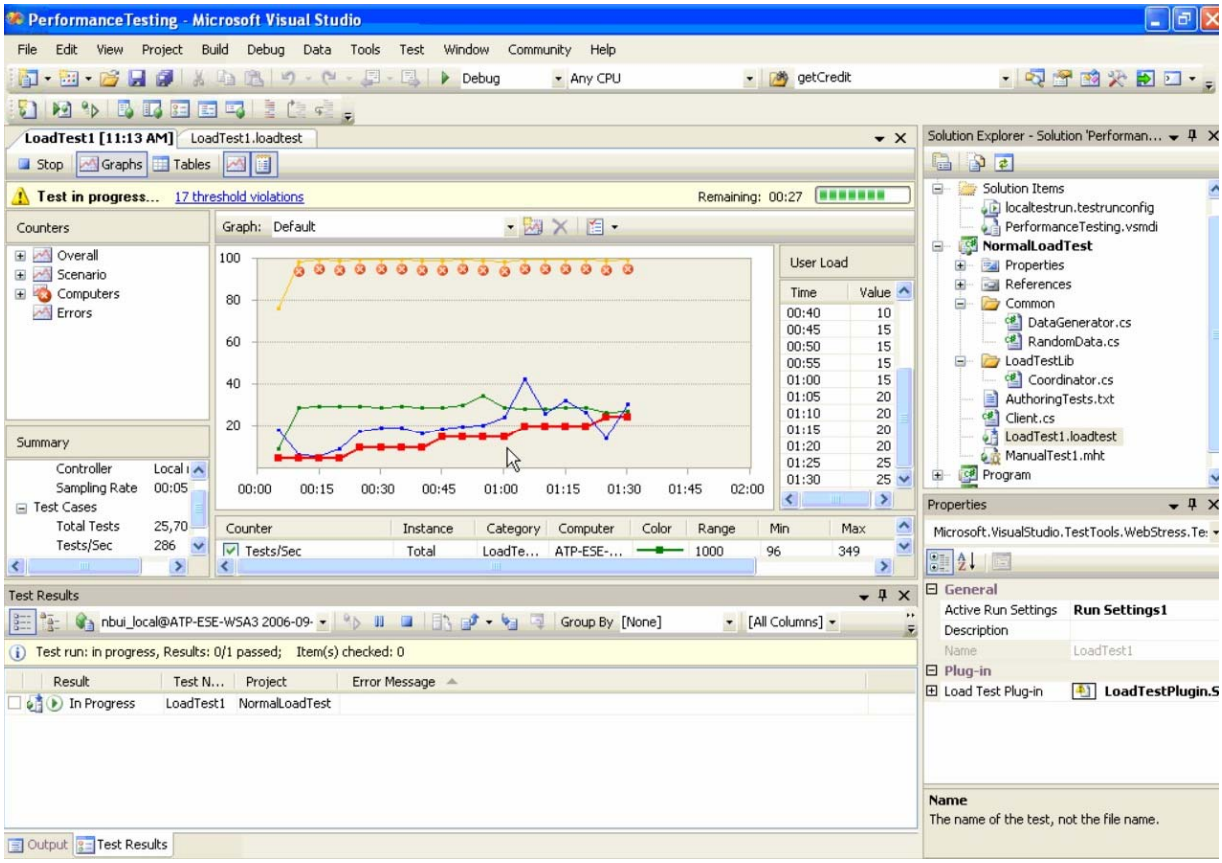

Figure 10 Benchmarking Results in DSLBench (integrated with VS load testing)