The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Model Driven Benchmark Generation for Web Services

Liming Zhu, Ian Gorton, Yan Liu
Empirical Software Engineering Program,
National ICT Australia
&
School of Computer Science and Engineering
University of New South Wales

Ngoc Bao Bui
Faculty of Information Technology,
University of Technology Sydney, Australia

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members
NICTA Partners

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Outline

- Unique Challenges in Web Services Performance Testing
- How Model Driven Development (MDD) Can Help
- Our Approach
  - Use OMG's MDA standards and an open source MDA framework implementation – AndroMDA
  - Design a domain specific "modelling" language for WS performance testing
  - Provide data collection and performance monitoring infrastructure
  - Generate deployable benchmark application
- Conclusion
- Limitations and Future Work

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Unique Challenges in WS Performance Testing

- Industry Observation
  - Industry reports e.g. "Web Services: A Nightmare for Testers"
- Limited Control Over
  - How services are consumed
  - Unforeseen usage scenarios
- Complexity
  - Coarse grained services (require one operation to handle vastly different payloads)
  - Large number of test combinations (endpoints, operations and data)
- Service Quality and SLA (Service Level Agreement)
  - Complicated: Web Services + Backend system + Networks (often Internet)
  - Ongoing monitoring/auditing during operation
  - Comformant with the standards
- Business (Service coordination essentially supports business transactions)
  - Service performance monitoring reflects critical business performance
  - Quickly build and performance test new services

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

How MDD Can Help - 1/2

MDD is about raising the level of abstraction for software development, providing more powerful concepts for capturing and reusing knowledge in a specific domain.

We apply MDD to the performance testing domain.

- Raise the level of abstraction
  - Have all the basic benefits of using models
  - Integrate with other models (design, analysis and business)
- Design a Domain Specific Modeling Language (DSML) for performance testing
  - Encourage best practices and modular designs through the language
  - Gain productivity through using a DSML
  - Decouple and reuse modules such as testing data, testing logic

How MDD Can Help - 2/2

- Use code generation "cartridge"
  - Hide Complexity and integrate best practices
  - Allow users to inherit best practices
- Use standards if possible
  - Leverage existing expertise
    - UML and Tooling
    - UML 2.0 Testing Profile
    - UML Profile for Schedulability, Performance and Time.
  - Integrate with SDLC
- Provide monitoring platform infrastructure
  - Save effort
  - Vendor neutral
  - Layered approach (monitoring of components, Web services, service coordination and business processes)

The imagination driving Australia's ICT future.

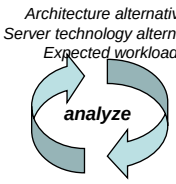
Our Approach - An Overview 1/3

MDD Approaches

• OMG's MDA standards (We follow MDA and use AndroMDA)

• proprietary approaches (MSFT DSL, Metacase, openArchitectureWare …)

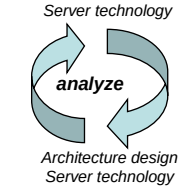
Architecture Design



MDAPerf

Architecture-based performance modeling and prediction tool

Benchmark



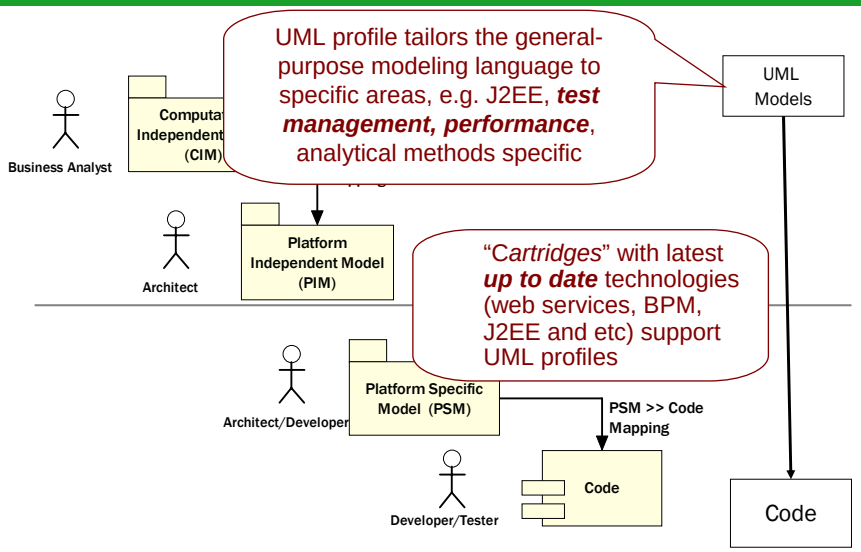
MDABench

Application benchmarking development toolkit

The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

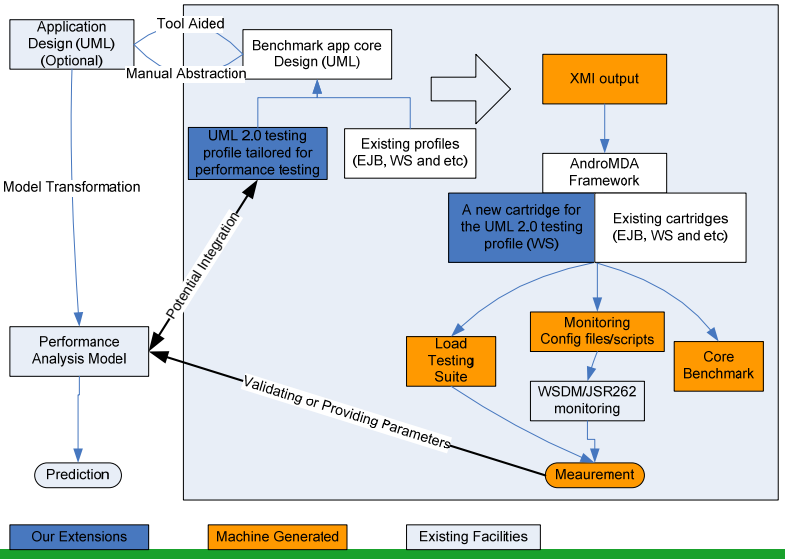
Our Approach - An Overview 2/3



The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Our Approach - An Overview 3/3



Integrate Best Practices

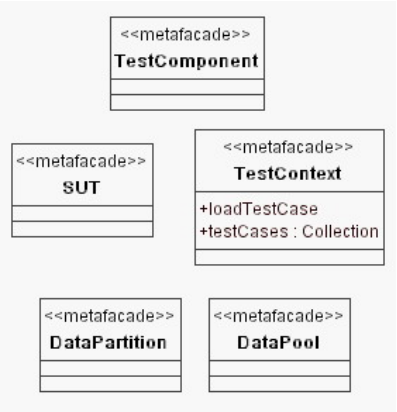
- Internal Designs
  - SPEC jAppServer /ECperf : clients, controllers and drivers
- Performance Metrics
  - TCP-W v2
    - SIRT (Web Service Interaction Response Time
    - SIPS (Service Interactions Per Second)
  - Distribution Statistics
  - Timing details
- Load Testing Configuration
  - Grinder 3
  - MSFT Visual Studio Load Testing

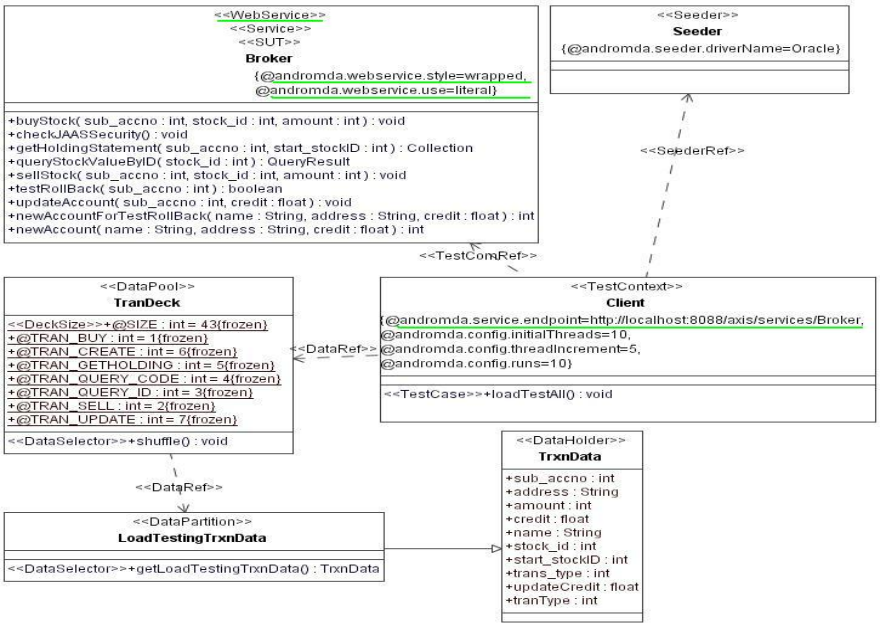
The imagination driving Australia's ICT future.

UML Profiles for Performance Testing

Extending UML 2.0 Testing Profile

- **Stereotypes**
  - SUT: The application to be test
  - TestContext: A collection of test cases
  - TestComponent: Classes of the application to be test
  - DataPool: A collection of explicit values that are used by a test context or test components during testing.
  - Data Partition: Logic values for a method parameter used in testing
- **Tagged Values**
  - Testing Control
    - config.InitialProcesses
    - config.processIncrement
    - config.processIncrementInterval
    - config.stablizationperiod
  - What to measure





The imagination driving Australia's ICT future.

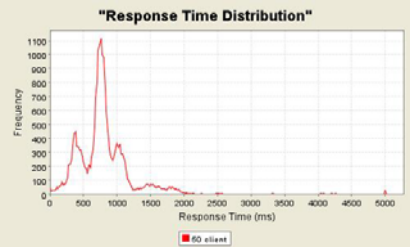
Performance Monitoring

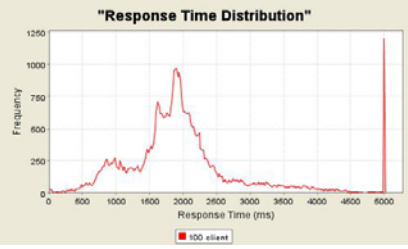
- Client side: Response time average/distribution/graph and throughput
- Server side : platform specific performance data
  - Through both consoles and generated scripts
  - Hide platform specific details in model to scripts generation
  - Previously
    - Use component technology management/monitoring API such as JMX monitoring, e.g. cache hit ratio, method invocation time
  - Now
    - OASIS Web Services Distributed Management (WSDM) compliant manageability endpoint : Performance Metrics
    - JSR 262: Web Services Connector for JavaTM Management Extensions (JMXTM) Agents

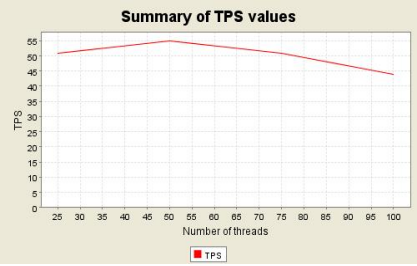
The imagination driving Australia's ICT future.

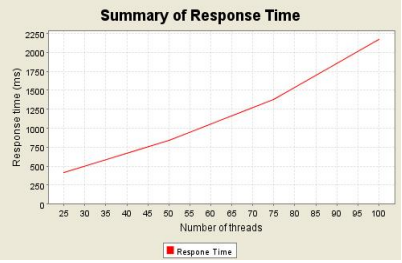
NATIONAL ICT AUSTRALIA

Automatically Generated Results









The imagination driving Australia's ICT future.

NATIONAL ICT AUSTRALIA

Conclusion

- The benchmark generation is compliant with MDA
  - Provide a domain specific modeling language for WS performance testing
  - Quick modeling and configuration of WS performance testing
  - First implementation of a tailored UML 2.0 Testing Profile
- Provide performance testing generation cartridges and monitoring infrastructure
  - Incorporate best practice of WS performance testing
  - A default implementation and test data generation saves a large amount of effort for normal load testing.
  - Reuse WS test data and test logic separately
- Save effort and improve quality by
  - plumbing and best practice integrated in code generation cartridges/framework
  - Working with latest technologies by tapping into AndroMDA's cartridge pool
  - Leveraging UML expertise and standards
- Part of a bigger picture of capacity planning research at NICTA

Limitations and Future Work

- Test data generation
  - Exception modelling
  - Integrate real world load profile
- Extend to other areas
  - Performance testing and monitoring for Web services coordination and choreography
  - Link with Business Activity Monitoring (BAM) and business process performance monitoring