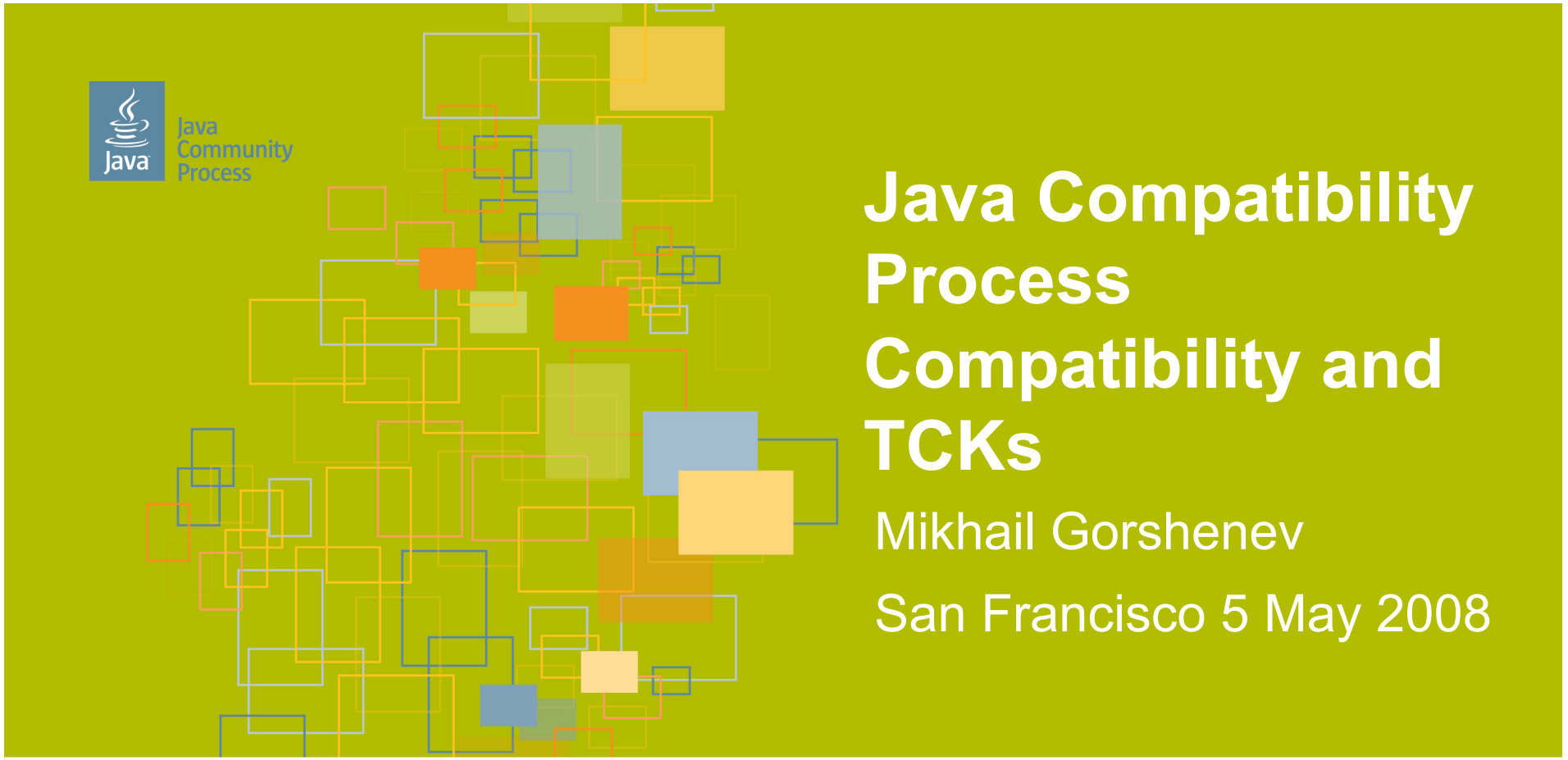
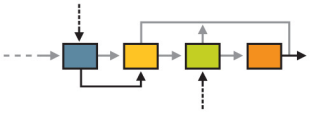


Java Compatibility Process Compatibility and TCKs

Mikhail Gorshenev

San Francisco 5 May 2008



Community Development of Java Technology Specifications

What is Compatibility?

- **Assurance that a Technology implementation is conformant to its specification**
- **Compatibility provides assurance:**
  - **Implementations match the specification**
  - **Implementations all meet (at least) a minimum level of quality**
  - **Specifications, rather than implementations, define programmatic behaviour**

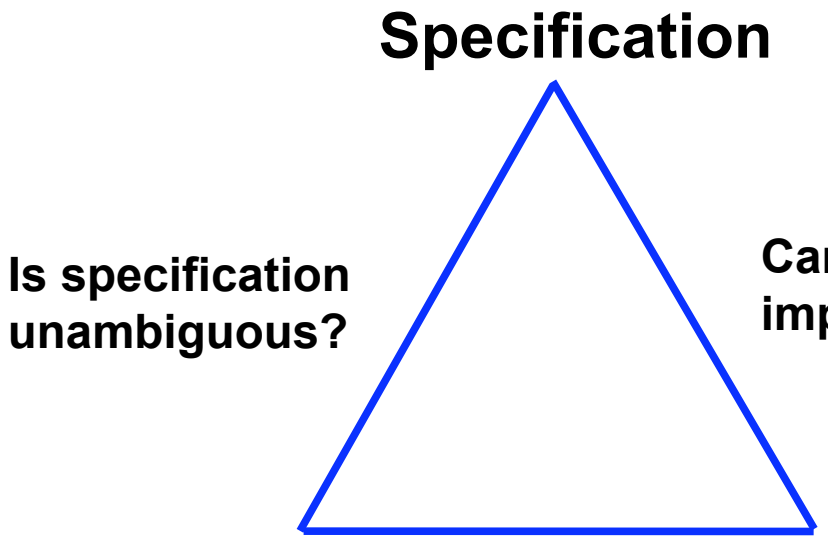
Why do we care about Compatibility?

- Promotes value in the Java™ brand
- Provides assurance of application portability
- Ensures a level playing field for all implementers
- Provides assurance to end users that all “required” (i.e. guaranteed) functionality is present
- Enables developers to write to a specification rather than to an implementation
- Improves (over time) the quality of the specification

What Defines a Java Technology in JCP?

- **Specification**
  - **Defines API requirements**
- **Reference Implementation (RI)**
  - **Proves the spec can be implemented**
- **Technology Compatibility Kit (TCK)**
  - **Ensures all implementations are conformant to the specification**

Why are all Three Parts Required?



Each supports the others during development

Specification

Java Community Process

- **A Specification should be complete and accurate**
- **Specifications have 3 audiences:**
  - **Developers**
  - **Licensees/Implementers**
  - **TCK developers**

Specification

- **Spec writing guidelines**
  - **Use javadoc**
  - **“Think assertions”**
  - **“Think testable”**
    - **Optional is bad (i.e., “may” rather than “must”)**
  - **RFC 2119**
  - **W3C QA and Conformance Working Group have published specification writing guidelines (http://www.w3.org/QA/WG/)**

What Is a Technology Compatibility Kit (TCK)?

Java Community Process

- **The suite of tests, tools and documentation that allows the implementer of a specification to determine if their implementation is compliant with that specification.**
- **Only compatible implementations (i.e., those that pass their TCKs) can be distributed to customers**
  - **Passing is more than just passing the tests**
  - **Compatibility requirements are defined by the TCK**

Sun microsystems

What Is a Technology Compatibility Kit (TCK)?

Java Community Process

- **Spec Lead responsibilities**
  - **Development**
  - **Licensing terms**
- **Maintenance Lead responsibilities**
  - **Appeals Process**
  - **TCK updates**

What Makes Up a Typical TCK?

- Test Suite
  - Tests
  - Test Framework
  - Exclude List
- Test Harness
- Documentation
  - TCK Users' Guide
    - How to Run the Test Suite
    - Compatibility Requirements
    - Appeals Process
  - Supporting Tools Documentation

Test Suite

- **Tests**
  - **Designed to demonstrate and verify implementation compliance to its specification**
- **Test framework**
  - **Defines the particular test environment used by the test harness to run the tests and collect test results**

Test Suite

- **Exclude List**
  - **Provides a means to remove invalid tests**
  - **Invalid tests are:**
    - **Test has bugs**
    - **Spec item covered by test is ambiguous**
    - **Test does not match spec**
    - **Test assumes unreasonable hardware/software requirements**
    - **Test is biased to a particular implementation**

Documentation

- **TCK Project Planning and Development Guide**
  - Rationale and examples for TCK development
  - TCK project scoping information
- **Java Technology Test Suite Development Guide**
  - Techniques for test development
- **TCK User's Guide template**
- **Example documentation for TCKs**
  - Compatibility requirements, appeals process, etc.
- **Latest versions available at**
  - http://java.sun.com/javame/reference/apis.jsp#testtools

Test Harness

- Tool to automate testing of an implementation for test:
  - Selection
  - Scheduling
  - Execution
  - Reporting
- Assures all required tests are run and pass
- Tests can easily be rerun for debugging purposes
- Manages testing process without introducing human error

Compatibility Requirements

- Defines how to “pass” the TCK
- Definitions and Rules listed in the TCK User's Guide identify:
  - Criteria for successfully passing the test suite
    - Example: All required tests must pass
  - Non-testable criteria that must be assured by the implementer. For example:
    - Tests cannot be changed by the implementer
    - Implementations cannot deviate from the specification where functionality isn’t explicitly tested

Compatibility Requirements

- Criteria for Signature Test
- These rules evolve over time and vary by Technology
  - Language standardized through use of templates

Appeals Process

- The process defined by the Spec Lead that allows implementers of the Specification to challenge one or more tests defined by the Specification's TCK
- The appeals process should identify:
  - Who can make challenges to the TCK
  - What challenges to the TCK may be submitted
  - How these challenges are submitted
  - How and by whom challenges are addressed
  - How accepted challenges to the TCK are managed

New TCK Requirements (for JCP 2.6)

- JCP 2.6 identifies new requirements for TCKs to better ensure commonality of coverage and quality of TCKs across all JSRs
- New requirements include:
  - Licensing terms provided to EC at each review
  - Documentation (for review by EC at Final Approval Ballot)
  - 100% Signature coverage for APIs
  - Test Harness to automate testing and reporting
  - TCK Coverage Document

TCK Coverage Document (new for JCP 2.6)

- **The TCK Coverage Document (TCD)**
  - An xecutive summary (one or two pages) included in FAB materials to assist the EC in evaluating the TCK
- **The TCD includes:**
  - Overview of the TCK documentation
  - Description of means used to validate TCK quality
  - Description of means used to validate TCK coverage
  - Test coverage numbers
  - Justification for the adequacy of the TCK
    - Quality and coverage

TCD Example (from Spec Lead Guide)

- **TCK Coverage Document for JSR-xxx:**
  - **Wombat API (Optional Package for Java SE)**

- **TCK Components**
  - **User's Guide**
    - **Compatibility Rules**
    - **Configuration instructions**
  - **Test Suite**
    - **API Tests**
    - **Signature Test**
    - **Framework code**
    - **Configuration Interview**

- **JavaTest™ Harness**

Terminology of Metrics

- Assertion: A specific statement of functionality or behavior required by a specification. A testable assertion is one that can be validated in an implementation by testing.
- Test: A binary application, data or script comprised of one or more Test Cases.
- Test Case: A single set of test inputs, execution conditions, and expected results developed to verify an implementation's conformance with a specific assertion.
- Assertion Breadth Coverage: Ratio of all assertions tested by at least one test case to the total number of testable assertions defined by the specification.
- Assertion Depth Coverage: For all assertions tested by test cases, the ratio of these test cases to the total number of test cases needed to completely test the assertions (assertions may require multiple test cases to adequately verify conformance to the specification).
- Method Coverage: Ratio of methods directly exercised by test cases to the total number of methods defined by the specification.

TCD Example (continued)

Coverage

- Total testable assertions: 426 as identified through the use of the Java CTT Spec Trac t o ol and hand markup (hand markup was required for the two referenced specifications)
- Tests: 304 tests (comprising 627 test cases)
- Assertion Breadth Coverage : 70% as measured by Spec Trac and 65% estimated for both referenced specifications
- Assertion Depth Coverage: Approx. 50% (estimated by sampling)
- Assertion Breadth Coverage varies by package and ranges from 30% to 95%
- A ss ertion Breadth Coverage of the Wombat data exchange protocol is estimated to be 90%
- Method Coverage (static): 94% as measured by the Java CTT API Coverage Tool
- Me thod Coverage (runtime against the RI): 97% as measured by XYZ code coverage utility

TCD Example (continued)

Quality Assurance

- TCK was run using representative configurations of the Reference Implementation on the following platforms:
  - Red Hat Linux/Sun Java SE v1.3.1 and 1.4.2
  - Windows 98, 2000, XP/Sun Java SE v1.3.1, 1.4.2
  - Solaris 9/Sun Java SE 1.4.1
  - Mac OS 10.2.6/Apple Java SE v1.4.2
- Code quality was demonstrated through use of code reviews and inspections
- User's Guide was constructed from the Java CTT TCK User's Guide template
- Documentation instructions and Configuration Interviews were exhaustively verified and tested

TCD Example (continued)

Justification of Adequacy

- As the first major release of this specification, the coverage is good and consistent with the coverage of similar JSRs at this level of spec maturity. Coverage variance is due to effects of late spec changes and the availability/cost of test development resources. Emphasis was given to covering critical spec areas as identified by the Spec Lead and Expert Group. We plan to increase coverage in weak areas during the next major specification update release.

Scoping a TCK Project

- TCKs are not trivial to produce, so **Begin Early!**
- Define the level of assertion coverage desired and plan adequate resources and time
- Creating a good TCK will require about the same amount of effort and time as the RI
- Take advantage of the Spec-RI-TCK triangle

Test Development

- **Several levels of conformance testing coverage:**
  - **Signature - checks completeness**
  - **Method - checks basic functionality**
  - **Assertion - checks required behaviour**
- **Must have 100% signature coverage**
- **Strive for 100% method coverage**
- **Goal should always be to provide as much assertion coverage as possible**

Assertion

- Statement which specifies some necessary aspect of the API
- Assertion attributes
  - testable
  - required
  - not an assertion (e.g., example text)
  - optional
  - implementation-specific
- Example 1: java.lang.Integer.toString(int i, int radix) method description
  - “If the radix is smaller than Character.MIN_RADIX or larger than Character.MAX_RADIX, then the radix 10 is used instead.”
- Example 2: java.lang.Runtime.exit(int Status) method description
  - “Terminates the currently running Java Virtual Machine. This method never returns normally”.

Assertion Coverage

- Percentage of testable assertions covered by the TCK tests
- Industry expects 100% assertion coverage for mature specifications
- Need perfect spec and RI to achieve 100%
- 75% assertion coverage is a good practical target for most new TCKs
- Very unlikely to achieve even 20% if TCK development is started too late

What Materials does Sun Provide?

- **In May 2001, Sun released "Java Compatibility Technology Tools" for use within JCP:**
  - **Java Compatibility Test Tools (under license)**
    - **JavaTestTM Harness**
    - **Spec Trac, Sig Test, API Cov**
  - **Documentation**
    - **TCK Project Planning and Development Guide**
    - **Java Technology Test Suite Development Guide**
    - **TCK User's Guide template**
    - **User Guide's for all the tools**
  - **See http://jcp.org/resources/tdk**

Java ME TCK Framework

- **Set of JavaTest plug-ins for the Java ME platform**
  - Support for Java ME profiles and configurations (CLDC, CDC, FP/ PBP/PP, MIDP)
  - Agents for Java ME application models (main, Xlet, MIDlet, Applet)
  - Test deployment infrastructure (application provisioning)
- **Redistributable binary available at no cost**
  - http://java.sun.com/javame/reference/apis.jsp#testtools
- **Based on open-source ME Framework software**
  - https://meframework.dev.java.net/
- **Java ME test examples**
- **Developer's Guide explains Java ME-specific aspects of** test suite creation

Documentation

- **TCK Project Planning and Development Guide**
  - **Rationale and examples for TCK development**
  - **TCK project scoping information**
- **Java Technology Test Suite Development Guide**
  - **Techniques for test development**
- **TCK User's Guide template**
- **Example documentation for TCKs**
  - **Compatibility requirements, appeals process, etc.**

JavaTest

Java Community Process

- **Sun's test harness for testing Java API compatibility**
- **Based on open-source JTHarness software**
  - https://jtharness.dev.java.net/
- **Flexible test sequencing and reporting tool for all kinds** of Java API compatibility testing
- **Configurable to most product testing environments and** devices
- **Binary is redistributable at no cost for JCP TCKs**
- **Java Test Architect's Guide explains configuring test** suites to operate under JavaTest

Sun microsystems

Spec Trac

- Automatic way of tracking assertions in a specification
- Calculates coverage for a package or API
  - Number of assertions tested
  - Percentage of assertions tested
- Highlights differences between revisions of a specification using the GUI
- Tracks tests which need change as a result of a revised specification
- GUI interface – allows for manual markup of an assertion and classification of that assertion

Sig Test

Java Community Process

- Tool to create a signature tests
- Open-source development
  - https://sigtest.dev.java.net/
- Verifies correctness of API signature
  - Members and attributes of the API
  - Equality of API member sets

API Coverage

- **Tool to measure the member coverage of a TCK**
  - **Much less precise than assertion level coverage**
  - **Shows breadth but not correctness of coverage**
- **Calculates percentage of methods, fields and constructors defined by the specification that are directly invoked by a TCK**
  - **Uses static analysis to determine coverage**
  - **95+% should be target for most TCKs**

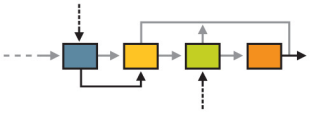
Training Course

- **Key parts of the TCK Development Guidelines and the Test Suite Development Guide have been captured in a web-based training course.**
- **If interested (it's free), please send mail to:**

  tsdg_course_beta@sun.com

Summary

- TCKs are a valuable tool for developme nt of specification and reference implementations
- Passing a TCK requires more than just passing the tests (i.e., pas sing is defined by the compatibility requirements)
- Plan your TCK project and begin te st development early to ensure a strong test suite
- Strive for high test assertion coverage
- Take advantage of Sun's test development tools



Community Development of Java Technology Specifications