The NSFC Key Research Plan: Trustworthy Software

He Jifeng
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
October, 2009
Background

- Infrastructure
- Economical drivers
- Competitiveness
Motivation

- Improve software trustworthiness
- Encourage invocation within software industry
- Re-energize software research and development
- Prepare students for careers in software sector
Challenges

- Gap between the mathematical world and the physical world

- Pursue solutions for the undecidable problems

- Unpredictability of open environment

- Additional complexity due to interaction, sharing and coordination
Goals

- Exploit and assess the fundamental laws about the trustworthiness, metric and evolution of software
- Provide design, verification, evolution and control methods which should enable new forms of analysis, testing, and validation of integrated systems
- Build development tools and running support environment which should be used to produce and verify safety-critical software
About NSFC Key Project

- **Name:** The fundamental research on trustworthy software
- **Launched by NSFC**
- **Budget:** 150 million RMB
- **Term:** 2007-2014
- **On-going projects:** 70 projects has been supported
Themes

- Modeling and measuring trustworthiness
- Design and verification of trustworthy software
- Software evolution and control
- Development and evaluation of the trustworthy substrates

Covered Fields:

- Theorem proving
- Object calculus and MDA
- Web transaction
- Model checking
- Network security
- Software evolution
- Software process
- Requirement analysis
- Software testing
- Trustworthy database
- Software metrics
- Compilation
- Symbolic computing
- ...

Microsoft Research Asia Verified Software Workshop 2009
Modeling and measuring trustworthiness

- How to comprehend understand the trustworthiness of software
- How to describe and reason about the trustworthiness
- How to assess and control the trustworthiness
Design and verification of trustworthy software

- How to accommodate the computation laws in an open physical environment to ensure the trustworthiness
- How to design the trustworthiness
- How to resolve the conflicts among the various attributes of the trustworthiness
Software evolution and control

- How to adopt the active safeguard mechanism for the trustworthiness
- How to acquaint and recognize the evolution of software and its running environment
- How to acquire the trustworthiness dynamically and control its variation
Development and evaluation of the trustworthy substrates

- How to design the proper architecture for the trustworthy environments
- How to set up and manage the trustworthiness chains.
- How to construct secure running environments in support of coordination, cooperation and autonomy in networks of trustworthy systems
- How to evaluate the trustworthiness of the running environments
Progress

- Foundation
- Methods and tools
- Applications
Foundation

- The analysis, verification and tool support for embedded systems based on computer algebra
- Theories and algorithms of real number trustworthy Computation
- Formal approach to web transaction
Analysis, verification and tool support for embedded systems based on computer algebra

- Analysis of program termination
  - The termination problem has been explored for a class of programs with linear constraints and linear assignments
  - The rank function generation technique is proposed to search the counterexample to discover the non-termination of nonlinear programs
Invariant generation

- An invariant generation technique is proposed for polynomial programs by semi-algebraic system (SAS) solving.

Tool development of symbolic computation

- Investigators have their own verification tool based on computer algebra: DISCOVERER and BOTTEMA.
- For hybrid system, the fast determination approach to the satisfiability of polynomial constraints can be proceeded by DISCOVERER tool.
- The newest version of **Maple 13** has included DISCOVERER as a standard package.
Theories and algorithms of real number trustworthy computation

- Theory and Implementation of trustworthy interval computation
- Error control of the floating point computation based on the accurate floating point algorithms
- Real number representation and computational method based on Sierpinski triangle coordinat
- Accelerated algorithms of the accurate floating point computation and accurate real number computation
Features of Web transaction:
- Compensation
- Fault handling
- Long-running computation

Various semantic frameworks for Web transaction
- Approach: UTP
- Extension: time and probability
The linking between various semantics
- Denational semantics, operational Semantics
- Algebraic Semantics
- Normal form approach

Verification for Web transaction
- Model checking and assertion method
- Detailed execution states need to be explored.
- Asynchronization and coordination for Web transaction
Methods and tools

- Pattern-based approach to prediction and metrics for source code
- Language and logic for proof-carrying program
- Program verification by the theory of differential equations
- Trustworthy computing framework based on virtual machine architecture
Pattern-based approach to prediction and metrics for source code

- Mining usage patterns from software library
  - Usage predictions from run time execution
  - Usage predictions from software library documents
  - Usage predictions from source code
Pattern-based approach to prediction and metrics for source code

- Java program metrics tool
- Class structure, McCabe value, inheritance depth...
- Static analysis techniques are applied

<table>
<thead>
<tr>
<th>Metric</th>
<th>Total</th>
<th>Mean</th>
<th>Maximum</th>
<th>Description</th>
<th>Resource causing Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth of Inheritance (avg/max per type)</td>
<td>1.811</td>
<td>6</td>
<td>6</td>
<td>Metric the number of level of inheritance</td>
<td>/PKUMetrics Plugin/src/metrics/ui/MetricsSR</td>
</tr>
<tr>
<td>Number of Subclasses (avg/max per type)</td>
<td>0.946</td>
<td>8</td>
<td>8</td>
<td>Metric the number of subclasses that extends the given...</td>
<td>/PKUMetrics Plugin/src/metrics/analyzer/MainWindow</td>
</tr>
<tr>
<td>Line Of Code (avg/max per compilationUnit)</td>
<td>2032</td>
<td>56.444</td>
<td>226</td>
<td>Metric the line length of pure code</td>
<td>/PKUMetrics Plugin/src/metrics/ui/MetricsSR</td>
</tr>
<tr>
<td>src</td>
<td>2032</td>
<td>56.444</td>
<td>226</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activator.java</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics.analyzer</td>
<td>1030</td>
<td>54.211</td>
<td>124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics.builder</td>
<td>54</td>
<td>27</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics.builder</td>
<td>3</td>
<td>1.5</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics.source</td>
<td>541</td>
<td>60.111</td>
<td>116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>metrics.ui</td>
<td>374</td>
<td>124.667</td>
<td>226</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line of code &amp; comment (avg/max per compilationUnit)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Metric the line length of code and comments</td>
<td>/PKUMetrics Plugin/src/metrics/ui/MetricsUL</td>
</tr>
<tr>
<td>Gross line of code (avg/max per compilationUnit)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Metric the gross line of length</td>
<td>/PKUMetrics Plugin/src/metrics/analyzer/acrossMethod</td>
</tr>
<tr>
<td>McCabe (avg/max per method)</td>
<td>1.781</td>
<td>12</td>
<td>McCabe</td>
<td></td>
<td>/PKUMetrics Plugin/src/metrics/analyzer/acrossMethod</td>
</tr>
<tr>
<td>Number of Parameters (avg/max per method)</td>
<td>1.046</td>
<td>4</td>
<td></td>
<td>Metric the number of parameters of the method</td>
<td>/PKUMetrics Plugin/src/metrics/analyzer/acrossMethod</td>
</tr>
</tbody>
</table>
Language and Logic for proof-carrying program

- Verification of operating system kernel
  - Reasoning about the interruption mechanism on an abstract machine by concurrent separation logic
  - Verification of the scheduling for preemptive threads by defining the operational semantics of block and unblock operations.

- Design of a pointer logic
Language and Logic for proof-carrying program

- Verification of miniOS operating system
  - written in assembly language
  - multi-threads
  - Coq: 8, 2000 lines Coq program
  - developing the domain specific logic

- Pointer logic implementation
  - Algorithms of verifying linear program
  - Automatic proof of verification conditions
Program verification by the theory of differential equations

- Modeling concurrent systems by differential equations
  - Discrete Petri-net for specifying systems
  - Transforming Discrete Petri-net into continuous one
  - Transforming continuous Petri-net into a set of differential equations
- Deadlock analysis for differential equations
  - Channel-based model
  - Shared-resource model
Program verification by the theory of differential equations

- Experiments for philosopher problem
- Compared with other model checkers: Spin, PAT, and TOTAL
- The number of philosophers is from 5 to 400.
- This approach is more efficient than others in performance.
Applications

- Trustworthy Embedded systems:
  - Lunar Probe Satellite #1
  - Remote Control System for Opening Ceremony of 2008 Olympic Games
  - Underground Traffic Control System

- Web Security
  - Networks of Financial Systems
  - Sensor Networks
Remote Control System for Opening Ceremony of 2008 Olympic Games

- Analysis, simulation and verification

Development Process of Trustworthy PLC Software
First-order prover with interpolation computation
Underground Traffic Control System

- Rigorous development method

**Diagram:**
- Braking Equipment (BE)
- Mode Direction Handle (MDH)
- ATP Processing Unit (PU)
- Speed Sensor (SS)
- Train Operation Display (TOD)
- Train Information Management (TIM)
- Trainlines (Vehicle Interface)

Automatic Train Control System

Microsoft Research Asia Verified Software Workshop 2009
Underground Traffic Control System

- Modeling, refinement, verification and automated code generation
- Formal methods integration
Discussion......
Thank you!