TIDE — A Generic Debugging Framework
Tool Demonstration

Mark van den Brand, Bas Cornelissen, Pieter Olivier, Jurgen Vinju

Centrum voor Wiskunde en Informatica (CWI)

April 3, 2005
Integrated Development Environments

- Generic IDEs
  - Generating an IDE from high level descriptions
  - Domain specific languages, language prototyping
  - Eclipse, ASF+SDF, Meta-Environment, SmartTools

TIDE is a generic framework for debugging.
Integrated Development Environments

- Generic IDEs
  - Generating an IDE from high level descriptions
  - Domain specific languages, language prototyping
  - Eclipse, ASF+SDF Meta-Environment, SmartTools

- A Generic IDE should be as full featured as any IDE:
  - Syntax directed editors, highlighting, hyperlinks, outlines
  - Typecheckers, compilers, interpreters, debuggers, refactorings
Integrated Development Environments

- Generic IDEs
  - Generating an IDE from high level descriptions
  - Domain specific languages, language prototyping
  - Eclipse, Asf+Sdf Meta-Environment, SmartTools

- A Generic IDE should be as full featured as any IDE:
  - Syntax directed editors, highlighting, hyperlinks, outlines
  - Typecheckers, compilers, interpreters, debuggers, refactorings

- A Generic IDE consists of:
  - Generic components ................. language independent
  - Component generators ................. language parametric
  - Other components ..................... language specific
**Integrated Development Environments**

- **Generic IDEs**
  - Generating an IDE from high level descriptions
  - Domain specific languages, language prototyping
  - Eclipse, Asf+Sdf Meta-Environment, SmartTools

- **A Generic IDE should be as full featured as any IDE:**
  - Syntax directed editors, highlighting, hyperlinks, outlines
  - Typecheckers, compilers, interpreters, debuggers, refactorings

- **A Generic IDE consists of:**
  - Generic components ................. language independent
  - Component generators ............... language parametric
  - Other components ................... language specific

- **A debugger** is an important component of every IDE
Integrated Development Environments

- Generic IDEs
  - Generating an IDE from high level descriptions
  - Domain specific languages, language prototyping
  - Eclipse, ASF+SDF Meta-Environment, SmartTools

- A Generic IDE should be as full featured as any IDE:
  - Syntax directed editors, highlighting, hyperlinks, outlines
  - Typecheckers, compilers, interpreters, debuggers, refactorings

- A Generic IDE consists of:
  - Generic components ...................... language independent
  - Component generators ................. language parametric
  - Other components ..................... language specific

- A debugger is an important component of every IDE

- TIDE is a generic framework for debugging
What is a debugger?

- A user-interface for analyzing dynamic behavior of programs
  - Visualization of control flow and data
  - Offers control over the dynamic execution
What is a debugger?

- A user-interface for analyzing dynamic behavior of programs
  - Visualization of control flow and data
  - Offers control over the dynamic execution
- Typical features:
  - source correspondence,
  - stepwise execution,
  - conditional breakpoints,
  - inspecting data: variables, call stacks, ...
What is a debugger?

- A user-interface for analyzing dynamic behavior of programs
  - Visualization of control flow and data
  - Offers control over the dynamic execution
- Typical features:
  - source correspondence,
  - stepwise execution,
  - conditional breakpoints,
  - inspecting data: variables, call stacks, . . .
- A debugger is smart:
  - Understands the runtime environment of a program
  - Understands the relationship between source and object code
What is a generic framework for debugging?

- Rapid debugger development
  - Generic user interface
  - Common debugging protocol
  - Reusable debugging components

What is generic about it:
- Heterogeneous languages
- Runtime neutral interpreters, compilers, VMs

What it is not:
- A debugger generator
- Optimized for a specific language
What is a generic framework for debugging?

- Rapid debugger development
  - Generic user interface
  - Common debugging protocol
  - Reusable debugging components

- What is generic about it:
  - Heterogeneous ........................ language independent
  - Distributed .......................... remote programs
  - Multi-threaded ........................ parallel programs
  - Runtime neutral ..................... interpreters, compilers, VMs
What is a generic framework for debugging?

- Rapid debugger development
  - Generic user interface
  - Common debugging protocol
  - Reusable debugging components

What is generic about it:
- Heterogeneous ....................... language independent
- Distributed ............................. remote programs
- Multi-threaded ........................ parallel programs
- Runtime neutral ......................... interpreters, compilers, VMs

What it is not:
- A debugger generator
- Optimized for a specific language
What is TIDE?

- Extensible user interface
What is TIDE?

- Extensible user interface
- Common debugging protocol
  - Processes ............... running programs, threads
  - Ports ..................... states of processes
  - Events ........ events that happen at certain ports
  - Actions ...................... event handlers
  - Rules .............. which actions at which events
What is TIDE?

- Extensible user interface
- Common debugging protocol
  - Processes ................. running programs, threads
  - Ports ..................... states of processes
  - Events ........ events that happen at certain ports
  - Actions ...................... event handlers
  - Rules ..................... which actions at which events
- TIDE adapters
  - Extend a runtime using the TIDE API
  - Map runtime information to events
  - Map actions to runtime behavior
Demonstration

- Existing TIDE adapters:
  - ASF+SDF, ToolBus ......................... written in C
  - gdb, jdb .................................. written in Java
  - Pico ...................................... written in ASF+SDF

- Demonstration outline:
  - Debugging an ASF+SDF specification
  - Debugging a Pico program
  - Multi-level debugging ASF+SDF and Pico
What does it take to create a TIDE adapter?

- Assumption 1: runtime can link with C or Java libraries.
  - Or create your own TIDE API implementation.
What does it take to create a TIDE adapter?

- Assumption 1: runtime can link with C or Java libraries.
  - Or create your own TIDE API implementation
- Assumption 2: runtime has “source correspondence” info:
  - Maps current point(s) of execution to source location(s)
  - Maps values in memory to source level names
What does it take to create a TIDE adapter?

- Assumption 1: runtime can link with C or Java libraries.
  - Or create your own TIDE API implementation
- Assumption 2: runtime has “source correspondence” info:
  - Maps current point(s) of execution to source location(s)
  - Maps values in memory to source level names
- Identify **logical breakpoints** in the runtime
  - E.g. before or after each “statement”
  - Insert a call to a step function at logical breakpoints
What does it take to create a TIDE adapter?

- Assumption 1: runtime can link with C or Java libraries.
  - Or create your own TIDE API implementation
- Assumption 2: runtime has “source correspondence” info:
  - Maps current point(s) of execution to source location(s)
  - Maps values in memory to source level names
- Identify **logical breakpoints** in the runtime
  - E.g. before or after each “statement”
  - Insert a call to a step function at logical breakpoints
- Implement **init**, **step** and **actions**:
  - init ................................. connect to TIDE, start process
  - step ...... synchronize with TIDE, send events, receive actions
  - actions ............................... implement abstract actions
What does it take to create a TIDE adapter?

- Assumption 1: runtime can link with C or Java libraries.
  - Or create your own TIDE API implementation
- Assumption 2: runtime has “source correspondence” info:
  - Maps current point(s) of execution to source location(s)
  - Maps values in memory to source level names
- Identify logical breakpoints in the runtime
  - E.g. before or after each “statement”
  - Insert a call to a step function at logical breakpoints
- Implement init, step and actions:
  - init ............................. connect to TIDE, start process
  - step ..... synchronize with TIDE, send events, receive actions
  - actions ............................ implement abstract actions
- Link the TIDE API and the new adapter with the program
TIDE provides a generic framework for debugging
http://www.cwi.nl/projects/MetaEnv/tide

Questions?