Supporting dynamic software tool integration via web service-based components

Nelson Yap\textsuperscript{1}, Hau Chean Chiong\textsuperscript{1}, John Grundy\textsuperscript{1,2} and Rebecca Berrigan\textsuperscript{2}

\textsuperscript{1}Department of Electrical and Computer Engineering  
\textsuperscript{2}Department of Computer Science  
University of Auckland, New Zealand
Outline

- Motivation: the tool integration/extension problem
- Our approach: web service-based “toolets”
- Example of usage
- Architecture
- Evaluation
- Conclusions and Future Research
Software Tool Integration and Extension

• Often a desire or even a need to integrate various software tools within an organisation

• Examples:
  - UML design tool -> code for Java IDE -> testing tool suite -> version control -> documentation tool

• Motivations:
  - Diversity of developer preferences - what aspects of an IDE to use vs more easily choosing alternatives
  - Opportunistic vs. Mandated Practice in organisation
  - Influence of software process on tool usage
  - Licensing issues for tools/tool extensions
  - Desire for “best of breed” facilities
Example: JEdit Plug-ins

Plugins list

ASWEC 2005 Presentation (c) John Grundy
Realisation: JEdit plug-in approach

JEdit IDE
- Plug-in Manager
- CVS Plug-in
- JLint Plug-in
- JRefactor Plug-in
- IMS Plug-in
- Rose IF Plug-in
- mySQL Plug-in

JEdit Host PC
- Local JLint, JRefactor Tools
- Local Rational Rose™ CASE tool

Plug-in IF

Remote JEdit Plug-in Repository
- Remote CVS Server
- Instant Messaging Server
- MySQL Server Database

Local Rational Rose™ CASE tool
Current Approaches

- **Data integration (loosely coupled):**
  - Shared file system/databases/version management tools
  - Exchange formats e.g. XMI
- **Control integration (tightly coupled):**
  - Typically API-based integration; plug-in managers
- **User interface integration (loose to tight coupling):**
  - Close integration of tool user interfaces e.g. HP SoftBench (wrapper) to plug-in approaches (shared)
- **Process integration:**
  - Various process-centred IDEs controlling when/how tools used
Our Approach: JEdit-WS

1. Develop remote WS comps for “toolets”

2. Advertise available toolet WS comps in UDDI registry

3. JEdit-WS client issues query to registry

4. Select toolet WS comps want to make use of

5. Integrate required toolet comps and UIs with JEdit-WS

6. Invoke Remote toolet comp via WS interface
Example: Finding Toilets
Refactoring

```java
java runtime = c:\jdk\jre\lib\rt.jar

classpath =

sourcepath = E:\WebServices\ThreadWorkingSpace\Thread-2-WorkingSpace\Java\EventPanel.java

java version = 1.3

b. WebServices\ThreadWorkingSpace\Thread-2-WorkingSpace\FindByPanel.java

b. WebServices\ThreadWorkingSpace\Thread-2-WorkingSpace\ConceptsTree.java

b. WebServices\ThreadWorkingSpace\Thread-2-WorkingSpace\InputPanel.java
```
Code Lint
CVS
Operation of JEdit-WS

JEdit IDE

- Plug-in Manager
- WS Toool UI Panel
  - Toool files parameters
  - Toool results
- JEdit File Editor API
- JEdit File Manager API
- Local Files

WS Toool Manager
- JRef. Client
- CVS Client
- JLint Client

UDDI Registry

JRefactor Toolset Service

JRefactor Program

Local Files

Toolet files & parameters

Toolet results

(1) (2) (3) (4) (5) (6) (7) (8) (9)

(previously registered)
Web service prototype

Client: jEdit

Web Service User Interface

Validate and serialize input

Deserialize Output and display

Web Service invocation

Server

Scheduler

Waiting queue

Deserialize and process Input parameters

Invoke Services

Code Lint
JRefactor
IM
CVS

Process and serialize output

SOAP
HTTP

ASWEC 2005 Presentation (c) John Grundy 13
Discovery Enhancement via “Aspects”

• Began extending the web service description language (WSDL) to better-characterise remote tool service characteristics

• Includes better support in tool service description for:
  - User interface(s) to add to client tool
  - Transport mechanism, security protocols etc to use
  - Data management approaches – local, remote, caching policy etc
  - Other service resource utilisation issues

• Using aspect-oriented techniques to achieve – characterise service, discover service, integrate service
Evaluation

- Group of experienced Java developers, familiar with the kinds of toolset services available to JEdit-WS
- Tasks involved discovering appropriate services; integrating into JEdit-WS; using on programming task
- Feedback on discovery and integration very positive
- Some issues with user interfaces of integrated tools; some issues with remote tool invocation/reporting of results of processing
- Some concerns with performance for toolsets of certain classes
- Difficulty of building toolsets moderate (requires detailed knowledge of web services but not JEdit)
Conclusions and Future Research

- Web services provide reasonable service-based infrastructure for extending IDEs in certain ways
- Suit toolset services that don’t have high user response requirement
- Allows choice of toolset services, sharing of toolsets, reduces some upgrade and consistency issues
- Need to better characterise services and integrate new user interfaces better into IDE client
- Data management, communication, security issues need further research work
- Need support for developing and testing toolsets
References