jETI: A Tool for Remote Tool Integration

Tiziana Margaria[1], Ralf Nagel[2], and Bernhard Steffen[2]

[1] Service Engineering for Distributed Systems,
Institute for Informatics, University of Göttingen, Germany
margaria@cs.uni-goettingen.de
[2] Chair of Programming Systems, University of Dortmund, Germany
Ralf.Nagel@udo.edu, steffen@cs.uni-dortmund.de

Abstract. We present jETI, a redesign of the Electronic Tools Integration platform (ETI), that addresses the major issues and concerns accumulated over seven years of experience with tool providers, tool users and students. Most important was here the reduction of the effort for integrating and updating tools. jETI combines Eclipse with Web Services functionality in order to provide (1) lightweight remote component (tool) integration, (2) distributed component (tool) libraries, (3) a graphical coordination environment, and (4) a distributed execution environment. These features will be illustrated in the course of building and executing remote heterogeneous tool sequences.

1 Motivation

The Electronic Tool Integration platform (ETI) [10] is an online platform specifically designed to support the distributed use of and experimentation with tools over the internet. Born in 1996 and online since early 1997, it offered a unique service to tool providers and users: its solution for the remote execution of tools and the internet-based administration of user-specific home areas on the ETI server was well ahead of the technology of those times. Since then, users can

- *retrieve information* about the tools,
- *execute tools* in *stand-alone mode*, or
- *combine functionalities of different tools* to obtain sequential programs called *coordination sequences* and run them in *tool-coordination mode*.

In particular, ETI is unique in allowing users to combine functionalities of tools of different application domains to solve problems a single tool never would be able to tackle.

Obviously, the richness of the *tool repository* plays a crucial role in the success of ETI: the benefit gained from our experimentation and coordination facilities grows with the amount and variety of integrated software-tools. The success of the ETI concept is thus highly sensitive to the process and costs of *tool integration* and *tool maintenance*.

In this paper we show how, taking advantage of newer technologies that internally base on Web Services and Java technology, we can

N. Halbwachs and L. Zuck (Eds.): TACAS 2005, LNCS 3440, pp. 557–562, 2005.

1. considerably simplify the integration process, and at the same time
2. flexibilize the distribution, version management and use of integrated tools,
3. broaden the scope of potential user profiles and roles, by seamlessly integrating ETI's coordination and synthesis features (cf. [8]) with a standard Java development environment, and
4. solve the scalability problem connected with tool maintenance and evolution.

The background and a first attempt to the new distributed way of tool integration for ETI have been described in [3]. Our current version of ETI, jETI,

- exploits Web Services technology [14, 13, 11] to further simplify the remote tool integration and execution,
- supports cross platform execution of the coordination models based on the quasi standard set by Java, and it naturally
- flexibilizes the original coordination level by seamlessly integrating the Eclipse development framework [2].

A more detailed description of jETI can be found in [4].

In the following, Section 2 sketches ETI's philosophy of remote tool integration, before we describe ETI's enhanced, formal methods-based coordination facility in Section 3, and ETI's framework view in Section 4. Our conclusions and directions are given in Section 5.

2 jETI as an Integration Tool

jETI's integration philosophy addresses the major obstacle for a wider adoption, as identified during seven years of experience with tool providers, tool users and students: the difficulty to provide the latest versions of the state-of-the-art tools. The tool integration process required on dedicated ETI servers was too complicated for both the tool providers and the ETI team, making it impossible to keep pace with the development of new versions and a wealth of new tools. jETI's new remote integration philosophy overcomes this problems, because it replaces the requirement of 'physical' tool integration by a very simple registration and publishing. This allows the provision of tool functionalities in a matter of minutes: fast enough to be fully demonstrated during our presentation. Moreover, whenever the portion of a tool's API which is relevant for a new version of a functionality remains unchanged, version updating is fully automatic!

Based on the Web Services functionality, the realization of this registration/publishing based integration philosophy required the implementation of four components, as illustrated in Figure 1:

1. a **HTML Tool Configurator**, which allows tool providers to register a new tool functionality just by filling our a simple template form,
2. the **jABC Component Server**, which (a) automatically generates appropriate Java classes from these specifications and (b) organizes all the registered tool functionalities, including the corresponding version control,

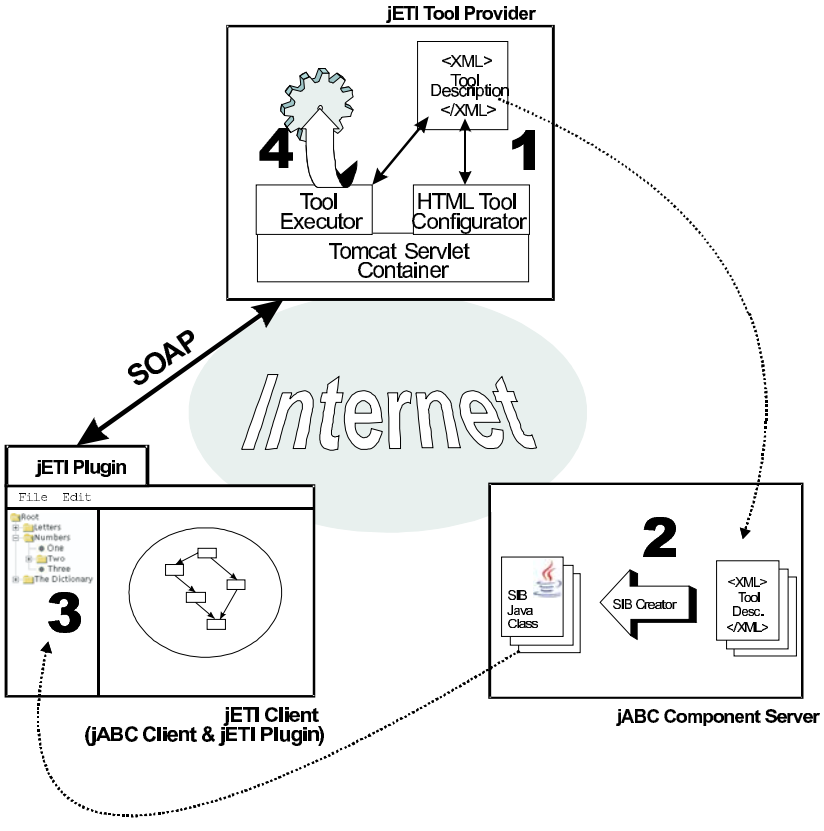

Fig. 1. The jETI Tool Integration Workflow

3. the **jETI client**, which automatically loads the relevant Java classes from the jABC Component Server and provides a flexible Java development environment for coordinating the so obtained tool functionalities. Depending on their goals and skill profile, users may just use our graphical coordination editor (as in Sect. 3) to experiment with the tools, or they may use the full development support of Eclipse to really embed remote functionalities into normal Java programs. Of course, this choice heavily influences the size of the required jETI-Client: only the first option is open to our envisaged 'pure HTML' solution.
4. a **Tool Executor**, which is able to steer the execution of the specified tools at the tool providers' site.

This approach enables *experts* to develop complex tools in Java on the basis of a library of remotely accessible tool functionalities, as well as *newcomers* to use jETI's formal methods-based, graphical coordination environment to safely combine adequate tool functionalities into heterogeneous tools.

3 Formal Methods in jETI: The Coordination Feature

jETI's philosophy of 'pure Java' totally eliminates the need for the proprietary high-level coordination language (HL) of the original ETI [8], as well as the proprietary format of the SIBs, ETI's elementary building blocks. In jETI, SIBs are now just classes that support a certain interface, which directly allows arbitrary tool coordination via Java programming, possibly supported by Eclipse [2] or other IDEs.

As Java programming-based coordination is only open to programming experts, jETI additionally provides a formal methods based, graphical coordination environment. This environment allows non-experts to graphically compose arbitrary tool functionalities under the control of a *type checker*, a *model checker* and a *model synthesis* tool, as shown in Figure 2 top right. Coordination models passing this control are directly remotely executable on every (distributed) platform providing a Java Virtual machine and a Tomcat Servlet Container.

However, not only the tool composition is under formal methods control. All the tool functionalities are taxonomically characterized by means of *ontologies*,

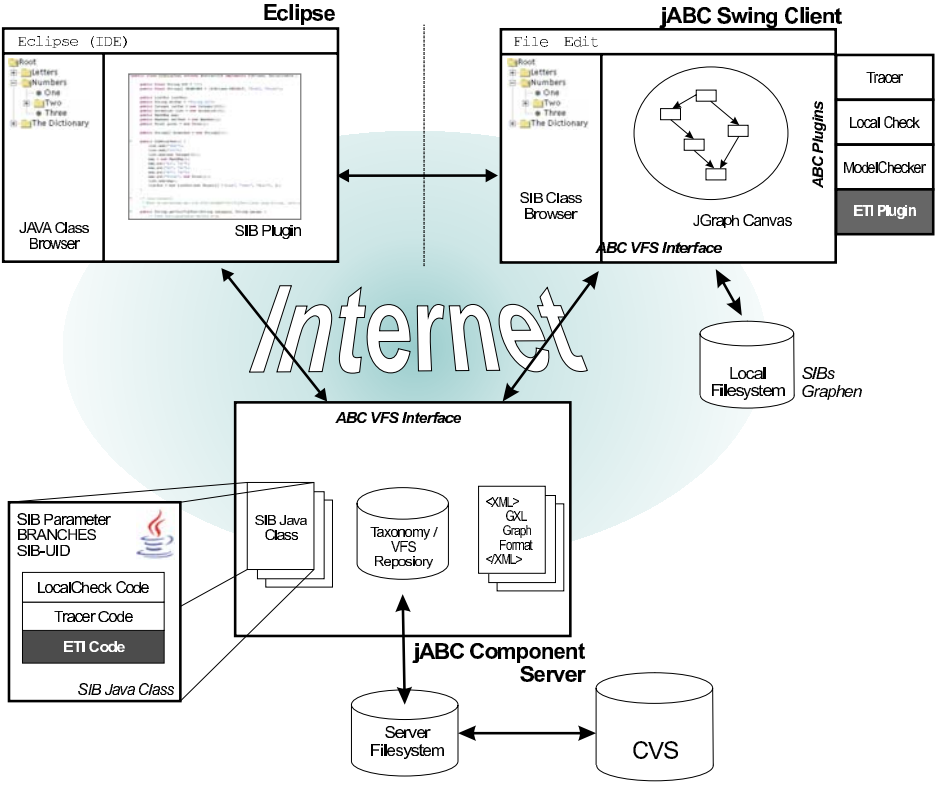

Fig. 2. Architecture of the jABC Framework

similar to the techniques adopted for the Semantic Web [1]. ETI supports a global classification, but users may also introduce their private classification scheme, which helps them to quickly identify the tools relevant for certain applications. In fact, the requirement for this organizational support of tool functionalities was a result of a common project with the CMU, aiming at introducing a larger variety of formal verification tools in the undergraduate curricula.

4 jETI: The Architecture

jETI can be seen as a tool that enhances other tools and frameworks by the integration, organization and execution of remote functionalities. E.g., the setup described above is based on jABC (cf. Figure 2), which is itself a framework for enhancing Java development environments (like Eclipse) with a graphical coordination level and dedicated control via formal methods. The charme of this architecture is that complex environment functionality can be added just via the plugin concept: this allows users to combine/exchange functionality in a transparent way, without touching the code of the kernel system. In our case, jABC can itself be seen as an Eclipse plugin, which, in addition to the ETI plugin, offers plugins for model checking, local checking, and a tracer. Dually, Eclipse can also be seen as an jABC plugin, enhancing the power of the jABC as a development environment. In particular, users may handily exchange Eclipse with their favorite Java development environment (or they may use jETI/jABC in a stand-alone fashion), and our game-based model checker [5] with their tool of trust, without having to touch anything of the jABC and jETI implementation.

5 Conclusions and Future Work

We have presented jETI, a tool for remote tool integration, which overcomes the bottlenecks of the ETI platform observed over the past seven years. Based on our remote tool integration and execution philosophy, jETI drastically lowers the entrance hurdle for tool providers and allows upgrading of tools essentially for free. In combination with the change to a 'pure Java' approach, jETI has the unique potential to become a standard for enhancing Java development environments with remote component execution, high level coordination, and formal methods-based control. We are planning to make jETI available as open source for this purpose in the near future.

Our current implementation must still be sees as a prototype, a status, which we want to overcome by the end of 2005, which, besides others, requires a redevelopment of the graphical user interface in particular for supporting the Semantic Web and the synthesis functionality. We plan to start the β-testing phase for a version hosted by us in Spring 2005, followed by a first system delivery to partners half a year later.

References

1. Tim Berners-Lee, James Hendler, Ora Lassila: *The Semantic Web* Scientific American, May 2001.
2. Eclipse Foundation: http://www.eclipse.org/
3. T. Margaria: *Web Services-Based Tool-Integration in the ETI Platform*, SoSyM, Int. Journal on Software and System Modelling, to appear, Springer Verlag (Springer Online First DOI: 10.1007/s10270-004-0072-z).
4. T. Margaria, R. Nagel, B. Steffen: *Remote Integration and Coordination of Verification Tools in JETI*, Proc. of ECBS 2005, 12th Annual IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems, 4-7 April 2005, Washington DC (USA), IEEE Press.
5. Markus Müller-Olm, Haiseung Yoo: *MetaGame: An Animation Tool for Model-Checking Games*, Proc. TACAS 2004, LNCS 2988, pp.163-167.
6. B. Steffen: "*Characteristic Formulae*," Proc. ICALP'89, Stresa (I), July 1989, LNCS 372, Springer Verlag, 1989.
7. B. Steffen, A. Ingolfsdottir: "*Characteristic Formulae for Finite State Processes*," *Information and Computation*, Vol. 110, No. 1, 1994.
8. B. Steffen, T. Margaria, V. Braun: *The Electronic Tool Integration platform: concepts and design*, [10], pp. 9-30.
9. B. Steffen, T. Margaria, A. Claßen: *Heterogeneous Analysis and Verification for Distributed Systems*, "SOFTWARE: Concepts and Tools" Vol. 17, N.1, pp. 13-25, March 1996, Springer Verlag.
10. *Special section on the Electronic Tool Integration Platform*, Int. Journal on *Software Tools for Technology Transfer*, Vol. 1, Springer Verlag, November 1997
11. SUN Microsystems. Java WebService Developer Pack *http://java.sun.com/ webservices/*
12. Tomcat homepage: http://jakarta.apache.org/tomcat/
13. WebServices.Org - homepage of the WebServices and SOA communities: http://www.webservices.org/
14. W3C. SOAP *http://www.w3.org/TR/SOAP/*
15. Web Service Choreography Interface (WSCI) 1.0, W3C Note, 8 August 2002, *http://www.w3.org/TR/2002/NOTE-wsci-20020808*.