GOTECH: Aspectizing Server-Side Distribution (a Demonstration Proposal)

Eli Tilevich Stephan Urbanski Yannis Smaragdakis
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332
{tilevich, stephan, yannis}@cc.gatech.edu

Abstract

This demonstration will present the GOTECH (General Object to EJB Conversion Helper) framework. GOTECH can be used with a large class of unaware applications to turn their objects into distributed objects with minimal programming effort. Our framework is developed on top of three main components: AspectJ (a high-level aspect language), XDoclet (a low-level aspect language), and NRMI [2] (a middleware facility that makes remote calls behave more like local calls).

This demonstration is a supplement to our paper in the ASE 2003 technical program [1]. Here, we do not describe specific technical contributions, but instead give a high level description of GOTECH.

Introduction

This demonstration presents a general framework for separating distribution concerns from application logic. Our approach is a mixture of aspect-oriented techniques and domain-specific tools. Just like all other research in aspect-orientation, our goal is to remove low-level technical barriers to the separation of distribution concerns—the assumption remains that the structure of the application is amenable to adding distribution. The specific technical substrate that we target is that of server-side Java applications as captured by the J2EE specification. This domain is technically challenging (due to complex conventions) and has been particularly important for applied software development in the last decade. We show how a combination of three tools can yield very powerful separation of distribution concerns in a server-side application. We call this separation "aspectization", following other aspect-oriented work.

In this demonstration we use our GOTECH framework, to turn an existing scientific application (a thermal plate simulator) into a distributed application. The application-specific code required for the distribution consists of only a few lines of annotations. The rest of the distribution-specific code is provided by the GOTECH framework.

The GOTECH Framework

The GOTECH framework offers the programmer an annotation language[1] for describing which classes of the original application need to be converted into EJBs and how (e.g. where on the network they need to be placed and what distribution semantics they support). The EJBs are then generated and deployed in an *application server*: a run-time system taking care of caching, distribution, persistence, etc. of EJBs. The result is a server-side application following the J2EE specification—the predominant server-side standard.

Converting an existing Java class to conform to the EJB protocol requires several changes and extensions. An EJB consists of the following parts:

- the actual bean class implementing the functionality
- a home interface to access life cycle methods (creation, termination, state transitions, persistent storing, etc.)
- a remote interface for the clients to access the bean
- a deployment descriptor (XML-formatted meta-data for application deployment).

In our approach this means deriving an EJB from the original class, generating the necessary interfaces and the deployment descriptor and finally redirecting all the calls to the original class from anywhere in the client to the newly created remote interface. The process of adding distribution consists of the following steps:

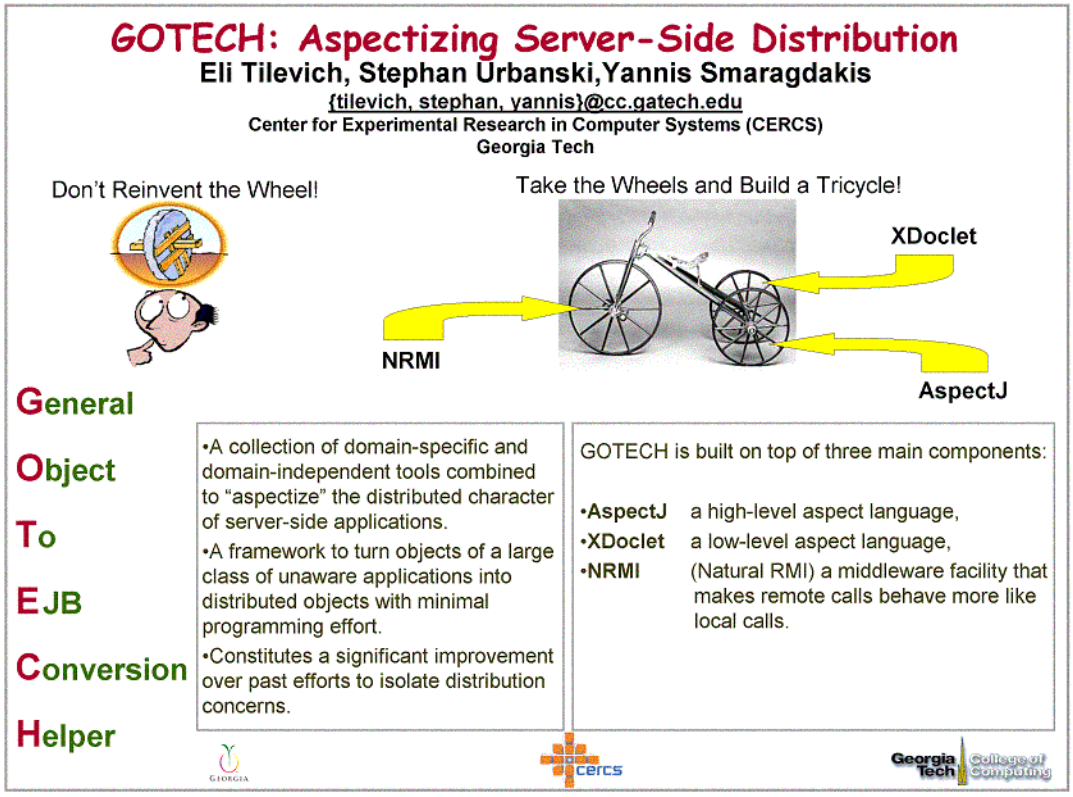
1. The programmer introduces annotations in the source
2. XDoclet processes the annotations and generates the aspect code for AspectJ
3. XDoclet does the EJB generation
4. XDoclet generates the EJB interface and deployment descriptor
5. AspectJ compiler compiles all generated code (including regular EJB code and AspectJ aspect code from step

1. The annotations are introduced in Java source comments as "JavaDoc tags". We use the term "annotation" instead of the term "tag" as much as possible to prevent confusion with the XDoclet "tags", i.e. the XDoclet aspect-language keywords, like `forAllClassMethods`.

1) to introduce distribution to the client by redirecting all client calls to the EJB instead of the original object.

(The XDoclet templates used in step 4 are among the pre-defined XDoclet templates and not part of the GOTECH framework.)

In high-level terms, GOTECH is interesting as an instance of a collaboration of generative and aspect-oriented techniques. The generative elements of GOTECH are very simple exactly because AspectJ handles much of the complexity of where to apply transformations and how. On the other hand, AspectJ alone would not suffice to implement GOTECH.



GOTECH Poster to be used for the presentation.

Availability

All source code for the GOTECH framework and the thermal plate simulator application is publicly available at http://j-orchestra.org.

References

[1] Eli Tilevich, Stephan Urbanski, Yannis Smaragdakis, and Marc Fleury, "Aspectizing Server-Side Distribution", to appear in Proc. *ASE 2003*.

[2] Eli Tilevich and Yannis Smaragdakis, "NRMI: Natural and Efficient Middleware", *Int. Conf. on Distributed Computer Systems (ICDCS)*, 2003. Extended version available from `http://www.cc.gatech.edu/~yannis` .