Reusing Existing Publish/Subscribe Infrastructures:

An Empirical Study of Middleware Reusability Costs and Trade-offs

ISR Institute for Software Research, University of California, Irvine

UCIrvine | University of California

Motivation

Publish/subscribe infrastructures are special class of middleware that support the development of event-driven applications. The popularization of this communication style has demanded an increasing number of application-specific features that publish/subscribe infrastructures need to support. Different **reuse strategies** have been adopted in the development of publish/subscribe infrastructures in the support of different application domains:

1) **Minimal core systems** such as Siena and Scribe, that provide simple and optimized services.
2) **Coordination languages** as Linda, IBM TSpaces and JavaSpaces that provide a common vocabulary for the development of distributed applications.
3) **One-size-fits-all infrastructures** as CORBA-NS and READY that support a large set of features.
4) **Reconfigurable compositional approaches** as GREEN, YANCEES and FACET that can be extended and configured according to different feature sets.

Whereas existing research focus on the benefits of each individual strategy, no research exist that analyzes and compares their reusability trade-offs. This work fills this gap.

Approach

This poster reports a quantitative and qualitative study of four different publish/subscribe infrastructures developed according to different reusability strategies. For such we use these infrastructures to support the requirements of EDEM usability monitoring application scenario. The resulting implementations are compared with one another and with a reference **built-from-scratch (or BFS)** implementation. Our investigation focused on a multi-perspective analysis of the reusability trade-offs in terms of **cognitive distance**, **adaptation costs** and **performance** of existing publish/subscribe infrastructures.

Benchmark

EDEM (Expectation-Driven Event Monitoring) is an event-driven application that relies on special rules (Event-Condition-Action) called agents that process and collect user interface events. Agents process and record events based on sequence detection, abstraction, summarization and recording operations. Existing infrastructures were extended to support EDEM requirements as shown in **Figure1**.

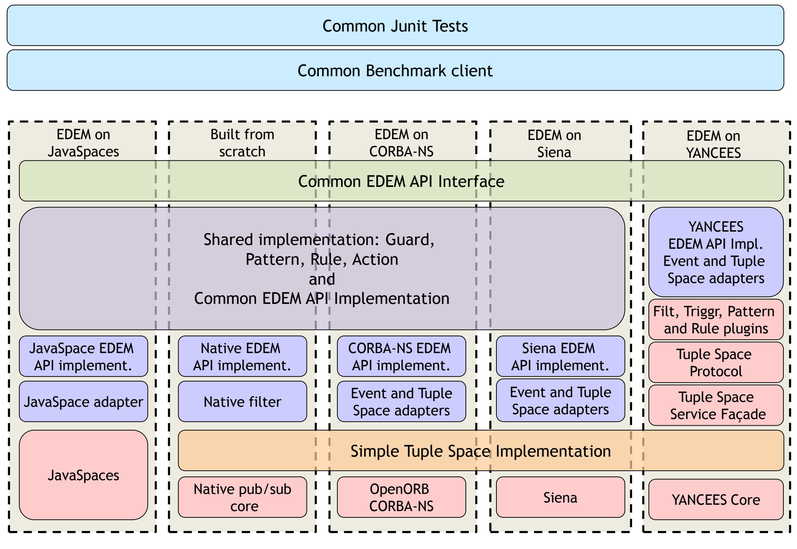


Figure 1: Benchmark where different publish/subscribe infrastructures were used to support EDEM requirements. Columns indicate individual implementations; boxes indicate main components. Components that crosscut different implementations indicate reuse.

Results

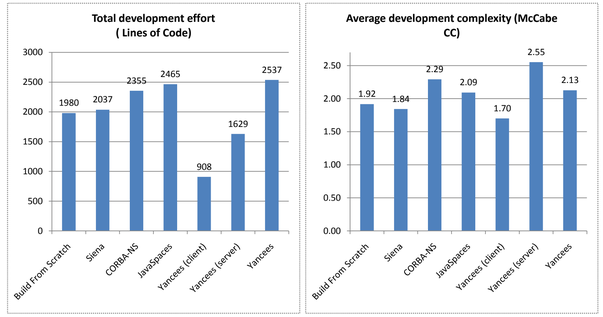
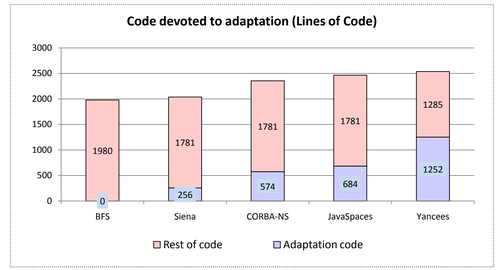


Figure 2: Development effort measured in Lines of Code (LOC) and Cyclomatic Complexity (CC) of using each one of the infrastructures in the support for EDEM requirements

Abstraction level and cognitive distance. The closer to the problem domain an abstraction is, the lower are the development costs. By customizing **YANCEES** to meet EDEM's requirements, the client side effort is reduced (YANCEES client). However, this benefit comes with extra development costs in YANCEES server side. These costs can overweight more simple strategies as Siena minimal core, for example. As a result, the use of compositional strategies as YANCEES only pays-off when the variability in the domain requires different server implementations, with slightly differences between them.

Variability and cognitive distance. The higher the variability of an infrastructure, the broader is its scope. However, as shown in our study with **CORBA-NS**, the price paid is, many times, the increase in the cognitive distance (measured in LOC and CC) associated to configuring and using the infrastructure API and its many options.

Simplicity and efficiency were prevailing. In the majority of our experiments, **Siena** outperformed the other notification servers in both development complexity and efficiency. The use of a simple API and efficient algorithms came very close to the BFS implementation for the selected application domain.



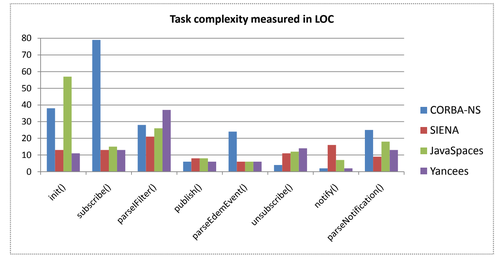


Figure 3: Total and specific adaptation costs (in LOC)

The role of abstraction mismatch. Hidden implementation details and semantic gaps can interfere with the overall reusability and performance. **JavaSpaces**, that at first sight seemed to be very fit to the problem domain requirements, suffered from different semantic mismatches in its subscription and notification capabilities. Moreover, server side implementation details hindered its performance.

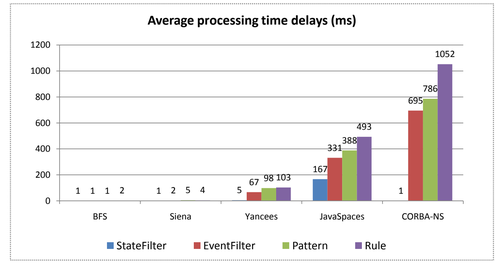


Figure 4: Average publication-notification times measured for each implementation

Contact Information
Professor David F. Redmiles
Roberto S. Silva Filho
Institute for Software Research
University of California
Irvine, California 92697-3425
{redmiles, rsilvafi}@ics.uci.edu

To learn more about YANCEES and have access to a prototype of the system and documentation, please visit the website:
http://isr.uci.edu/projects/yancees

This research was supported by the U.S. National Science Foundation under grant numbers 0534775, 0205724 and 0326105, an IBM Eclipse Technology Exchange Grant, and by the Intel Corporation.

June 2008