Component reuse is a promising direction to develop software more efficiently and cost effectively. One part of software development is the actual programming with an integrated development environment (IDE). We studied three Java IDEs and how they support reuse-oriented software development. We derived evaluation criteria from a known reuse model. As a conclusion we suggest that current Java IDEs need to improve their support for the reuse process.

INTRODUCTION

To cope with the current trend to produce quality software in tightening schedules software developers see reuse as one possible answer (e.g., Lim, 1997; McIlroy, 1968). Reuse of components is one approach to handle reuse (Biggerstaff & Richter, 1987). The basic idea in component reuse is to use some results of the development effort more than once (Basili et al., 1992; Krueger, 1992). To be successful, reuse has to be systematic: it has to be planned in advance and it must be acknowledged in every phase of software development cycle (Lim, 1997). One part of this cycle is programming. Here integrated development environments (IDEs) are especially important. Early IDEs included such tools as an editor and a compiler but currently these environments may include, among other things, source code control, library management, support for workgroups, and version control (Kölling & Rosenberg, 1996).
Java has emerged as one of the most popular programming languages and its advantage is that it closely follows emerging trends in software development. One such trend is the support for component-based development in the form of JavaBeans standard. JavaBeans brings component technology to the Java platform. With JavaBeans you can create reusable, platform-independent components (Sun, 2000).

Our research question is: “Do Java IDEs support the creation and reuse of code components?” As the only components supported by the chosen IDEs are JavaBeans, we limited our study to JavaBeans components.

**SELECTION OF JAVA IDEs AND RESEARCH METHOD**

We selected three Java IDEs that reflect the current state-of-the-practice in Java programming. We chose the environments considering the market share and how well the supplier is known. Using our criteria we selected:

- Forte for Java Community Edition 1.0 Windows Version by Sun Microsystems Inc.,
- Borland JBuilder 3.0 Professional by Inprise Corporation, and
- VisualAge for Java Enterprise Edition Version 3.0 by IBM.

We planned to use Visual J++ 6.0 Professional Edition from Microsoft but we excluded it for its strong orientation towards ActiveX and Windows.

We wanted to find out how Java IDE supports reuse processes involved in component (JavaBean) reuse. To base our evaluation framework on a well known model we chose Lim’s (1997) reuse model over alternatives (e.g., NATO, 1992, Karlson, 1995, see Forsell et al., 2000). Lim’s model is not biased toward any specific implementation technology and it includes code components as well as other software development artifacts (assets in Lim’s vocabulary). Figure 1 shows Lim’s reuse model, its four major activities, and tasks in them.

*Figure 1. The Reuse Process (Lim, 1997)*

```
1. Managing the Reuse Infrastructure

2. Producing Reusable Assets
   2.1 Analyzing Domain
   2.2 Producing Assets
   2.3 Maintaining and Enhancing Assets

3. Brokering Reusable Assets
   3.1 Assessing Assets for Brokering
   3.2 Procuring Assets
   3.3 Certifying Assets
   3.4 Adding Assets
   3.5 Deleting Assets

4. Consuming Reusable Assets
   4.1 Identifying System and Asset Requirements
   4.2 Locating Assets
   4.3 Assessing Assets for Consumption
   4.4 Integrating Assets
```
Providing Automated Holistic Process and Knowledge Assistance during Software Modernization
www.igi-global.com/chapter/providing-automated-holistic-process-and-knowledge-assistance-during-software-modernization/96613?camid=4v1a