Senseo: Enriching Eclipse's Static Source Views with Dynamic Metrics*

David Röthlisberger[1], Marcel Härry[1], Alex Villazón[2], Danilo Ansaloni[2],
Walter Binder[2], Oscar Nierstrasz[1], Philippe Moret[2]

[1]*Software Composition Group, University of Bern, Switzerland*
[2]*University of Lugano, Switzerland*

Abstract

Maintaining object-oriented systems that use inheritance and polymorphism is difficult, since runtime information, such as which methods are actually invoked at a call site, is not visible in the static source code. We have implemented Senseo, an Eclipse plugin enhancing Eclipse's static source views with various dynamic metrics, such as runtime types, the number of objects created, or the amount of memory allocated in particular methods.

1. Introduction

The use of inheritance, subtyping, and polymorphism in object-oriented applications leads to scattered code that is hard to maintain when just reading the static source code. For a polymorphic call site, for instance, the invoked methods may not be known before runtime. Since IDEs such as Eclipse only show a purely static perspective of a software system, they do not help developers understand the execution flow. For instance, Java developers are encouraged to declare variables and method arguments using interface types. However, finding the runtime types taken by a variable or method argument is often impossible without gathering runtime information before.

Traditionally, debuggers and profilers are used for exploring runtime program behavior. Debuggers are interactive tools, allowing breakpoints to be set in order to investigate the execution flow or to inspect values of variables at particular program locations. They are not intended for collecting comprehensive type information and dynamic metrics over several program executions. Profilers support performance optimization, focussing on measuring dynamic metrics that relate to resource consumption, such as CPU time or memory allocation. They usually do not collect information about runtime types of senders, receivers, arguments or return values.

*In: *25th International Conference on Software Maintenance*, 2009

Senseo collects both runtime type information and performance-related metrics. It integrates the gathered information in the Eclipse IDE so as to support a wide range of software engineering tasks, including program understanding, reverse engineering, software maintenance, and performance tuning. In contrast, debuggers and profilers are designed to support one particular use-case. While the information gathered by debuggers or profilers is usually presented only for a single run of a program, *Senseo* supports metrics aggregation over multiple runs.

2. Dynamic Metrics

Senseo gathers and integrates in Eclipse the following dynamic metrics.

Message sending. In object-oriented programs, understanding how objects send messages to each other at runtime is crucial. However, often the concretely invoked method is difficult to reach from the static source code, as called methods are not always implemented in the statically defined type (*i.e.*, class), but in its super- or subtypes, when inheritance, respectively dynamic binding, are used. *Senseo* enhances all executed call sites in the IDE with a list of the actually invoked methods. Optionally, *Senseo* gathers additional information, such as the runtime types of receivers, arguments, or return values.

Number of invocations. This metric has two variants: Presenting how often a method was invoked and how many method invocations a particular method triggered. This metric helps developers locate hot methods.

Number of created objects. This metric illustrates how many objects are instantiated by a method, either just within the method body, or aggregated for all (direct and indirect) callees. It supports developers locate inefficient code creating a high number of objects.

Allocated memory. As objects of different types vary in size, *Senseo* also shows the amount of memory allocated by the executions of methods. Comparing this metric with the number of created objects pinpoints methods creating many large objects; these methods are primary candidates

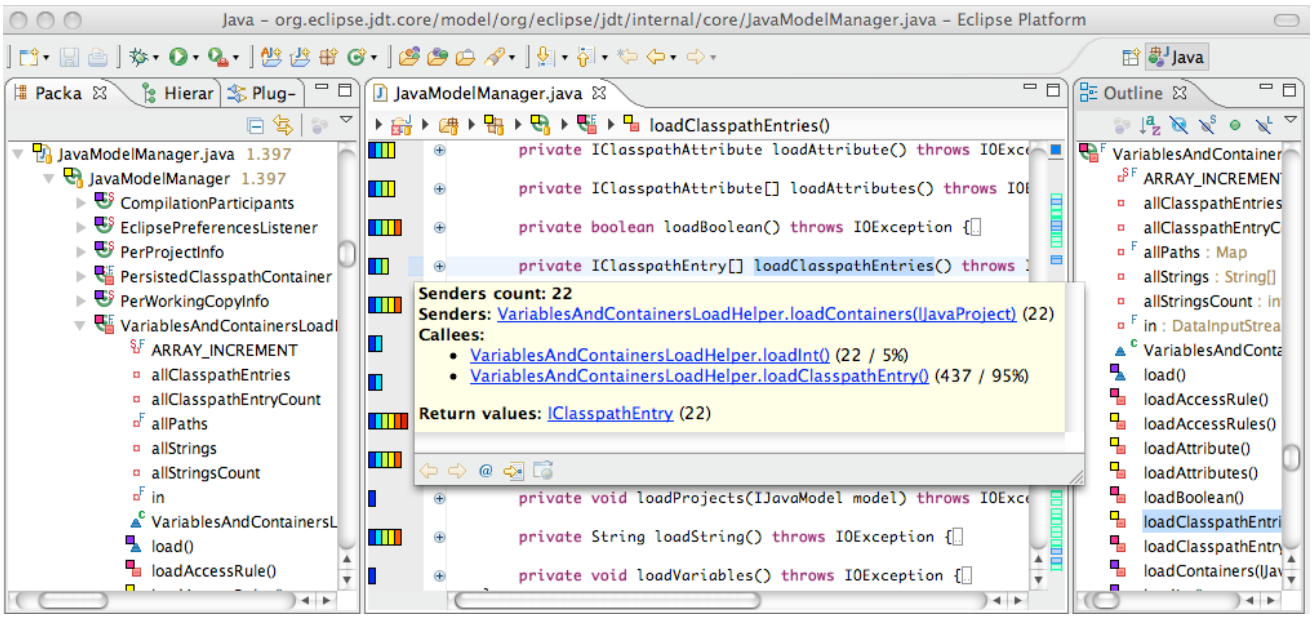

Figure 1. Senseo: Dynamic metrics in package tree, outline, ruler columns, and source hovers

for optimizations.

3. Senseo

Senseo integrates these dynamic metrics in Eclipse's source perspectives by enhancing the package tree, the outline, the ruler columns next to the source editor, and the hovers appearing in the source code.

Hovers. *Senseo* enhances method declaration hovers to show all senders, all callees, and optionally also all receiver and argument types with which particular methods were invoked at runtime. Hovers also show metric values, such as number of invocations, number of created objects, or number of occurrences of particular callees, receiver types, etc. Developers can interact with the hover, for instance, in order to directly navigate to called methods. For invoked methods in the method body, *Senseo* shows similar information in the hover, that is, receiver or argument types, number of invocations, but also performance-related information of the various invoked methods, such as the number of created objects. This is particularly useful to estimate the costs imposed by particular method calls while reading or writing source code.

Ruler columns. *Senseo* extends the ruler columns in the source editor to show annotations for dynamic metrics. The overview ruler to the right of the source editor shows metric values using three different icons in a hot/cold color scheme: *blue* represents a low, *yellow* a median, and *red* a high metric value. The ruler column on the left hand, which just covers the visible part of a source file, uses a more fine-grained scale from 1 to 6 to visualize dynamic metrics (see Figure 1). A completely filled bar thus denotes methods with highest metric values. These annotations in the columns also provide hovers showing exact metric values. *Senseo* allows developers to dynamically configure the presented metrics or to show the combination of multiple metrics, for instance, combining the number of created objects with the amount of allocated memory in a method.

Package Explorer. To show dynamic metrics for higher-level artifacts, such as classes or packages, *Senseo* aggregates the metric values from single methods and visualizes them in Eclipse's package explorer or the outline by using three different icons employing the same heat coloring scheme as in the overview ruler, that is, *blue*, *yellow*, and *red*. This enhancement eases the location of hot spots in the entire system, *i.e.*, classes or packages heavily consuming memory or instantiating many objects.

4. Availability

Senseo relies on MAJOR to gather dynamic metrics. Both tools are freely available. A complementary research paper submitted to ICSM'09 explains the principles and techniques underlying *Senseo*. Currently, we are conducting a user study to evaluate the impact of *Senseo* on developers' productivity in various software engineering tasks.

Acknowledgments. We gratefully acknowledge the financial support of the Swiss National Science Foundation for the projects "Bringing Models Closer to Code" (SNF Project No. 200020-121594, Oct. 2008 – Sept. 2010) and "FERRARI – Framework for Efficient Rewriting and Reification Applying Runtime Instrumentation" (SNF Project No. 200021-118016/1, Oct. 2007 – Sept. 2010).