# OGAN: Visualizing Object Interaction Scenarios Based on Dynamic Interaction Context

Satoshi Munakata, Takashi Ishio, Katsuro Inoue
Osaka University
1-5 Yamadaoka, Suita, Osaka, 565-0871, Japan
{mnktsts, ishio, inoue}@ist.osaka-u.ac.jp

## Abstract

*Visualizing an execution trace of an object-oriented system as sequence diagrams is effective to understand the behavior of the system. To support developers to understand concrete interaction among classes, our tool named OGAN extracts sequence diagrams representing interaction scenarios for a pair of classes specified by a user. OGAN classifies objects into groups based on their neighbor classes that directly interact with the objects, and visualizes interaction scenarios for each pair of object groups.*

## 1. Introduction

A sequence diagram extracted from an execution trace visualizes actual collaborations of objects that provide a larger unit of program comprehension than classes [2]. Although several tools support sequence diagram extraction, a sequence diagram extracted from an execution trace is too complicated for developers to understand interaction between a pair of interesting classes since such a diagram represents the whole execution trace involving a large number of objects and method calls.

An approach to focusing on the behavior of a particular class is extracting a collaboration diagram for a class instead of a sequence diagram [3]. Another approach is specification mining (or protocol mining) that extracts a finite state machine representing the behavior of an instance of a class from execution traces [4]. While a collaboration diagram and a finite state machine may approximate all possible sequences of method calls for a single class, they do not provide concrete examples of the behavior of the class.

Our tool named OGAN extracts a set of sequence diagrams that each visualizes an interaction scenario for a pair of classes of interest to a user. Each sequence diagram shows an interaction among a different set of classes interacting with the user-specified classes. These diagrams help a user to understand concrete examples of interaction among the classes.

To extract interaction scenarios, we compute dynamic interaction context for each instance of the user-specified classes in an execution trace; dynamic interaction context of an object is a set of classes that directly interact with the object. If two instances of a class have different context, OGAN extracts two independent diagrams since a diagram involves a different set of actors from another.

In the rest of this paper, Section 2 explains how the tool works. Section 3 describes the summary and future work.

## 2. Extracting Object Interaction with OGAN

OGAN is an extension for Amida, our sequence diagram extraction toolkit [1]. OGAN takes as input an execution trace of a Java program recorded by Amida Profiler and a pair of classes $c_1$ and $c_2$ that a user would like to inspect. OGAN extracts interaction scenarios from the trace, and visualizes the scenarios on Amida Viewer.

Amida Profiler records a sequence of method call events and return events into a file; an event comprises timestamp, thread ID, method signature actually executed, caller object ID, and callee object ID.

To explain how OGAN work with a trace, we use a Java program named `scheduler` that is a small calendar application to manage personal schedule data in this section. We executed `scheduler` and added three events into the calendar. The recorded execution trace involves 8551 method calls and 2908 objects of 38 classes. We analyze interaction between two classes in `scheduler` that represent a date: a GUI class `DateCell` and a data model class `CalendarDate`.

OGAN computes dynamic interaction context $Use(o)$ and $Used(o)$ for each instance $o$ of $c_1$ and $c_2$. If an object $o1$ calls another object $o2$ in an input execution trace, we extract the following context information: $Class(o1) \in Use(o2)$ and $Class(o2) \in Used(o1)$ where $Class(o)$ rep-

**Table 1. The groups of DateCell objects:** $G(\mathrm{DateCell}) = \{s_1, s_2, s_3, s_4\}$

| Group ID | #Instances | Related Classes: $Use(o), Used(o)$ $(o \in s_k)$ |
|---|---|---|
| $s_1$ | 882 | { `MonthTableModel` }, {} |
| $s_2$ | 33 | { `MonthTableModel`, `DateCellRenderer` }, {} |
| $s_3$ | 90 | { `MonthTableModel`, `DateCellRenderer` }, {`CalendarDate`} |
| $s_4$ | 3 | { `MonthTableModel`, `DateCellRenderer`, `CalendarFrame`}, {`CalendarDate`} |

**Table 2. Relationship among DateCell groups $\{s_1, s_2, s_3, s_4\}$ and CalendarDate groups $\{t_1, t_2, t_3\}$. A number following a group ID is the number of instances in the group.**

| | $t_1$ (637) | $t_2$ (90) | $t_3$ (3) |
|---|---|---|---|
| $s_1$ (882) | | | |
| $s_2$ (33) | | | |
| $s_3$ (90) | | 90 pairs (1-to-1) | |
| $s_4$ (3) | | | 3 pairs (1-to-1) |

resents the class name of an object $o$. Using the dynamic interaction context, OGAN classifies the objects of class $c_k$ into groups $G(c_k) = \{G_1, \cdots, G_l\}$ such that:

$$\forall o1 \in G_i, o2 \in G_j, Use(o1) = Use(o2) \wedge Used(o1) = Used(o2) \Leftrightarrow i = j \ \ (1 \leq i, j \leq l)$$

This classification rule means that the objects in a group interact with the same set of classes. Table 1 shows the result of classification for `DateCell` class in `scheduler`.

After the classification step, OGAN computes relationship among groups $G(c_1) = \{s_1, \cdots, s_l\}$ and $G(c_2) = \{t_1, \cdots, t_m\}$. If $o1 \in s_i$ calls a method of $o2 \in t_j$, OGAN extracts a call relation $\langle s_i, t_j \rangle$. Table 2 shows relationship among `DateCell` groups $\{s_1, s_2, s_3, s_4\}$ and `CalendarDate` groups $\{t_1, t_2, t_3\}$; `DateCell` objects in $s_3$ and $s_4$ call their corresponding `CalendarDate` objects in $t_2$ and $t_3$, respectively.

OGAN recognizes relationship among object groups as *interaction scenarios*. For each relationship $\langle s_i, t_j \rangle$, OGAN randomly selects a pair of *seed* objects that directly interact with each other from the pair of groups. OGAN extracts a sequence of method call and return events directly related to the seed objects, and shows the events as a sequence diagram using Amida Viewer. Figure 1 is a diagram representing an interaction scenario $\langle s_4, t_3 \rangle$; `DateCell` requests schedule data to `CalendarDate` to draw GUI.

## 3. Summary and Future Work

OGAN extracts a set of sequence diagrams that are suitable to understand concrete interaction scenarios between a pair of classes. In future work, we would like to implement automatic visualization of difference among extracted interaction scenarios. We are also planning to investigate a way to select a better representative set of objects.


**Figure 1. An extracted sequence diagram representing an interaction scenario $\langle s_4, t_3 \rangle$: a DateCell requests schedule data to a CalendarDate.**

## References


[1] T. Ishio, Y. Watanabe, and K. Inoue. AMIDA: a sequence diagram extraction toolkit supporting automatic phase detection. In *Companion Volume of ICSE*, pages 969–970, 2008.

[2] T. Richner and S. Ducasse. Using dynamic information for the iterative recovery of collaborations and roles. In *Proceedings of ICSM*, pages 34–43, 2002.

[3] D. Röthlisberger and O. Greevy. Representing and integrating dynamic collaborations in IDEs. In *Proceedings of WCRE*, pages 74–77, 2008.

[4] N. Walkinshaw and K. Bogdanov. Inferring finite-state models with temporal constraints. In *Proceedings of ASE*, pages 248–257, 2008.