Software Engineering Laboratory

Feature-level Phase Detection for Execution Trace Using Object Cache

Yui Watanabe, Takashi Ishio, Katsuro Inoue

Osaka University, Japan

Contents

- **Automatic phase detection for execution traces of object-oriented programs**

1. Background
   - Visualizing program behavior
2. Algorithm
   - An automatic phase detection technique
3. Case study and Result
   - analyzing several use-case scenarios on two industrial systems
4. Summary and future works

Visualizing Program Behavior

- **Object Oriented Programs are difficult to maintain because of dynamic binding**
  - Visualization of program behavior is useful for developers to understand and debug OO-programs

- **Many tools are proposed to visualize dynamic behavior**
  - e.g. : AMIDA
    - A tool to visualize a Java execution trace as a sequence diagram

Technical issue

- **How to handle a huge amount of events included in an execution trace?**
  - Approaches to reduce the size of an execution trace
    1. Filtering utility and library methods
    2. Visualizing an overview of an execution trace
    3. A query based interface to select interesting events
  - To understand an overview of an execution trace
  - To investigate the detail of interesting features

- **Dividing an execution trace into small Phases corresponding to features**
  - Developers can visualize only interesting features.

Definition of “Phase”

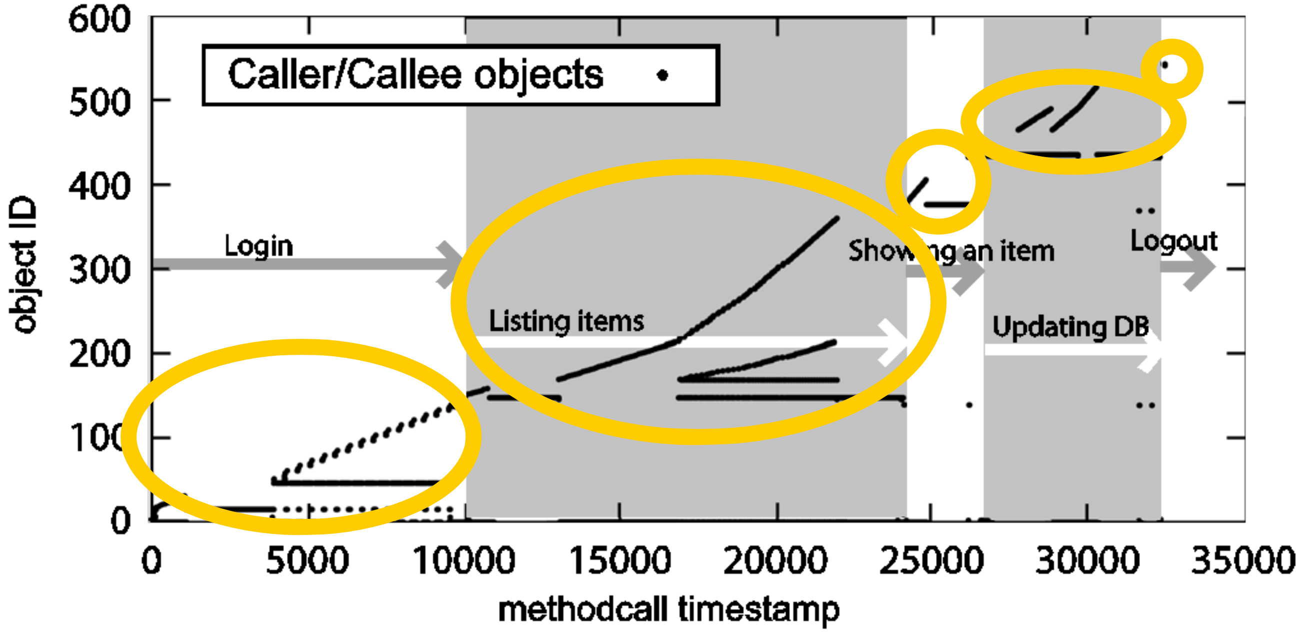
- **A Phase in a execution trace**
  - A consecutive sequence of run-time events in an execution trace
  - An execution trace = a sequence of phases
  - **Feature-level phase**
    - Corresponding to an execution of a feature in the system
  - **Minor phase**
    - Corresponding to one of the tasks to achieve a feature
- **A trace comprises several feature-level phases.**
- **A feature-level phases comprises several minor phases.**

<Phases of a Sample Trace>

Feature-level phase	Minor phase (18)
1. Login	**Show login form**
	Login
	Get pre-user settings
	Show entrance page
2. Listing items in DB	**Get management information**
	Get pre-user items
	Get list of items
	Show list of items
3. Show the detail of an item	**Get an item ID**
	Get a detail of the item
	Show the item information
4. Updating the item information	**Get an item ID**
	Update the item information
	Get a detail of the item
	Show the item information
5. Logout	**Logout**
	Show login form
	Shutdown the system

Key idea: different objects work for different features

- Caller and callee object ID in each method calls in the sample trace



- Monitoring changing of a working set of objects using a **Least-Recently-Used (LRU) cache**

Phase Detection Process

1. **Execute a program and record an execution trace**

2. **Detect phase transitions**
   - Each phase uses its own working set of objects.
   - ➔ Changing of working set of objects = phase transition
3. **Identify the head event of each phases**
   - The beginning of a phase corresponds to a method call event following the end of a method belonging to the previous phase.

➔ **Output: the list of the events that is the head of the phases**

Recording an execution trace

- **Each method call event has the following attributes:**
  - Timestamp
  - Caller object ID
  - Callee object ID
  - Call stack information
    - The depth of the call stack

- **A profiler based on JVMTI**

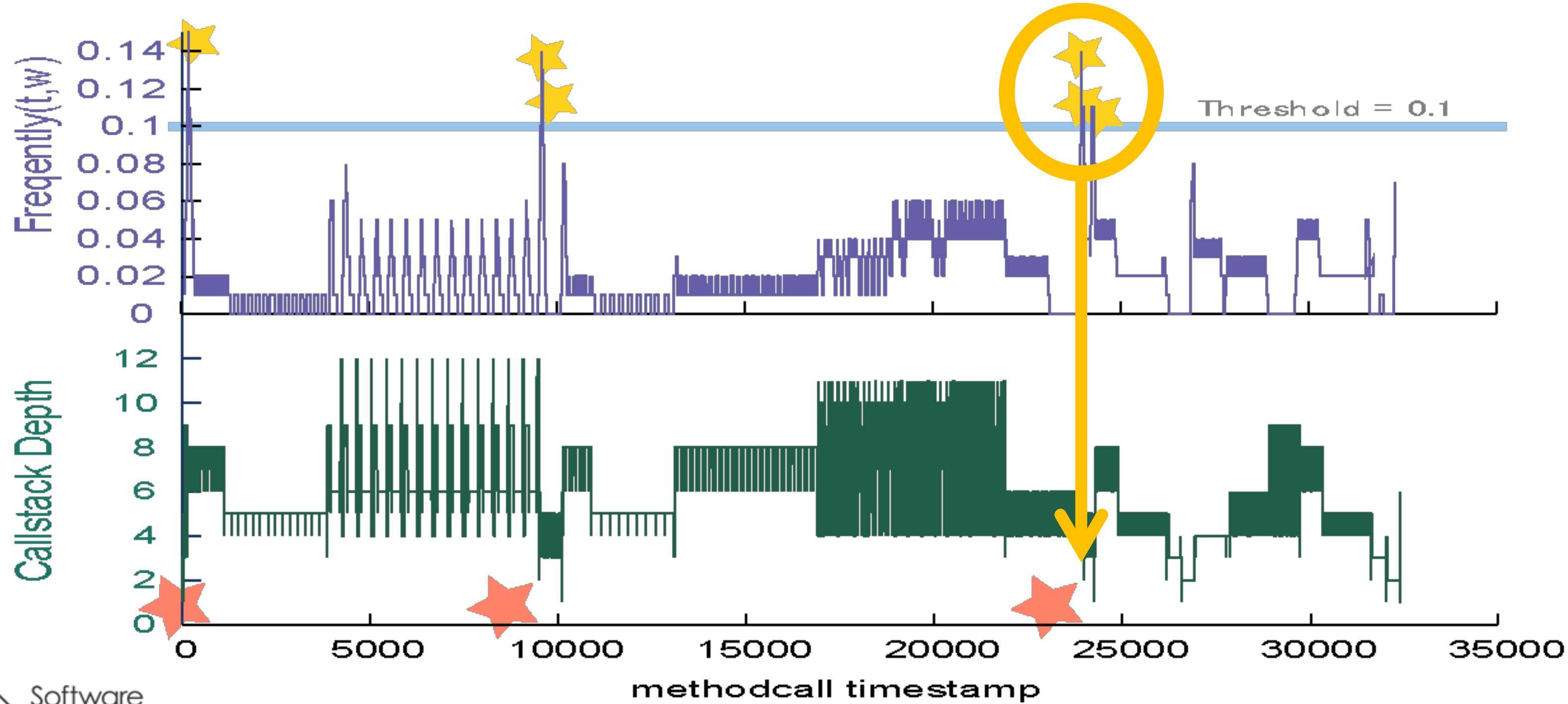
Detecting Phase Transitions

- **Observing the working set of objects using a LRU cache**
  - Push the CallerID and CalleeID into the LRU cache
  - Record whether the cache is updated and calculate frequency

Timestamp	...	94	95	96	97	98	99	100	101	102	103	104	...
CallerID	…	137	137	-1	2	2	146	147	8	146	11	148	…
CalleeID	…	145	137	2	141	146	147	8	148	11	148	149	…
LRU Cache (cache size = 6)	…	145	137	2	141	146	147	8	148	11	148	149	…
	…	137	145	-1	2	2	146	147	8	146	11	148	…
	…	146	146	137	-1	141	2	146	147	148	146	11	…
	…	141	141	145	137	-1	141	2	146	8	8	146	…
	…	2	2	146	145	137	-1	141	2	147	147	8	…
	…	-1	-1	141	146	145	137	-1	141	2	2	147	…
Update Flag	…	0	0	0	0	0	1	1	1	1	0	1	…
Frequency (window = 5)	…	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.8	0.8	0.8	…

Identifying the Head Event of each phase

- **For each events that have higher frequency**
  - Go back to a event that is likely to trigger the new phase
  - Identify an event who has the local-minimum depth of the call stack



Case Study

- **Can we get correct phases by our approach?**
  - Compare phases automatically detected by our approach with phases manually identified by developers
- **How do the parameters effect to result ?**
  - Use various "Cache size" and "Window size"
    - Cache size : the size of a LRU cache
    - Window size : the sliding window calculating frequency

Procedure of the Case Study

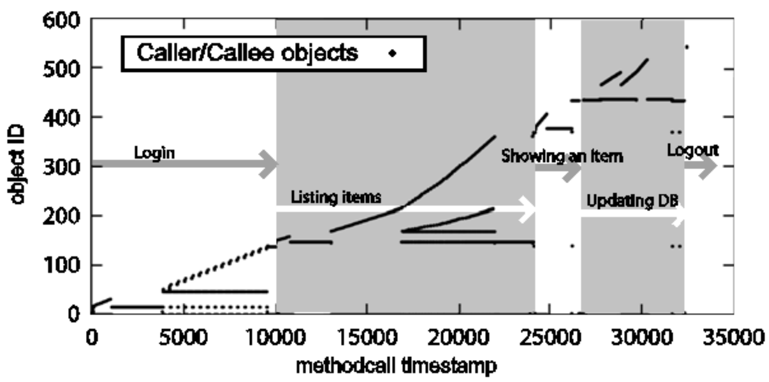
1. **Record execution traces from 2 industrial systems**
   - Tool Management System: 1 program, 4 scenarios, 4 traces
   - Library Management System: 5 programs, 1scenario, 5 traces
2. **Ask developers of the systems to manually identify all phases in each trace**
   - As correct feature-level phases and minor phases
3. **Detect phases by our method with various parameter settings**
   - 9 traces × various parameter settings = about 10,000 outputs
   - Less than 5 minutes on a workstation (Xeon 3.0 GHz)
4. **Compare all phases detected by our approach with correct phases manually identified by developers**

Result of the Case Study

Evaluation

- The number of output phases with each parameter settings
- Comparing the head event of output phases with one of parameter changes
- Precisions and recalls with several parameter settings

<Working objects of the Sample Trace>

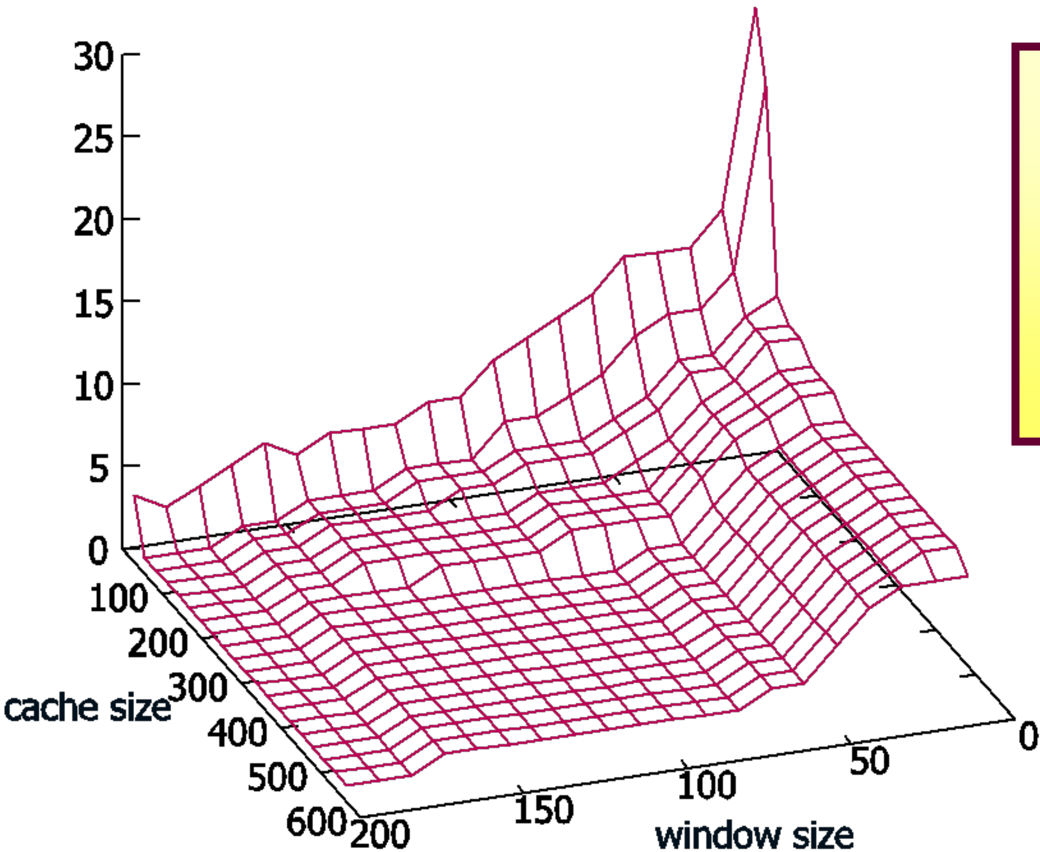


<Phases of the Sample Trace>

Feature-level phase	Minor phase (18)
1. Login	Show login form
	Login
	Get pre-user settings
	Show entrance page
2. Listing items in database	Get management information
	Get pre-user items
	Get list of items
	Show list of items
3. Show the detail of an item	Get an item ID
	Get a detail of the item
	Show the item information
4. Updating the item information	Get an item ID
	Update the item information
	Get a detail of the item
	Show the item information
5. Logout	Logout
	Show login form
	Shutdown the system

The number of output phases

- **with various cache size and window size**

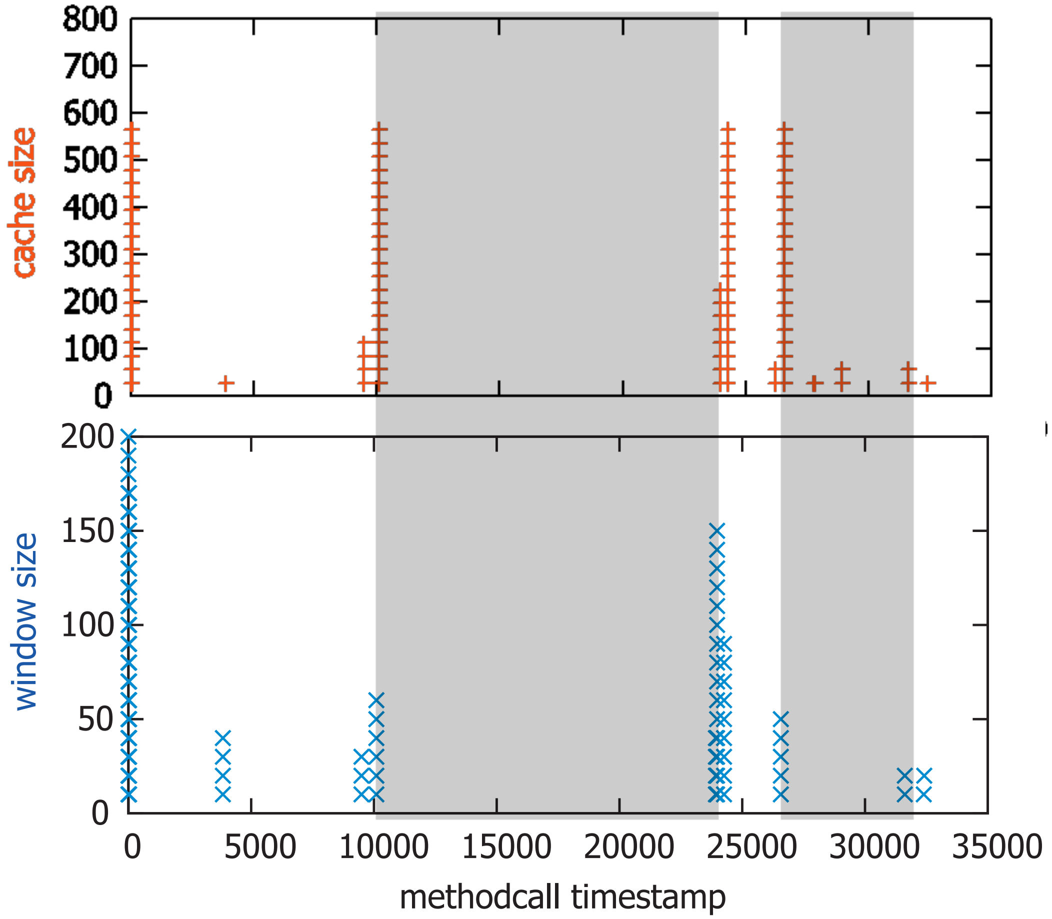


A smaller cache size / window size lead to output a large number of phases.

Effect of ether cache size / window size

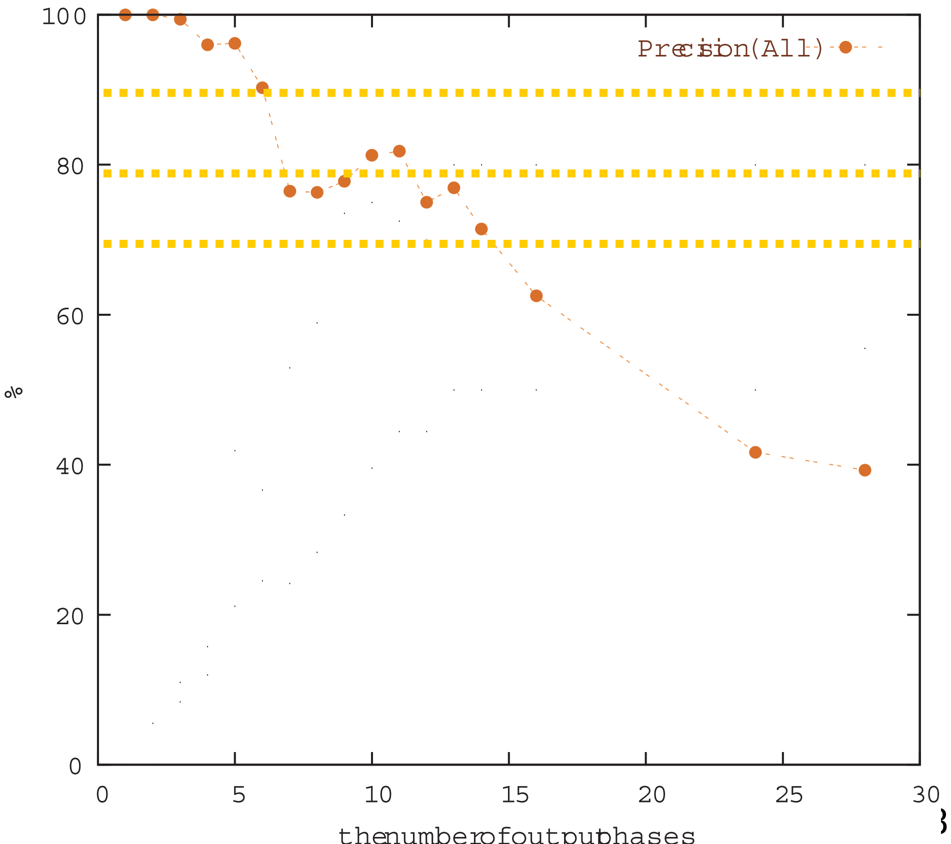
- **Result from Various cache size and fixed window size**
- **Result from Various window size and fixed cache size**

The result is **stable.**



Precision with several parameter settings

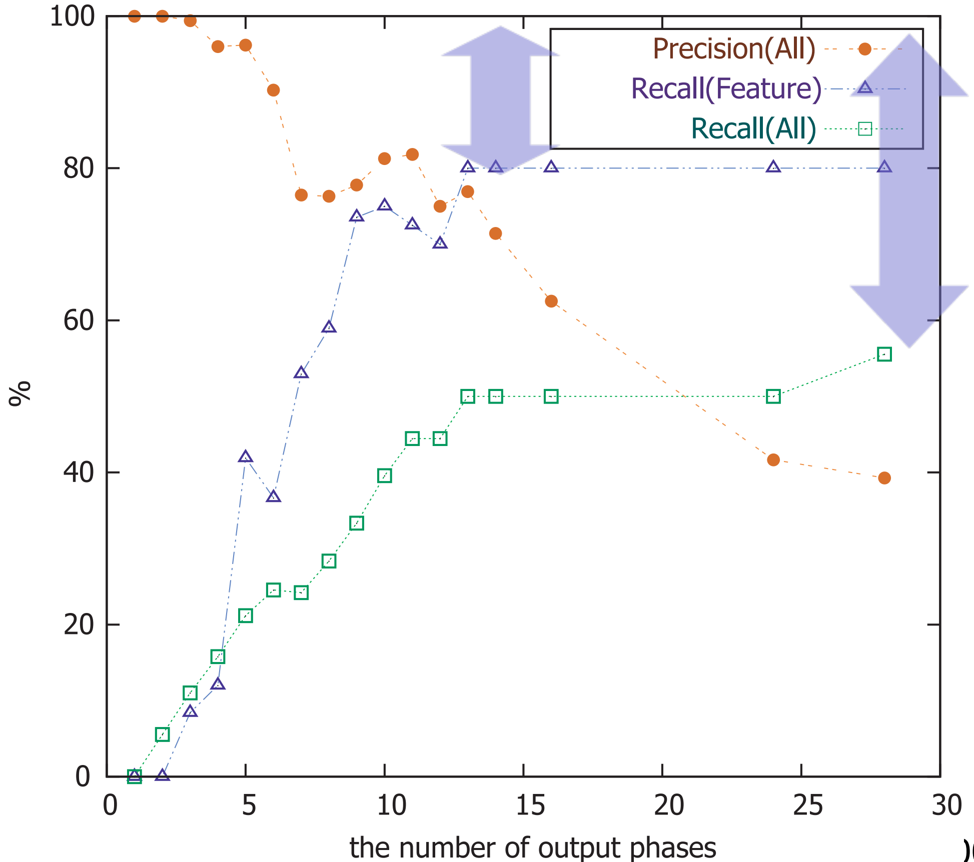
- **Average precision of all parameter settings that result the same number of output phase**



Very high precision with smaller number of output phases

Recall with several parameter settings

- **Average recalls of all parameter settings that result the same number of output phases**



Increasing with the number of output phases

Never detected some correct phases comprising a extremely small number of objects and method call events.

Average Precision and Recall for all traces

- **Average precision and Recall for various parameter settings that detect the same number of phases**
  - Tool Management System (Feature-level phases : 3 to 5)

#Phases	Recall(Feature)	Recall(All)	Precision
5	0.56	0.39	0.93
10	0.90	0.48	0.80

  - Library Management System (Feature-level phases : 15)

#Phases	Recall(Feature)	Recall(All)	Precision
10	0.24	0.20	0.99
15	0.53	0.29	0.98
20	0.45	0.38	0.96

Developers can apply our approach if they could estimate the number of feature-level phases from a use-case scenario.

Summary

- **A novel approach to efficiently detecting phases using a LRU cache for observing a working set of objects**
  - Light weight and easy to implement
  - Detect phases with precision
  - With only a little knowledge on an execution trace

- **Future work**
  - to investigate a way to automatically map an execution trace to an use-case scenario
  - to investigate how the algorithm work in concurrent systems other than enterprise systems