The Evaluation of *Daikon*: utilization of *Daikon* in the POI Data Inspection System

Kuyul Noh, Changki Kim, Jonggul Park and Jaeha Song

4WD Team

Master of Software Engineering Program

School of Computer Science

Carnegie Mellon University

{knoh, ckkim, jgpark, jaehas}@andrew.cmu.edu

1. Introduction

Program invariant is a property that is true at a particular program point or points in a program. It explicates data structures and algorithms and is helpful for programming tasks from design to maintenance. *Daikon* is an implementation of dynamic detection of likely invariants; that is, *Daikon* invariant detector reports likely program invariants.

In this paper, we first describe the basic architecture of *Daikon* and the POI Data inspection system that is the target application for the *Daikon* tool. Then, we mention the scope and approach of this project. Next, we focus our experiments, their results, and the comments on the results we obtained or we didn't obtain. Then, we propose the possible future works. Finally we conclude with the summary of this report.

2. Background

2.1 *Daikon*

There are two ways to obtain invariants, static analysis and dynamic analysis. Static analysis is generally used to generate and verify annotations and analyze the code for type and memory safety. The most common static analysis is dataflow analysis. On the other hand, Dynamic analysis is primarily used to detect possible invariants. It does not require additional initial input from a human such as annotations or specifications. However, it must execute the program being analyzed with a large test suite to infer possible invariants.

Daikon is one of most successful tools for detecting invariants among the dynamic analysis techniques. It can detect properties in Java, C++, and Perl. It is a technique for postulating likely invariants from program runs: it runs target program, examines the values that it computes, and looks for patterns and relationships over those values, reporting the ones that are always true over an entire test suite and that satisfy certain other conditions. Figure 1 shows the high-level architecture of *Daikon* invariant detector.

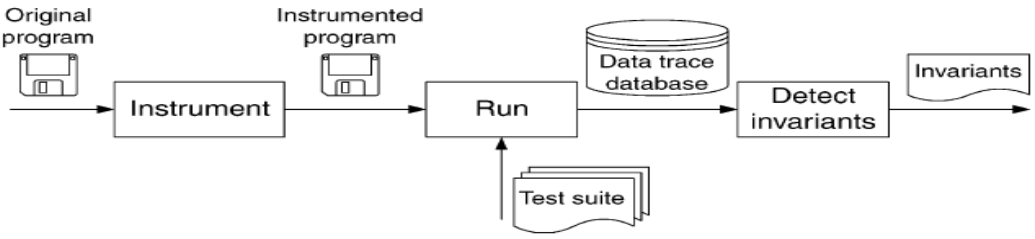

Figure 1: Architecture of the *Daikon* tool

The outputs of *Daikon* are likely invariants: they are not guaranteed to be universally true, because the test suite might not characterize all possible executions of the program. The inference step tests a set of possible invariants against the values captured from the instrumented variables: those invariants that are tested to a sufficient degree without falsification are reported to the programmer. All of the steps in Figure 1 are progressed automatically by tool except selecting a test suite.

2.2 POI Data Inspection System

The car navigation system is no longer optional in the automobiles today. One of the most important parts of the car navigation system is the quality of the data representing the Point of Interest (POI), locations of buildings, restaurants, public offices, etc. that drivers are interested. It is closely related to the good services for the customer. Figure 2 shows where our system is in within the whole car navigation business process.

In Figure 2, the raw POI data files come from the third party content providers in MS Access file format. It has 2 million records and each record has several meaningful columns such as name, address, and zip code.

The quality of raw POI data is relatively poor because there are logical and duplication errors that are injected during the initial creation of the raw POI data file by the third parties. A typical error is duplication of the same content. The error that is more complicated is logical error. For example, it is a logical error when a record mentions fifth floor of a three-

story building.

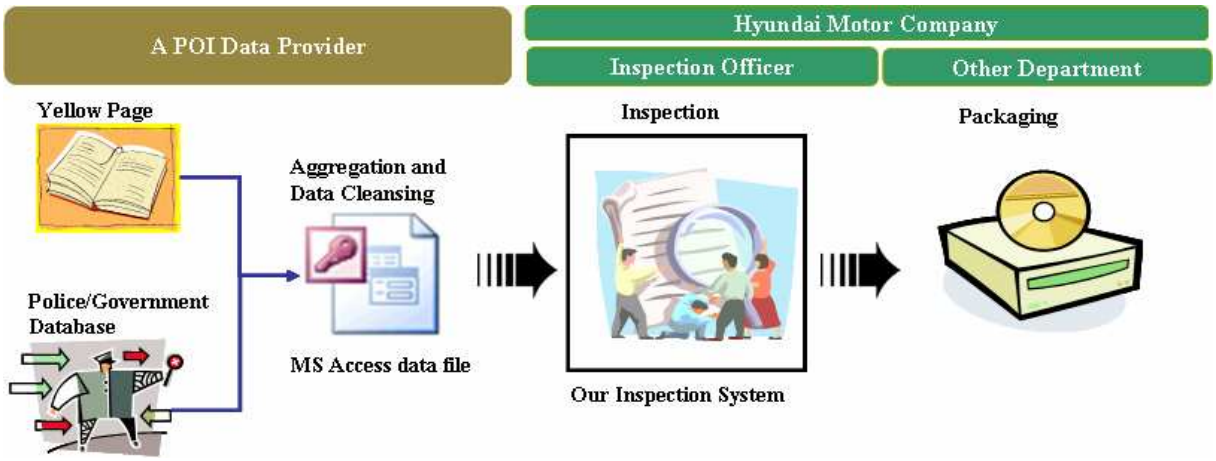

Figure 2: Business flow of POI data inspection system

No one knew how many POI raw data had defects such as logical inconsistency or duplication error. Our customer have guessed the error rate through the manual inspection through data sampling and asked the content provider to fix it when the defect rates are unacceptably high. In this way, it took a long time to make a final product of good quality. It might prevent the customer from leading in the car navigation market. Therefore, they wanted to have a tool that could automate POI data inspection and reduce the time for product.

Our POI inspection system can resolve this problematic situation. It would be specialized tool to help our customer, HMC (Hyundai Motor Company) solve the previous, tedious works automatically. After inspection, our system generates the refined POI data as output that has good quality for the next packaging step.

2.3 Scope and Approach

Our primary goal through this project is to apply *Daikon* to our POI Data inspection system and to find how *Daikon* can be applied to database centric application. As a result, we wanted to improve the correctness of the POI data and the performance of our inspection system. To achieve this goal, we considered the following two strategies.

- **Reduce inspection time by applying *Daikon* as preprocessor in runtime**

In our inspection system, it takes a long time to inspect all the raw POI data with the entire rules. This approach was motivated by the long inspection time. If only the optimal rules, not

the entire rules, can be applied to the inspection process, we can reduce the inspection time dramatically in runtime. Therefore we intended to use *Daikon* as a preprocessor to find the optimal rules prior to the main inspection operation. Figure 3 shows the overall diagram of this approach.

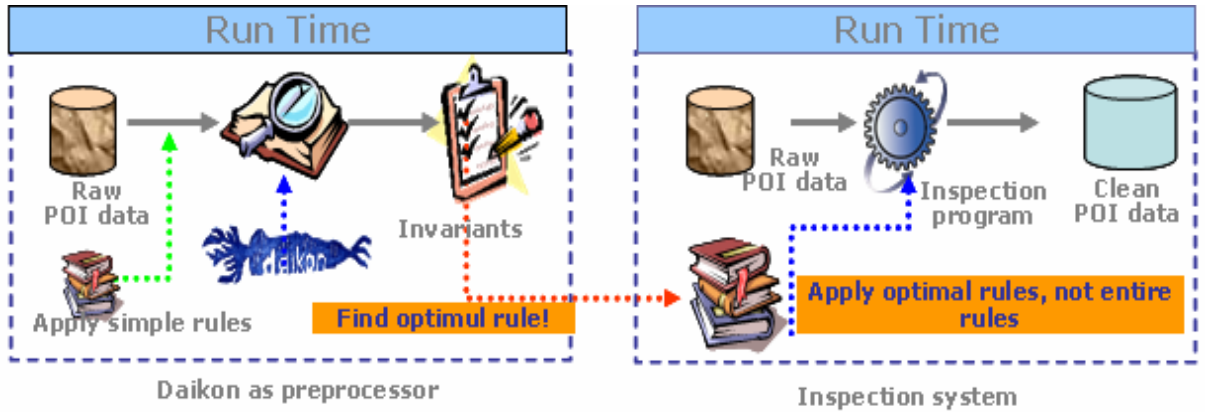

Figure 3: *Daikon* as preprocessor in run time

- **Find new rules by applying *Daikon* in design time**

To detect the logical and duplication errors, our customer and we already came up with the 200 inspection rules. However, the rules we found were not perfect; there is still room for improvement. This approach was motivated by imperfectness of the inspection rules. Therefore, we intended to use *Daikon* to find new rules or refine existing rule set in design time. It may enable us to make our existing rule set robust. Figure 4 shows the overall diagram of this approach.

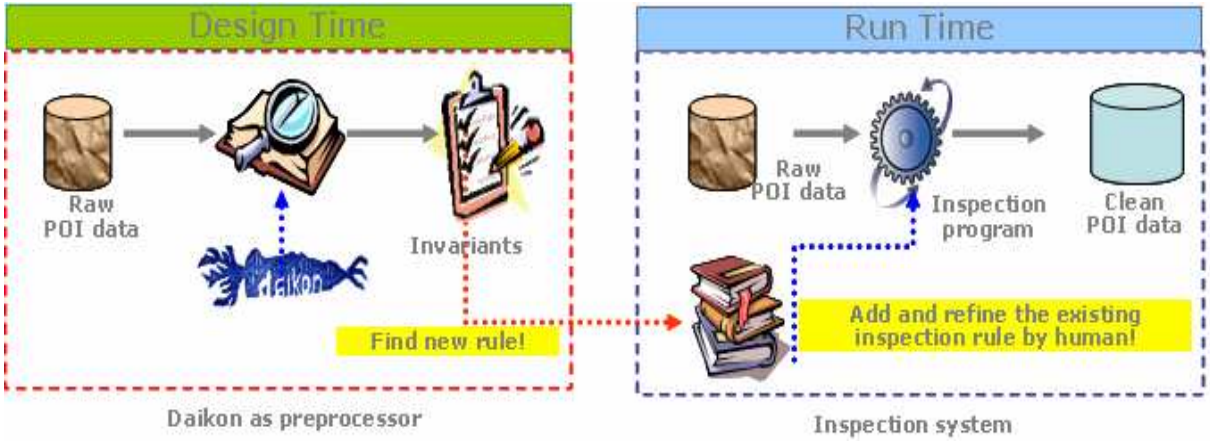

Figure 4: *Daikon* as preprocessor in design time

3. Experiments

3.1 Environments

Similar to the traditional UNIX client server styled applications, the tool also consists of two parts. One part is front end that provides users with interface to the *Daikon* engine and converts input data from some other format into its expected input format. Another part is *Daikon* engine itself for the back end processing. For the Java language, the tool has two kinds of front end: *dfej* and *Chicory*.

dfej is the old one of the two front ends, and it has an Eclipse plug-in integration. Since Eclipse is the official Integrated Development Environment for our project, it is reasonable to try with this option for our front end. Therefore, as our first trial, we installed the *Daikon* Eclipse plug-in on our system. Then, it was easy to derive *Daikon* instrumented source files in a dedicated project by clicking existing java source file, java source package, or even java project.

In addition to the first option, we also tried to test the second option: using *Chicory*, the newly provided pure Java front end that is more recommended by the developers because of the following reasons:

- It supports Java 5.
- It is much easier to use.
- It is pure Java implementation, so it has better platform independence.

Although the first option has Eclipse plug-in, *Chicory* became easier to use after we wrote a simple Ant script for the tool integration. The following figure shows the steps of invariants detection in *Daikon* front ends.

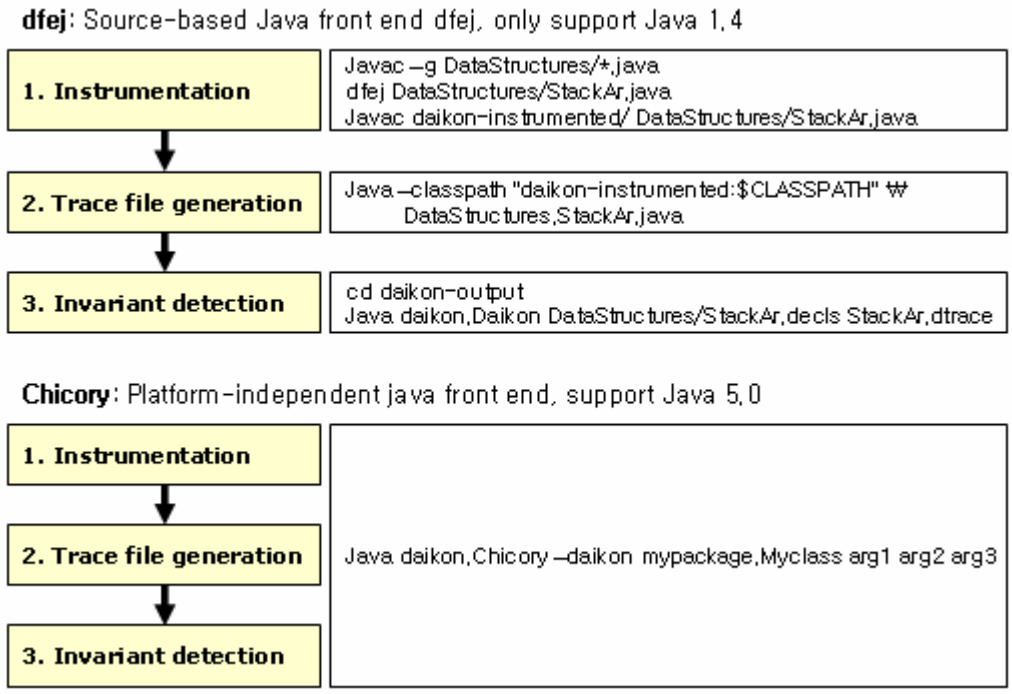

Figure 5: *Daikon* front ends and their steps

Daikon tool binaries and its documentations can be found and downloaded in the following web site:

http://pag.csail.mit.edu/*Daikon*/download/

For the test data, we prepared a 700 MB, 2 million POI data items for input. By connecting this file through JDBC-ODBC bridge driver for MS Access, we read the data records into a predefined data object by invoking its constructor. By using this configuration, we could experiment the tool.

3.2 Approach 1: *Daikon* as preprocessor in run time

3.2.1 Experiment

What we tried to do with *Daikon* is to find whether our POI data conform to the hundreds of rules(hereafter “predefined rules”) that our team and our customer came up with after analyzing the database schema and real data. For the rules that match the invariants *Daikon* found, we can regard our data as conforming to those rules without inspection. With the rules that *Daikon* found, we can decrease the number of rules with which we will inspect our POI

data. Because the time taken for inspection per rule is around 10 minutes for 2 million POI data, if the time reduced by *Daikon* is much more than the time it takes for *Daikon* to find the invariants among 2 million data, we can use *Daikon* at run time to reduce the execution time for inspection.

To find the invariants among the POI data, the first thing we did was to make a code with which we can find the invariants of the POI data. For the purpose of experimentation, we made the POI_I_COMMOM class whose instantiated object represents one row of POI_I_COMMON table which is the most important table we should inspect. In the POI_I_COMMON class, main function exists as a driver that instantiates as many objects as the number of POI data.

Instantiating objects without any additional manipulation in POI_I_COMMON class ended up finding a small number of rules that matches our predefined rules. In order to extract as many rule-related invariant as possible, we need to add some code that is related to the predefined rules. So, we made Rules class that has the boolean array which has the information about the violation of the rules. Whenever the POI_I_COMMON object is instantiated, the rules array will be filled with true or false value for each rule. For the rule that POI data conforms to, invariant will be “the boolean value of the rule will always true” whereas the invariant will be “the boolean value of the rule will be true or false” for the rules that POI data doesn’t follow.

For the rules that span one row, that is, rules that can be determined whether they are false or true with only one object, it is easy. For example, rule 1 is “column CP_ID is one of integer {1,2,3,4,6,13,15,17,21,22,23,25}”. In this case, when instantiating POI_I_COMMON object, we can check whether the object conforms to the rule 1 and set the rule 1 to false or true.

For the rules that span one table, that is, rules that can be determined after comparing all the rows in the table, we need another manipulation. For example, rule 2 is “column POI_ID is unique”. In this case, using “HashSet contains” method, if there is duplicated object, set rule 2 to false and if not, put that integer in the HashSet.

With this approach, we got the invariants we wanted. The output from the *Daikon* is as follows.

Case (record num)	Expected Invariants	Found Invariants (*Daikon* output)	Time Spent(min)
10,000	Rule 0 is true Rule 1 is true Rule 2 is true	Rules.rules[] == [1, 1, 1]	1:50
100,000	Rule 0 is true or false Rule 1 is true Rule 2 is true	Rules.rules[] one of {[0, 1, 1], [1, 1, 1]}	6:47
110,000	Rule 0 is true or false Rule 1 is true Rule 2 is true	Rules.rules[] one of {[0, 1, 1], [1, 1, 1]}	8:00
200,000	Rule 0 is true or false Rule 1 is true Rule 2 is true	Rules.rules[] one of {[0, 1, 1], [1, 1, 1]}	15:30

**** Rule 0 is "CP_ID is one of {1, 2, 3, 4, 6, 13, 15, 17, 21, 22, 23, 25}***

**** Rule 1 is "POI_DI is unique"***

**** Rule 2 is "FNAME, ADDR, ADDR2 is unique"***

Figure 6: Invariants of the approach 1

3.2.2 Analysis of the results

We found the rules that we want to find. With 10,000 POI data that conform to the rule, *Daikon* proved that it is. With 100,000 POI data that have data that don't conform to the rule 0, *Daikon* proved that rule 0 can be 0(false) or 1(true) whereas all the data conform to rule 1 and rule 2. At this time, we can inspect 100,000 POI data with only rule 1. We can reduce around 20 minutes that it takes to inspect POI data with rule 1 and rule 2 if the time taken for *Daikon* execution is less than 20 minutes.

Unfortunately, we reached the conclusion that the execution time for *Daikon* is longer than the reduction time stemming from decreased rules based on the results of *Daikon*. Just with 200,000 POI data, the time taken for *Daikon* execution was already 15 minutes. And it increases linearly with data. This approach is infeasible.

3.3 Approach 2: *Daikon* as preprocessor in design time

3.3.1 Experiment

The invariant produced by *Daikon* in Approach 1 suggests another impressive intuition. The impressive result is that we found more meaningful rules that we didn't give much attention to. Our project is mainly inspection of the wrong or illogical data. This depends on the rules we define. The more precisely defined the rule is, the cleaner the POI data after inspection are. In the course of Approach 1, we got to know that the rules can be made robust with the addition of the rules we found with *Daikon*. The following rules are those we found as additional invariants.

POI_CODE !=null
LARGE_CD !=null
FNAME !=null
PNAME !=null
ADDR !=null
BDG_FLOOR>=0
GUIDE_X1>=0
GUIDE_Y1>=0
POI_ID !=CP_ID
FNAME.toString !=PNAME.toString
CNAME.toString !=PNAME.toString
ADDR.toString != ADDR2.toString
PRIMARY_BUN==0 ➔SECONDARY_BUN==0
CENTER_X1 > CENTER_Y1
CENTER_X1 > GUIDE_Y1
CENTER_Y1 != GUIDE_X1
GUIDE_X1==0 ➔ GUIDE_Y1==0
GUIDE_Y1==0 ➔ GUIDE_X1==0

Figure 7: Interesting invariants (rules) of the approach 2

3.3.2 Analysis of the results

This brings us to approach 2. We can use *Daikon* in our project as a way of making our rules robust at design time. For example, PRIMARY_BUN→SECONDAY_BUN means that if the first street number is 0, then the second street number is 0. The street number is

composed of two fields in South Korea. If we do not know the street number with the current information, we are supposed to set the first street number and the second street number to 0. This is the rule that we did not consider as rule before we discovered this with *Daikon*.

4. Lessons Learned

Through this project, we got lessons about the characteristics of dynamic analysis engine as well as its value. In addition, we got to know the way we can get benefits by applying the tool in our project. Finally, we learned the dynamic analysis tool could be used not only for the application code invariant capturing, but also for data rule capturing.

4.1 General characteristics of *Daikon*

We found several characteristics of *Daikon* during this tool evaluation project.

First, we learned the characteristics of *Daikon*. This tool is a runtime learning machine that can discover more precise invariants as we use more test cases. However, there is still limitation: when we put test cases at some limit of time, it does not result better anymore. In our case, when we put 10 thousands records as test cases, we got some amount of invariants including spurious ones. After testing with 100 thousands records, we got a little more invariants with less spurious ones. However, when we tried with 200 thousands, there were no more invariants. Therefore, we can see the learning engine has some limitation.

Here is another characteristic of the tool. Basically, the tool can detect the invariants of application codes by running the codes with provided test cases. Therefore, the quality of the resulting invariants is dependant on the quality of the input test cases.

Another characteristic of the tool is the that the tool is very sensitive to the value of variables in the code. Especially the point of initialization is important to detect the invariants related to the variables. For example, if we set instance variable '*foo*' of a class in its constructor, the variable's invariant can be 'not null'. Otherwise, we cannot get the invariant even if the variable is always not null except for the time it was initialized. Therefore, if the invariant is important, we should initialize the variables inside constructor.

4.2 Benefits to our project

Generally, the tool is to be used to gather invariants in application code. Different from the general way of using the tool, we tried to apply it for gathering the rules inside the database. As a result, we found some beneficial way of using the tool in our Studio project. We can use the tool to find the rules of inspection we might have missed.

Actually, we tried to use the tool at runtime to filter the rules. However, this trial showed less valuable results. The time spent in running the tool engine was significantly larger to get data than the time just using hand-made checking code running. Therefore, our first trial to apply this tool to our project was not fruitful.

Our second approach was to use the tool at design time rather than at runtime. By using the tool at design time we could exploit the tool's ability to improve the quality of our rule set. We found new valuable rules. In addition, we will use the invariants to validate the correctness of the existing rules. For this purpose, the tool was significantly beneficial to our actual Studio project.

4.3 Drawbacks

Most of all, there were too many spurious invariants found, so it is required to put extra effort to filtering the meaningless invariants out from the entire results.

Next, it is required to put more efforts to create proper test cases. This is overhead. This overhead becomes heavier when the tool is applied to application code invariant checking rather than data checking like our application. Moreover, the tool also needs test data to be prepared by users to drive the invariant capturing. These overhead is not trivial, so maybe sometimes the efforts input are less than the benefits of its output.

Last, in case of data centric application such as our POI data inspection system, the comparison is required between *Daikon* based approach and traditional SQL (Sequential Query Language) query based approach in terms of the performance, implementation difficulty, and reusability for other application. In general cases, traditional SQL query based approach has better performance and easy buildability.

5. Conclusion

In our approach, *Daikon* has not helpful to verify the inspection program and extract clear variants from POI database. There are several problems.

First, *Daikon* does not extract sufficient variants from POI database directly. We tried to detect the invariants several times by changing the instrumentation and input options, but it did not work in our cases. In addition, it is too expensive approach for just verification of the inspection application.

Second, *Daikon* has poor performance to extract invariants from the data items in database. Mostly, it comes from the data output processing in the database. Because our target POI database has huge data items, the throughput of the processing is quite low.

However, there are several beneficial things in *Daikon* when applied to our project. Most of all, *Daikon* can be used to refine the inspection rule from POI data. During the review of the generated invariants, we found meaningful inspection rules from the invariants that we did not catch before. Even though, we were struggle against the many insignificant invariants, we could found meaningful inspection rule from the invariants.

Another possible area to adopt *Daikon* in our studio program is the verification of the inspection and cleansing program. Especially, by dynamically discovering invariants, we believe *Daikon* will be a good tool to verify the application. Because our problem domain has huge data to handle, the dynamic approach is required for the performance.

6. Reference

[1] Michael D.Ernst, “Dynamically Detecting Likely Program Invariants,” PhD Disertation, University of Washington, August 2000.

[2] Michael D.Ernst, http://pag.csail.mit.edu/*Daikon*

[3] *Daikon* Invariant Detect User Manual

[4] Michael D.Ernst, Jake Cockrell, “Dynamically Discovering Likely Program Invariants to Support Program Evolution,” IEEE Transaction on Software Eng., vol. 27, no.2 Feb. 2001.

Appendix 1. POI_I_COMMON Table Specification of POI database

Column ID	Description	Primary Key	NULL	Data Type	Length
POI_ID	POI Sequential Code	PK	N	Number	Long
CP_ID	Data Provider			Number	Long
POI_CODE	POI Category			Text	6
LARGE_CD	Top level Administration code			Text	2
MIDDLE_CD	Middle level Administration code			Text	3
SMALL_CD	Bottom level Administration code			Text	5
MMS_CODE	MMS Administration code			Text	10
FNAME	Formal Name			Text	100
ENAME	English Name			Text	100
ANAME	Alternative Name			Text	100
CNAME	Branch Name			Text	100
PNAME	Search Name			Text	100
mZIP_CODE	Zip code			Text	7
ADDR	Address 1			Text	100
ADDR2	Address 2			Text	50
PRIMARY_BUN	House Number 1			Number	Long
SECONDARY_BUN	House Number 2			Number	Long
SAN_BUN	Mountain Number			Text	1
IFLOOR	Floor			Text	9
TELE_A	Telephone Number			Text	4
TELE_B	Telephone Number			Text	4
TELE_C	Telephone Number			Text	4
TELE_D	Telephone Number			Text	1
BUSINESS_NO	License Number			Text	12
ROAD	Road Name			Memo	
NBDG_NAME	Building Name			Text	50
BDG_FLOOR	Building Story			Number	Long
TILE_ID	Map ID			Number	Long
CENTER_X1	X coordinate		N	Number	Long

CENTER_Y1	Y coordinate		N	Number	Long
GUIDE_X1	Guide X coordinate			Number	Long
GUIDE_Y1	Guide Y coordinate			Number	Long
IS_DGUIDE	Driver Guide			Text	1
TARGET_POI_ID	Target POI_ID			Number	Long
POI_KIND	POI Category			Text	1
INSERT_DATE	Creation date			Text	50
UPDATE_DATE	Change date			Text	50
KIND_CODE	Document Category			Text	5
BIGO	Remark			Text	1

Appendix 2. Inspection Rule related to POI_I_COMMON Table

Rule ID	Rule Description	Related Table	Error Type Description
R01	POI_ID is unique		Uniqueness Violation
R02	CP_ID has a restricted code set	Self Contained Code	Reference Violation
R03	POI_CODE has a reference column	POI_C _CLASS	Reference Violation
R04	LARGE_CD has a hierarchy structure	Government Code	Hierarchy data reference Error
R05	MIDDLE_CD has a parent and children	Government Code	Hierarchy data reference Error
R06	SMALL_CD has a parent	Government Code	Hierarchy data reference Error
R07	MMS_CODE has a reference column		Reference Violation
R08	FNAME + ADDR + ADDR2 is unique		FNAME + ADDR + ADDR2 Duplication
R09	FNAME should not contain any special character set		Special Character Error
R10	ENAME + ADDR + ADDR2 is unique		ENAME + ADDR + ADDR2 Duplication
R11	ENAME should not contain any special character set		Special Character Error
R12	ANAME + ADDR + ADDR2 is unique		ANAME + ADDR + ADDR2 Duplication
R13	ANAME should not contain any special character set		Special Character Error
R14	CNAME + ADDR + ADDR2 is unique		CNAME + ADDR + ADDR2 Duplication
R15	CNAME should not contain any special character set		Special Character Error
R16	PNAME + ADDR + ADDR2 is unique		PNAME + ADDR + ADDR2 Duplication
R17	PNAME should not contain any special character set		Special Character Error
R18	mZIP_CODE has own code structure including "-" character		Internal data logic inconsistency
R19	mZIP_CODE should not contain any special character set		Special Character Error
R20	ADDR should not equl ADDR2		Internal data logic inconsistency
R21	Special character set are not allowed in ADDR2		Special Character Error
R22	Length of PRIMARY_BUN is greater than four.(Exception: length of cheju <= 4)		Internal data logic inconsistency

R23	SECONDARY_BUN cannot exists without PRIMARY_BUN		Internal data logic inconsistency
R24	SAN_BUN has a restricted code set	1,0	Reference Violation
R25	IFLOOR has own code naming rule(eg:B001-F005)		Internal data logic inconsistency
R26	TELE_A has a restricted code set. Only Number is allowed	Area Code	Reference Violation
R27	TELE_B should consist of only number		Character Type Violation
R28	TELE_C should consist of only number		Character Type Violation
R29	TELE_D should consist of only number		Character Type Violation
R30	BUSINESS_NO+FNAME is unique(Candidate Key)		BUSINESS_NO+FNAME Duplication
R31	"Enter" key is not permitted in ROAD which is "memo type"		Special Character Error
R32	BDG_FLOOR is greater or equal than IFLOOR		Internal data logic inconsistency
R33	TILE_ID+CENTER_x1+CENTER_Y1+mZIP_CODE is unique		TILE_ID+CENTER_x1+CENTER_Y1+mZIP_CODE Duplication
R34	CENTER_X1+CENTER_Y1+GUIDE_X1+GUIDE_Y1 is unique		CENTER_X1+CENTER_Y1+GUIDE_X1+GUIDE_Y1 Duplication
R35	CENTER_X1 range is restricted(within Korean Penninsula)		Internal data logic inconsistency
R36	CENTER_Y1 range is restricted(within Korean Penninsula)		Internal data logic inconsistency
R37	GUIDE_X1 range is restricted(within Korean Penninsula)		Internal data logic inconsistency
R38	GUIDE_Y4 range is restricted(within Korean Penninsula)		Internal data logic inconsistency
R39	IS_DGUIDE has a restricted code set	1,0	Reference Violation
R40	TARGET_POI_ID has a self-reference column	Recursive reference	Reference Violation
R41	POI_KIND has a restricted code set	Self Contained Code	Reference Violation
R42	INSERT_DATE shoulbe date type		Attribute Type Violation
R43	UPDATE_DATE shoulbe date type		Attribute Type Violation

R44	KIND_CODE has a restricted code set	Self Contained Code	Reference Violation
R45	BIGO has a restricted code set	Self Contained Code	Reference Violation

Appendix 3. Detected Invariants

Bold red line is the meaningful invariant that was not in the predefined rules.

```
Buildfile: C:\eclipse\workspace\DaikonPrototype\build.xml
printinv:
    [java] ==============================================================================
    [java] prototype.ClassForRule2:::OBJECT
    [java] this.FNAME != null
    [java] this.ADDR != null
    [java] this.FNAME.toString != this.ADDR.toString
    [java] this.FNAME.toString != this.ADDR2.toString
    [java] this.ADDR.toString != this.ADDR2.toString
    [java] ==============================================================================
    [java]      prototype.ClassForRule2.ClassForRule2(java.lang.String,      java.lang.String,
java.lang.String):::ENTER
    [java] name != null
    [java] addr != null
    [java] name.toString != addr.toString
    [java] name.toString != addr2.toString
    [java] addr.toString != addr2.toString
    [java] ==============================================================================
    [java]      prototype.ClassForRule2.ClassForRule2(java.lang.String,      java.lang.String,
java.lang.String):::EXIT
    [java] this.FNAME == orig(name)
    [java] this.ADDR == orig(addr)
    [java] this.ADDR2 == orig(addr2)
    [java] name.toString != addr.toString
    [java] name.toString != addr2.toString
    [java] name.toString == this.FNAME.toString
    [java] name.toString == orig(name.toString)
    [java] addr.toString != addr2.toString
    [java] addr.toString == this.ADDR.toString
    [java] addr.toString == orig(addr.toString)
    [java] addr2.toString == this.ADDR2.toString
    [java] addr2.toString == orig(addr2.toString)
    [java] ==============================================================================
    [java] prototype.ClassForRule2.equals(java.lang.Object):::ENTER
    [java] object != null
    [java] object.class == "prototype.ClassForRule2"
    [java] this.ADDR2.toString == ""
    [java] this.FNAME.toString > this.ADDR2.toString
    [java] this.ADDR.toString > this.ADDR2.toString
    [java] ==============================================================================
    [java] prototype.ClassForRule2.equals(java.lang.Object):::EXIT37
    [java] return == false
    [java] this.ADDR2 == null
    [java] ==============================================================================
    [java]  prototype.ClassForRule2.equals(java.lang.Object):::EXIT37;condition="not(return  ==
true)"
    [java] ==============================================================================
    [java] prototype.ClassForRule2.equals(java.lang.Object):::EXIT39
    [java] return == false
    [java] this.ADDR2 != null
    [java] ==============================================================================
    [java]  prototype.ClassForRule2.equals(java.lang.Object):::EXIT39;condition="not(return  ==
true)"
    [java] ==============================================================================
    [java] prototype.ClassForRule2.equals(java.lang.Object):::EXIT40
    [java] return == true
    [java] this.ADDR2 == null
    [java] ==============================================================================
    [java]    prototype.ClassForRule2.equals(java.lang.Object):::EXIT40;condition="return    ==
true"
    [java] ==============================================================================
```

```
    [java] prototype.ClassForRule2.equals(java.lang.Object):::EXIT
    [java] this.FNAME == orig(this.FNAME)
    [java] this.ADDR == orig(this.ADDR)
    [java] this.ADDR2 == orig(this.ADDR2)
    [java] (return == false)  ==>  (this.ADDR.toString > this.ADDR2.toString)
    [java] (return == false)  ==>  (this.ADDR2.toString == "")
    [java] (return == false)  ==>  (this.FNAME.toString > this.ADDR2.toString)
    [java] (return == true)  ==>  (this.ADDR2 == null)
    [java] this.ADDR2.toString == ""
    [java] this.FNAME.toString > this.ADDR2.toString
    [java] this.FNAME.toString == orig(this.FNAME.toString)
    [java] this.ADDR.toString > this.ADDR2.toString
    [java] this.ADDR.toString == orig(this.ADDR.toString)
    [java] this.ADDR2.toString == orig(this.ADDR2.toString)
    [java] ==========================================================================
    [java] prototype.ClassForRule2.equals(java.lang.Object):::EXIT;condition="return == true"
    [java] return == true
    [java] this.ADDR2 == null
    [java] ==========================================================================
    [java]   prototype.ClassForRule2.equals(java.lang.Object):::EXIT;condition="not(return   ==
true)"
    [java] return == false
    [java] ==========================================================================
    [java] prototype.ClassForRule2.hashCode():::ENTER
    [java] ==========================================================================
    [java] prototype.ClassForRule2.hashCode():::EXIT
    [java] this.FNAME == orig(this.FNAME)
    [java] this.ADDR == orig(this.ADDR)
    [java] this.ADDR2 == orig(this.ADDR2)
    [java] return != 0
    [java] this.FNAME.toString == orig(this.FNAME.toString)
    [java] this.ADDR.toString == orig(this.ADDR.toString)
    [java] this.ADDR2.toString == orig(this.ADDR2.toString)
    [java] ==========================================================================
    [java] prototype.POI_I_COMMON:::OBJECT
    [java] this.CP_ID >= 2
    [java] this.POI_CODE != null
    [java] this.LARGE_CD != null
    [java] this.FNAME != null
    [java] this.PNAME != null
    [java] this.ADDR != null
    [java] this.PRIMARY_BUN >= 0
    [java] this.SECONDARY_BUN >= 0
    [java] this.BDG_FLOOR >= 0
    [java] this.GUIDE_X1 >= 0
    [java] this.GUIDE_Y1 >= 0
    [java] this.POI_ID != this.CP_ID
    [java] this.POI_ID != this.SECONDARY_BUN
    [java] this.POI_ID < this.CENTER_X1
    [java] this.POI_ID < this.CENTER_Y1
    [java] this.POI_ID != this.GUIDE_X1
    [java] this.POI_ID != this.GUIDE_Y1
    [java] this.CP_ID < this.CENTER_X1
    [java] this.CP_ID < this.CENTER_Y1
    [java] this.CP_ID != this.GUIDE_X1
    [java] this.CP_ID != this.GUIDE_Y1
    [java] this.POI_CODE.toString < this.LARGE_CD.toString
    [java] this.POI_CODE.toString != this.FNAME.toString
    [java] this.POI_CODE.toString != this.CNAME.toString
    [java] this.POI_CODE.toString != this.PNAME.toString
    [java] this.POI_CODE.toString < this.ZIP_CODE.toString
    [java] this.POI_CODE.toString < this.ADDR.toString
    [java] this.POI_CODE.toString != this.ADDR2.toString
    [java] this.POI_CODE.toString < this.iFLOOR.toString
    [java] this.LARGE_CD.toString != this.FNAME.toString
    [java] this.LARGE_CD.toString != this.CNAME.toString
    [java] this.LARGE_CD.toString != this.PNAME.toString
    [java] this.LARGE_CD.toString != this.ZIP_CODE.toString
    [java] this.LARGE_CD.toString < this.ADDR.toString
```

```
    [java] this.LARGE_CD.toString != this.ADDR2.toString
    [java] this.LARGE_CD.toString < this.iFLOOR.toString
    [java] this.FNAME.toString != this.PNAME.toString
    [java] this.FNAME.toString != this.ZIP_CODE.toString
    [java] this.FNAME.toString != this.ADDR.toString
    [java] this.FNAME.toString != this.ADDR2.toString
    [java] this.FNAME.toString != this.iFLOOR.toString
    [java] this.CNAME.toString != this.PNAME.toString
    [java] this.CNAME.toString != this.ZIP_CODE.toString
    [java] this.CNAME.toString != this.ADDR.toString
    [java] this.CNAME.toString != this.ADDR2.toString
    [java] this.CNAME.toString != this.iFLOOR.toString
    [java] this.PNAME.toString != this.ZIP_CODE.toString
    [java] this.PNAME.toString != this.ADDR.toString
    [java] this.PNAME.toString != this.ADDR2.toString
    [java] this.PNAME.toString != this.iFLOOR.toString
    [java] this.ZIP_CODE.toString < this.ADDR.toString
    [java] this.ZIP_CODE.toString != this.ADDR2.toString
    [java] this.ZIP_CODE.toString < this.iFLOOR.toString
    [java] this.ADDR.toString != this.ADDR2.toString
    [java] this.ADDR.toString > this.iFLOOR.toString
    [java] this.ADDR2.toString != this.iFLOOR.toString
    [java] (this.PRIMARY_BUN == 0) ==> (this.SECONDARY_BUN == 0)
    [java] this.PRIMARY_BUN < this.CENTER_X1
    [java] this.PRIMARY_BUN < this.CENTER_Y1
    [java] this.SECONDARY_BUN < this.CENTER_X1
    [java] this.SECONDARY_BUN < this.CENTER_Y1
    [java] this.BDG_FLOOR < this.CENTER_X1
    [java] this.BDG_FLOOR < this.CENTER_Y1
    [java] this.CENTER_X1 > this.CENTER_Y1
    [java] this.CENTER_X1 > this.GUIDE_Y1
    [java] this.CENTER_Y1 != this.GUIDE_X1
    [java] (this.GUIDE_X1 == 0) ==> (this.GUIDE_Y1 == 0)
    [java] (this.GUIDE_Y1 == 0) ==> (this.GUIDE_X1 == 0)
    [java] this.GUIDE_X1 >= this.GUIDE_Y1
    [java] ================================================================================
    [java] prototype.POI_I_COMMON.POI_I_COMMON(int, int, java.lang.String, java.lang.String,
java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String,
java.lang.String, int, int, java.lang.String, int, int, int, int, int):::ENTER
    [java] cp_id >= 2
    [java] poi_code != null
    [java] large_cd != null
    [java] fname != null
    [java] pname != null
    [java] addr != null
    [java] primary_bun >= 0
    [java] secondary_bun >= 0
    [java] bdg_floor >= 0
    [java] guide_x1 >= 0
    [java] guide_y1 >= 0
    [java] poi_id != cp_id
    [java] poi_id != secondary_bun
    [java] poi_id < center_x1
    [java] poi_id < center_y1
    [java] poi_id != guide_x1
    [java] poi_id != guide_y1
    [java] cp_id < center_x1
    [java] cp_id < center_y1
    [java] cp_id != guide_x1
    [java] cp_id != guide_y1
    [java] poi_code.toString < large_cd.toString
    [java] poi_code.toString != fname.toString
    [java] poi_code.toString != cname.toString
    [java] poi_code.toString != pname.toString
    [java] poi_code.toString < zip_code.toString
    [java] poi_code.toString < addr.toString
    [java] poi_code.toString != addr2.toString
    [java] poi_code.toString < iFloor.toString
    [java] large_cd.toString != fname.toString
```

```
    [java] large_cd.toString != cname.toString
    [java] large_cd.toString != pname.toString
    [java] large_cd.toString != zip_code.toString
    [java] large_cd.toString < addr.toString
    [java] large_cd.toString != addr2.toString
    [java] large_cd.toString < iFloor.toString
    [java] fname.toString != pname.toString
    [java] fname.toString != zip_code.toString
    [java] fname.toString != addr.toString
    [java] fname.toString != addr2.toString
    [java] fname.toString != iFloor.toString
    [java] cname.toString != pname.toString
    [java] cname.toString != zip_code.toString
    [java] cname.toString != addr.toString
    [java] cname.toString != addr2.toString
    [java] cname.toString != iFloor.toString
    [java] pname.toString != zip_code.toString
    [java] pname.toString != addr.toString
    [java] pname.toString != addr2.toString
    [java] pname.toString != iFloor.toString
    [java] zip_code.toString < addr.toString
    [java] zip_code.toString != addr2.toString
    [java] zip_code.toString < iFloor.toString
    [java] addr.toString != addr2.toString
    [java] addr.toString > iFloor.toString
    [java] addr2.toString != iFloor.toString
    [java] (primary_bun == 0) ==> (secondary_bun == 0)
    [java] primary_bun < center_x1
    [java] primary_bun < center_y1
    [java] secondary_bun < center_x1
    [java] secondary_bun < center_y1
    [java] bdg_floor < center_x1
    [java] bdg_floor < center_y1
    [java] center_x1 > center_y1
    [java] center_x1 > guide_y1
    [java] center_y1 != guide_x1
    [java] (guide_x1 == 0) ==> (guide_y1 == 0)
    [java] (guide_y1 == 0) ==> (guide_x1 == 0)
    [java] guide_x1 >= guide_y1
    [java] ================================================================================
    [java]  prototype.POI_I_COMMON.POI_I_COMMON(int,  int,  java.lang.String,  java.lang.String,
java.lang.String,  java.lang.String,  java.lang.String,  java.lang.String,  java.lang.String,
java.lang.String, int, int, java.lang.String, int, int, int, int, int):::EXIT
    [java] this.POI_ID == orig(poi_id)
    [java] this.CP_ID == orig(cp_id)
    [java] this.POI_CODE == orig(poi_code)
    [java] this.LARGE_CD == orig(large_cd)
    [java] this.FNAME == orig(fname)
    [java] this.CNAME == orig(cname)
    [java] this.PNAME == orig(pname)
    [java] this.ZIP_CODE == orig(zip_code)
    [java] this.ADDR == orig(addr)
    [java] this.ADDR2 == orig(addr2)
    [java] this.PRIMARY_BUN == orig(primary_bun)
    [java] this.SECONDARY_BUN == orig(secondary_bun)
    [java] this.iFLOOR == orig(iFloor)
    [java] this.BDG_FLOOR == orig(bdg_floor)
    [java] this.CENTER_X1 == orig(center_x1)
    [java] this.CENTER_Y1 == orig(center_y1)
    [java] this.GUIDE_X1 == orig(guide_x1)
    [java] this.GUIDE_Y1 == orig(guide_y1)
    [java] poi_code.toString < large_cd.toString
    [java] poi_code.toString != fname.toString
    [java] poi_code.toString != cname.toString
    [java] poi_code.toString != pname.toString
    [java] poi_code.toString < zip_code.toString
    [java] poi_code.toString < addr.toString
    [java] poi_code.toString != addr2.toString
    [java] poi_code.toString < iFloor.toString
    [java] poi_code.toString == this.POI_CODE.toString
```

```
[java] poi_code.toString == orig(poi_code.toString)
[java] large_cd.toString != fname.toString
[java] large_cd.toString != cname.toString
[java] large_cd.toString != pname.toString
[java] large_cd.toString != zip_code.toString
[java] large_cd.toString < addr.toString
[java] large_cd.toString != addr2.toString
[java] large_cd.toString < iFloor.toString
[java] large_cd.toString == this.LARGE_CD.toString
[java] large_cd.toString == orig(large_cd.toString)
[java] fname.toString != pname.toString
[java] fname.toString != zip_code.toString
[java] fname.toString != addr.toString
[java] fname.toString != addr2.toString
[java] fname.toString != iFloor.toString
[java] fname.toString == this.FNAME.toString
[java] fname.toString == orig(fname.toString)
[java] cname.toString != pname.toString
[java] cname.toString != zip_code.toString
[java] cname.toString != addr.toString
[java] cname.toString != addr2.toString
[java] cname.toString != iFloor.toString
[java] cname.toString == this.CNAME.toString
[java] cname.toString == orig(cname.toString)
[java] pname.toString != zip_code.toString
[java] pname.toString != addr.toString
[java] pname.toString != addr2.toString
[java] pname.toString != iFloor.toString
[java] pname.toString == this.PNAME.toString
[java] pname.toString == orig(pname.toString)
[java] zip_code.toString < addr.toString
[java] zip_code.toString != addr2.toString
[java] zip_code.toString < iFloor.toString
[java] zip_code.toString == this.ZIP_CODE.toString
[java] zip_code.toString == orig(zip_code.toString)
[java] addr.toString != addr2.toString
[java] addr.toString > iFloor.toString
[java] addr.toString == this.ADDR.toString
[java] addr.toString == orig(addr.toString)
[java] addr2.toString != iFloor.toString
[java] addr2.toString == this.ADDR2.toString
[java] addr2.toString == orig(addr2.toString)
[java] iFloor.toString == this.iFLOOR.toString
[java] iFloor.toString == orig(iFloor.toString)
[java] ==========================================================================
[java] prototype.POI_I_COMMON.main(java.lang.String[]):::ENTER
[java] args has only one value
[java] args.class == "java.lang.String[]"
[java] args[] == []
[java] ==========================================================================
[java] prototype.POI_I_COMMON.main(java.lang.String[]):::EXIT
[java] args[] == []
[java] ==========================================================================
[java] prototype.Rules:::CLASS
[java] prototype.Rules.rules has only one value
[java] prototype.Rules.rules != null
[java] prototype.Rules.rules[] one of { [0, 1, 1], [1, 1, 1] }
[java] prototype.Rules.poi_id_DuplicationCheck has only one value
[java] prototype.Rules.poi_id_DuplicationCheck != null
[java] prototype.Rules.rule2 has only one value
[java] prototype.Rules.rule2 != null
[java] size(prototype.Rules.rules[]) == 3
[java] ==========================================================================
[java] prototype.Rules.containsCPID(int):::ENTER
[java] cp_id >= 2
[java] cp_id >= size(prototype.Rules.rules[])-1
[java] ==========================================================================
[java] prototype.Rules.containsCPID(int):::EXIT37
[java] return == true
[java] orig(cp_id) one of { 2, 3, 25 }
```

```
    [java] return in prototype.Rules.rules[]
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT37;condition="return == true"
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT39
    [java] return == false
    [java] prototype.Rules.rules[] == [1, 1, 1]
    [java] prototype.Rules.rules[] elements == true
    [java] orig(cp_id) == 5
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT39;condition="not(return == true)"
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT
    [java] prototype.Rules.rules == orig(prototype.Rules.rules)
    [java] prototype.Rules.rules[] == orig(prototype.Rules.rules[])
    [java]                       prototype.Rules.poi_id_DuplicationCheck                       ==
orig(prototype.Rules.poi_id_DuplicationCheck)
    [java] prototype.Rules.rule2 == orig(prototype.Rules.rule2)
    [java] (return == false)  <==>  (orig(cp_id) == 5)
    [java] (return == false)  ==>  (prototype.Rules.rules[] == [1, 1, 1])
    [java] (return == false)  ==>  (prototype.Rules.rules[] elements == true)
    [java] (return == true)  <==>  (orig(cp_id) one of { 2, 3, 25 })
    [java] (return == true)  ==>  (prototype.Rules.rules[] one of { [0, 1, 1], [1, 1, 1] })
    [java] (return == true)  ==>  (return in prototype.Rules.rules[])
    [java] orig(cp_id) >= size(prototype.Rules.rules[])-1
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT;condition="return == true"
    [java] return == true
    [java] orig(cp_id) one of { 2, 3, 25 }
    [java] return in prototype.Rules.rules[]
    [java] ================================================================================
    [java] prototype.Rules.containsCPID(int):::EXIT;condition="not(return == true)"
    [java] return == false
    [java] prototype.Rules.rules[] == [1, 1, 1]
    [java] prototype.Rules.rules[] elements == true
    [java] orig(cp_id) == 5
    [java] ================================================================================
    [java] prototype.Rules.uniqueness(java.lang.Integer):::ENTER
    [java] a != null
    [java] ================================================================================
    [java] prototype.Rules.uniqueness(java.lang.Integer):::EXIT30
    [java] ================================================================================
    [java] prototype.Rules.uniqueness(java.lang.Integer):::EXIT30;condition="return == true"
    [java] ================================================================================
    [java] prototype.Rules.uniqueness(java.lang.Integer):::EXIT
    [java] prototype.Rules.rules == orig(prototype.Rules.rules)
    [java] prototype.Rules.rules[] == orig(prototype.Rules.rules[])
    [java]                       prototype.Rules.poi_id_DuplicationCheck                       ==
orig(prototype.Rules.poi_id_DuplicationCheck)
    [java] prototype.Rules.rule2 == orig(prototype.Rules.rule2)
    [java] return == true
    [java] return in prototype.Rules.rules[]
    [java] ================================================================================
    [java] prototype.Rules.uniqueness(java.lang.Integer):::EXIT;condition="return == true"
BUILD SUCCESSFUL
Total time: 7 seconds
```