

ASSESSING THE IMPACT OF FRAMEWORK CHANGES USING COMPONENT RANKING

Reishi Yokomori

Nanzan University, Japan

Harvey Siy

University of Nebraska at Omaha, USA

Masami Noro

Nanzan University, Japan

Katsuro Inoue

Osaka University, Japan

BACKGROUND

- Most of today's software applications are built on libraries or frameworks.
 - Not only the application, but also framework grow during the development.
 - Research question:
 - How do developers work through their evolution?
 - We would like to know how to assess the impact of framework changes.
 - In this study, we analyze the evolution of software based on use relationship between components.




PURPOSE OF STUDY

We analyzed in an actual open source project;

- To find characteristics for the evolution of use relation between **framework components** and **application software**.
- To show that the analysis of use relation is a viable analysis methodology for studying software evolution.

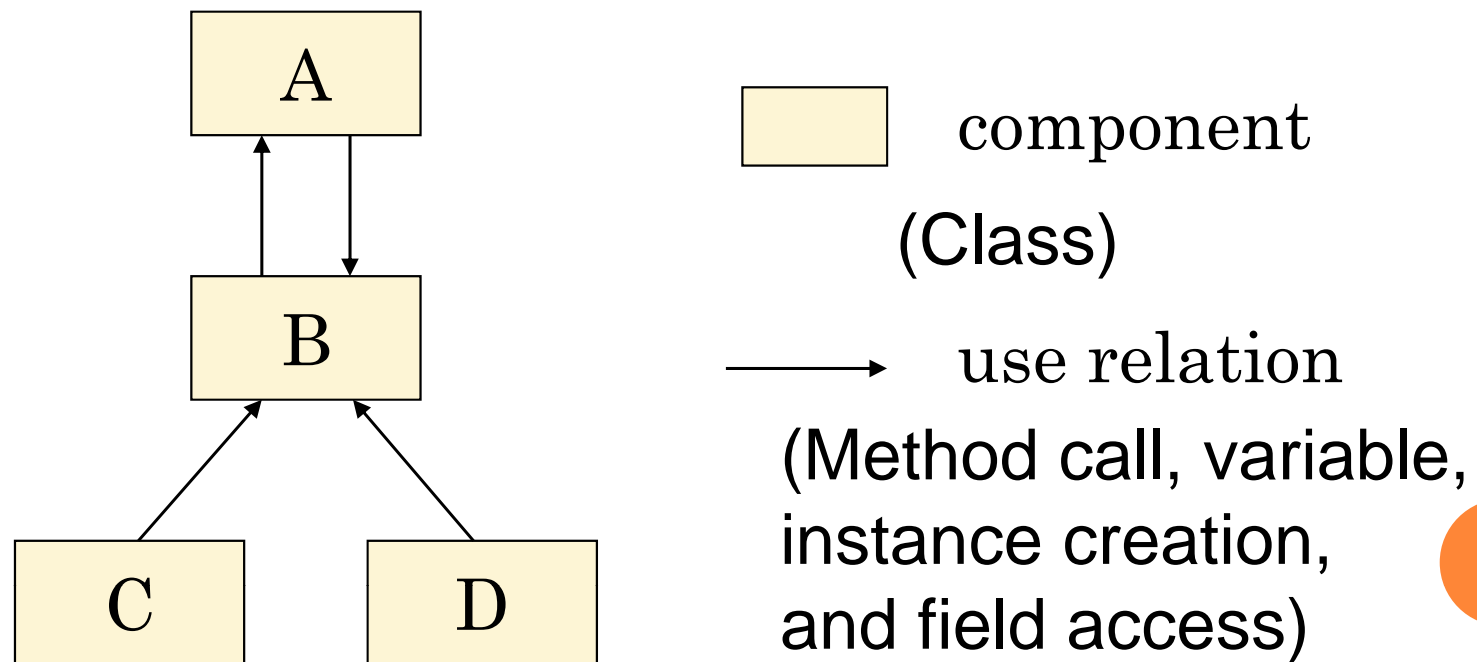


TARGET SOFTWARE

- We will analyze the evolution of use relation between framework and application.
 - The target framework: **JHotDraw**
 - a Java-based GUI framework for technical and structured graphics.
 - The target application: **JARP**
 - a Java-based Petri net tool
 - It uses **JHotDraw** as a framework for editing a Petri net, drawing the result, and so on.
- We will analyze use relation based on **component graph** and **component rank**. 

COMPONENT GRAPH

- It models use relations in software.
 - **Nodes : component**
 - **Edges : use relation**
 - incoming and outgoing edges for each component

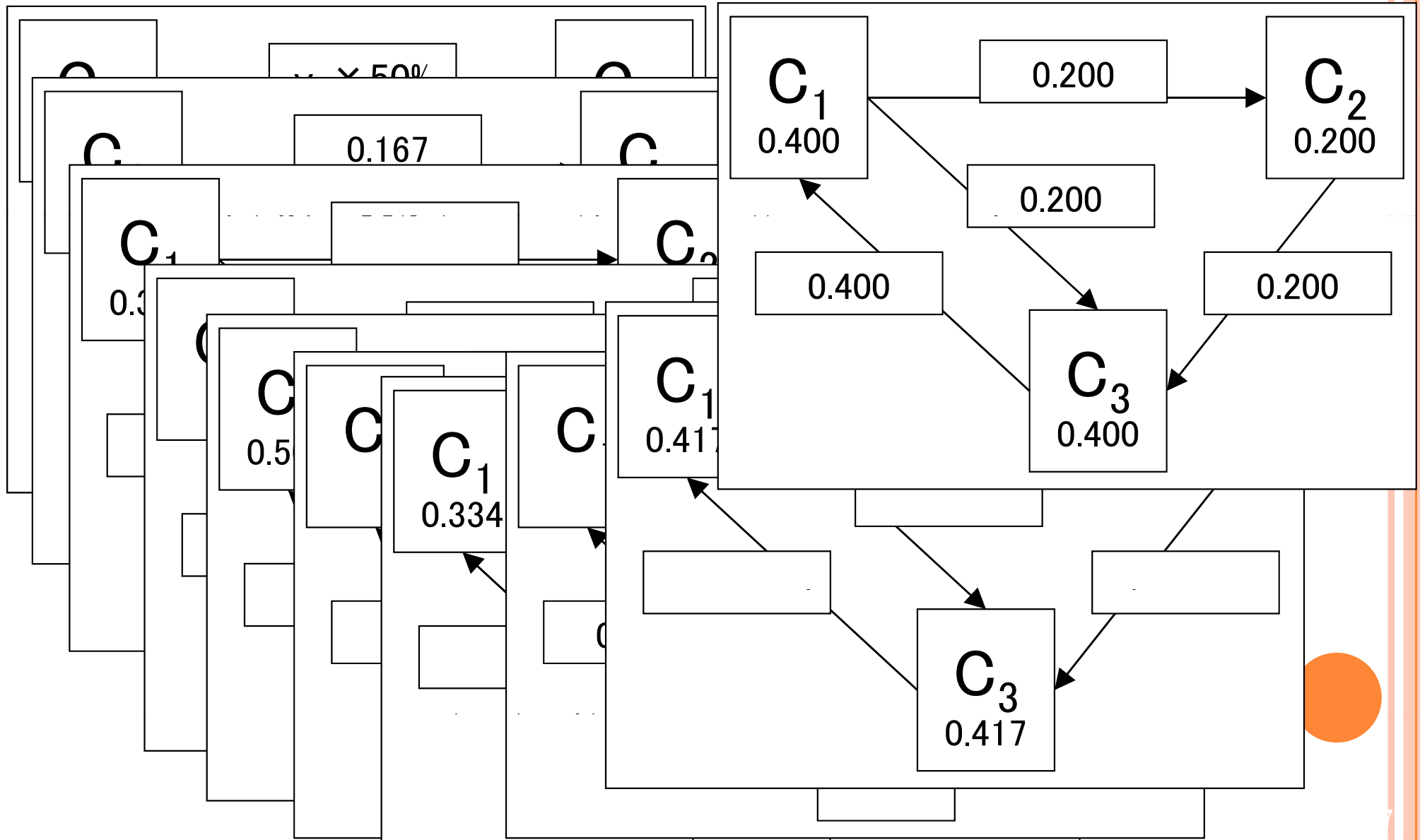


COMPONENT RANK

- is a ranking method for components.
 - How much component is used?
- is based on use relation between components.
 - If the component is frequently used, its rank goes up.
 - Both direct and indirect use relation are taken into consideration.
- is calculated from the component graph.
 - Value of each component is a steady-state distribution on a Markov chain.
 - Components are sorted by the value of each component.



CALCULATION PROCESS OF COMPONENT RANK



METRICS FOR ANALYSIS

- Edges from **application** component to **framework** component
 - **Incoming edges** to each **framework** component
 - **Outgoing edges** from each **application** component
- **Component rank**



RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



GENERAL INFORMATION

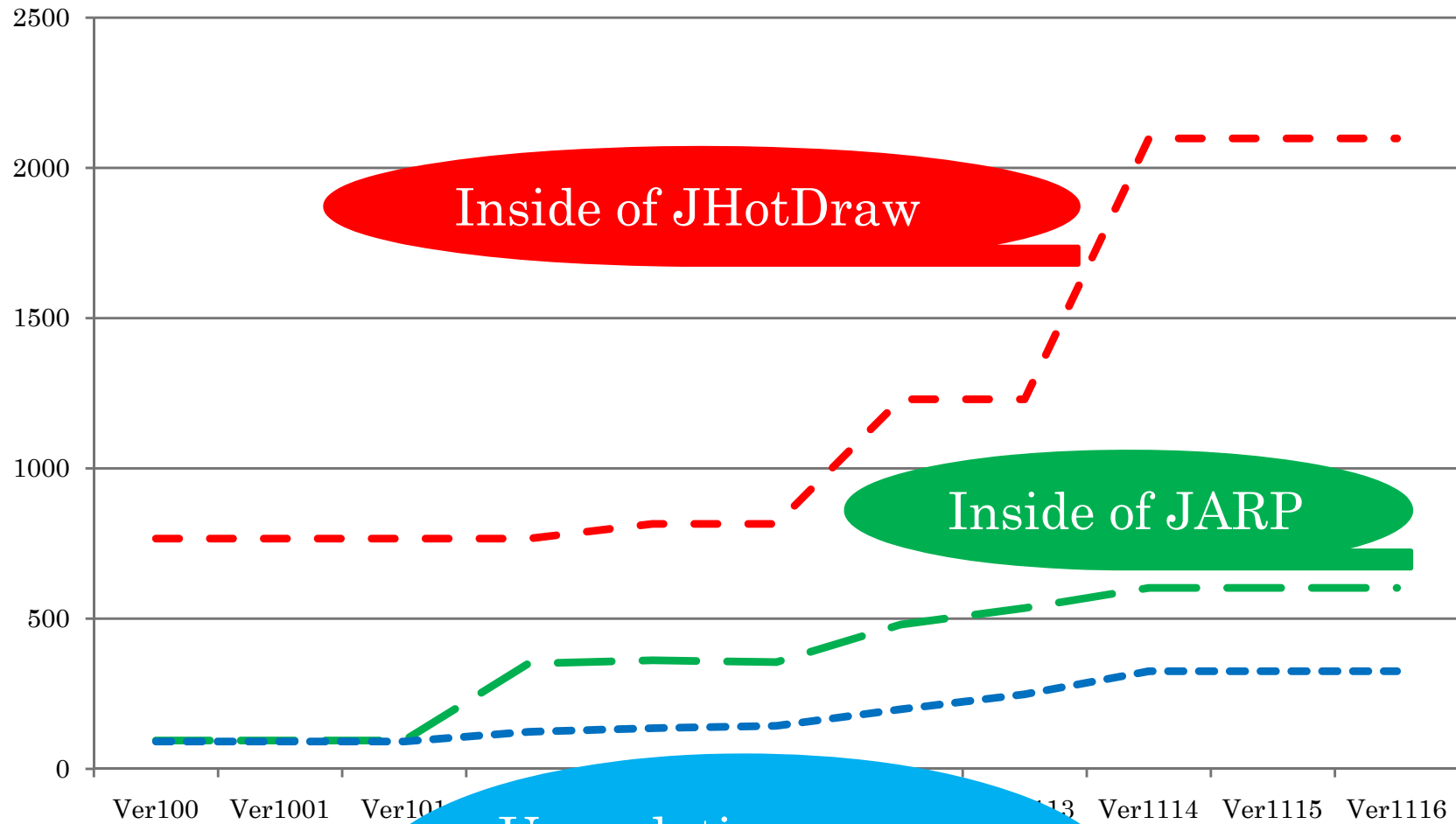
- JARP has 11 versions.

Table 2. The history of JARP

	Versions	Date	JHD	Class	LOC
1	1.0.0	2001/1/21	5.1	196(41+155)	23K
2	1.0.0.1	2001/1/26	5.1	196(41+155)	23K
3	1.0.1	2001/1/27	5.1	196(41+155)	23K
4	1.1.9	2001/4/30	5.1	284(129+155)	29K
5	1.1.10	2001/10/14	5.2	304(133+171)	31K
6	1.1.11	2001/11/1	5.2	312(141+171)	32K
7	1.1.12	2001/12/12	5.3	416(174+242)	42K
8	1.1.13	2003/4/22	5.3	433(191+242)	44K
9	1.1.14	2004/6/24	5.4	740(215+525)	82K
10	1.1.15	2005/2/11	5.4	740(215+525)	82K
11	1.1.16	2006/7/30	5.4	740(215+525)	82K

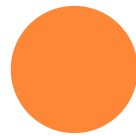


THE EVOLUTION OF USE RELATION



Use relation
between framework
and application

Gradually increase



RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



MAJOR EVOLUTION PERIODS IN JARP DEVELOPMENT

- We suggested a metrics to detect important updates in the development process.

Large size of
Modification



A lot of changes
in use relations



Component rank
is also changed.

- In the previous experiment, change of component rank was useful to guess an impact of the update.
 - **major-scale feature implementation**
 - **maintenance to core components**
 - **refactoring and restructuring of system**



MAJOR EVOLUTION PERIODS IN JARP DEVELOPMENT

Table 3. Component rank update metrics

	Versions	All	JARP	JHotDraw
1	1.0.0	-	-	-
2	1.0.0.1	0	0	0
3	1.0.1	0	0	0
4	1.1.9	0.06	0.26	0.01
5	1.1.10	0.02	0.02	0.03
6	1.1.11	0.02	0.03	0.01
7	1.1.12	0.05	0.07	0.04
8	1.1.13	0.01	0.01	0
9	1.1.14	0.18	0.05	0.21
10	1.1.15	0	0	0
11	1.1.16	0	0	0

Class allocation
is changed
drastically

JHotDraw is
Upgraded



MAJOR EVOLUTION PERIODS IN JARP DEVELOPMENT

- We can confirm which update has an impact on the system by using the change of component rank.
- We can also confirm which subsystem is affected by the update.
 - In ver. 1.1.9, functions of **mainwindow** are divided into subcomponents, and **tools** package is produced.
 - In ver. 1.1.12, functions of **PetriNetImpl** are divided into subcomponents, and **edition** and **simulation** packages are produced.
 - In ver. 1.1.10, 12, 14, JHotDraw is upgraded.

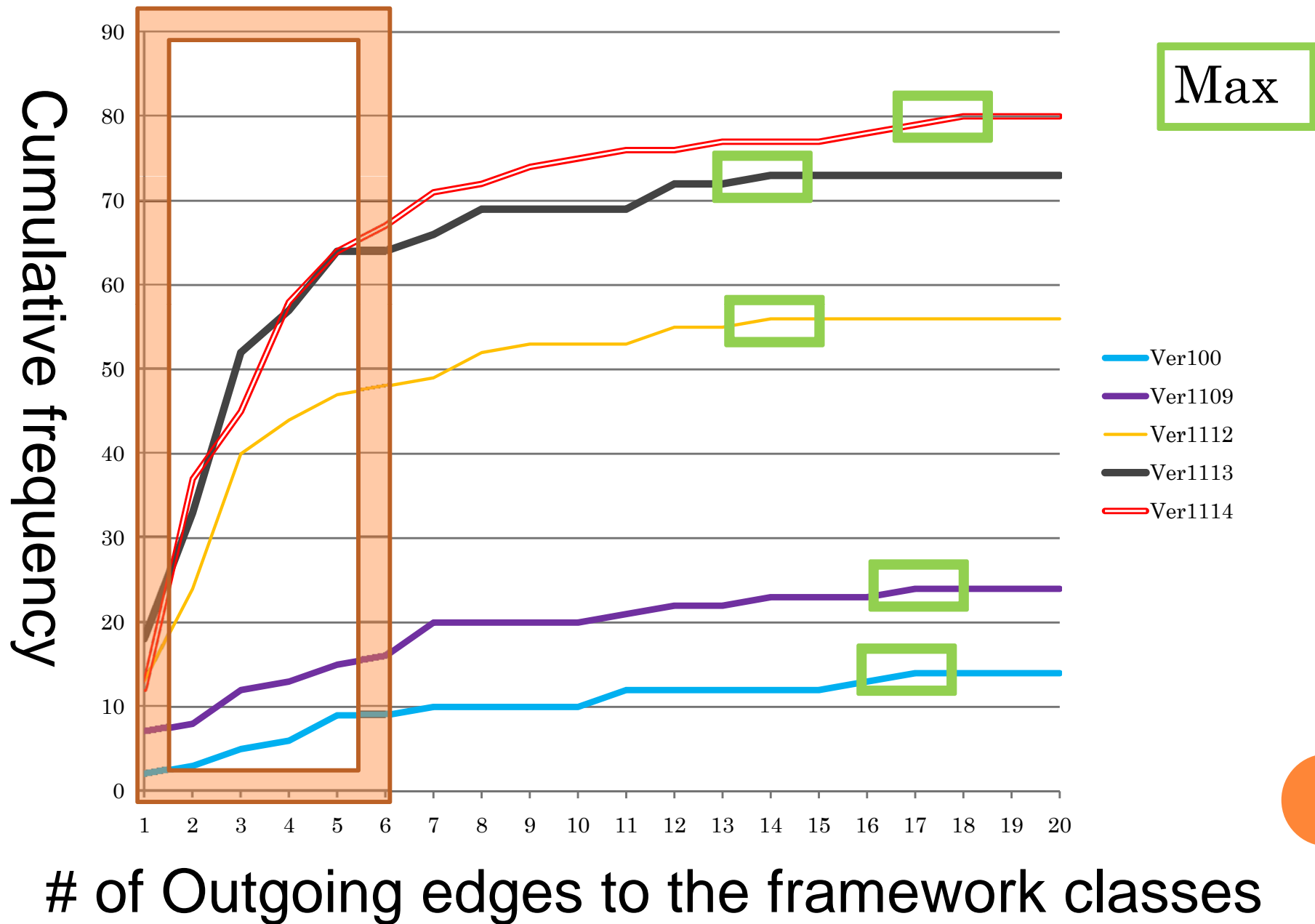


RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



EVOLUTION OF OUTGOING EDGES



EVOLUTION OF OUTGOING EDGES

	Ver 1.0.0		Ver 1.1.9	
1	JDrawingView	17	JDrawingView	17
2	MainWindow	16	PetriPlaceImpl	14
3	PetriPlaceImpl	11	PetriTransitionImpl	12
4	PetriTransitionImpl	11	PetriSelectionTool	11
5	PetriConnectionHandle	7	PetriNetImpl	7
6	PetriSelectionTool	5	PetriConnectionHandle	7
7	PetriArcImpl	5	PetriDragTracker	7
8	PetriSimulationTool	5	EditionTool	7
9	JHDLoadTool	4	PetriArcImpl	6
10	PetriDragTracker	3	PetriSimulationTool	5

Disappeared!!

	Ver 1.1.12		Ver 1.1.14	
	PlaceImpl	14	PlaceImpl	18
	TransitionImpl	12	ArcImpl	17
	SelectionToolEx	12	TransitionImpl	16
	ArcImpl	9	DrawingPreview	13
	PetriConnectionHandle	8	BendpointHandle	11
	PasteCommand	8	JDrawingView	10
	CreationTool	8	WeightHandle	9
	MainWindow	7	TokensHandle	9
	JDrawingView	6	PasteCommand	8
	PetriNetImpl	5	PetriNetImpl	7

Drastically decreased!!

EVOLUTION OF OUTGOING EDGES

- In summary, the number of outgoing edges increases little by little.
 - The number of components which have outgoing edges increases.
- However, the maximum number doesn't change so much.
 - If a large component has a lot of outgoing edges, it becomes a target of refactoring.
 - Decomposed into several components which have a small number of outgoing edges.

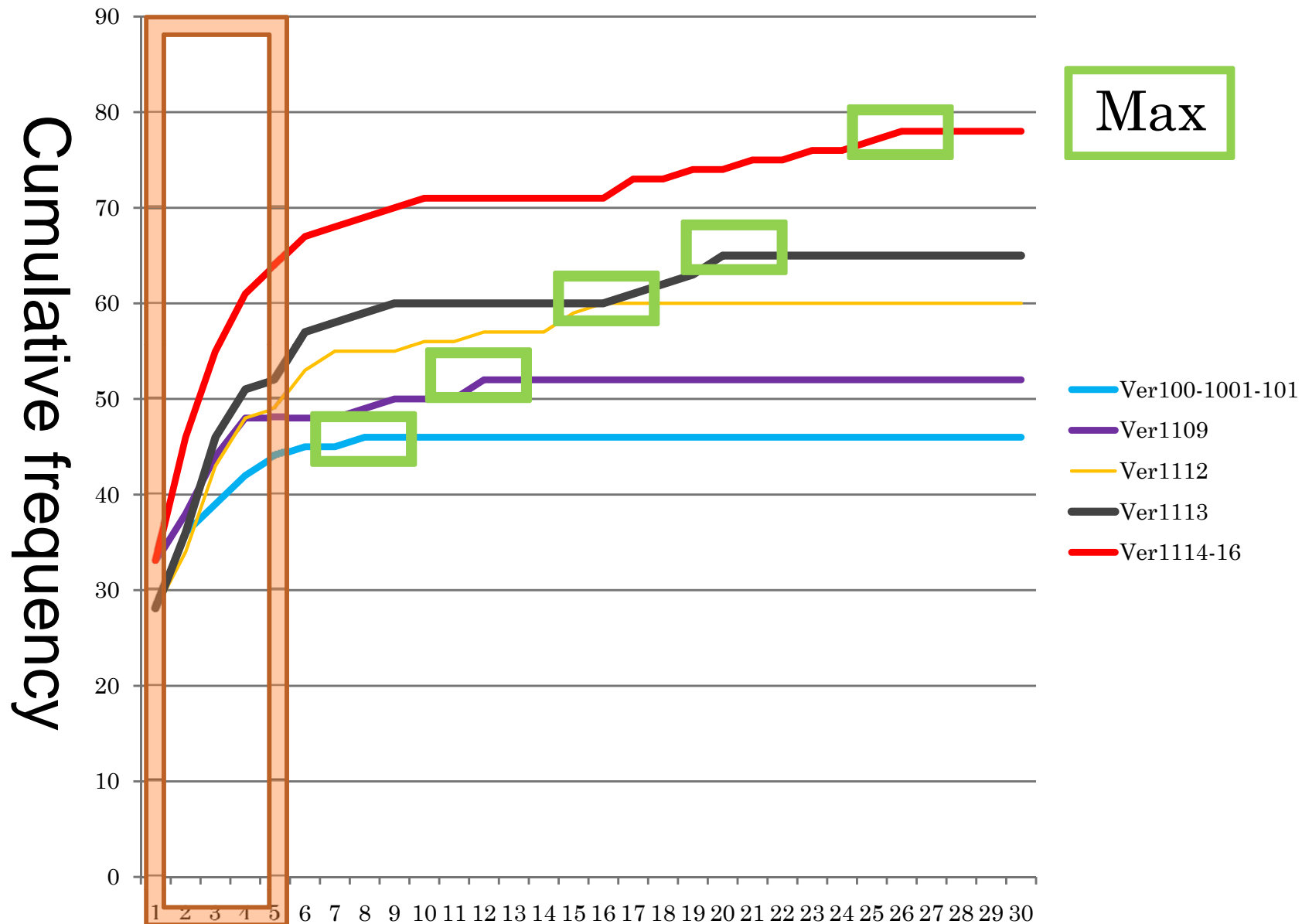


RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



EVOLUTION OF INCOMING EDGES




of Incoming edges from the application classes



EVOLUTION OF INCOMING EDGES

	JH 5.1 in JARP 1.0.0		JH 5.1 in JARP 1.1.9		
1	DrawingView	8	DrawingView	12	Increasing!!
2	Drawing	6	DrawingEditor	12	
3	util.StorableInput	5	Drawing	9	Increasing!!
4	Figure	5	Figure	8	
5	util.StorableOutput	4	util.StorableOutput	4	Increasing!!
6	Tool	4	util.StorableInput	4	
7	DrawingEditor	4	Tool	4	Increasing!!
8	ConnectionFigure	3	standard.AbstractFigure	4	
9	Connector	3	figures.AttributeFigure	3	Increasing!!
10	standard.AbstractFigure	3	figures.TextFigure	3	
	JH 5.3 in JARP 1.1.12		JH 5.4 in JARP 1.1.14		
	Drawing	16	Figure	26	
	util.UndoableCommand	15	FigureAttributeConstant	25	
	DrawingEditor	15	util.Command	23	
	DrawingView	12	DrawingView	21	
	Figure	10	DrawingEditor	19	
	util.StandardStorageFormat	7	util.UndoableCommand	17	
	util.Command	7	Drawing	17	
	Tool	6	FigureEnumeration	10	
	standard.AlignCommand	6	util.StandardStorageFormat	9	
	standard.StandardDrawingView	6	Tool	8	

EVOLUTION OF INCOMING EDGES

- For almost all of the components, the number of incoming edges increases.
 - The maximum number also increases.
 - Existing interface is not changed.
- The number of framework components which have incoming edges doesn't increase so much.
 - Interface of framework is well-organized. 

RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



TRANSITION OF COMPONENT RANK - JARP(APPLICATION)

	Ver 1.0.0	Ver 1.1.9	Ver 1.1.12	Ver 1.1.14
1	PetriNet	FindFilter	FindFilter	PetriNet
2	PetriNetEditor	FindProgressCallback	FindProgressCallback	FindFilter
3	PetriNetComponent	Config	Config	FindProgressCallback
4	PetriTransition	Name	Name	PetriNetEditor
5	PetriArc	EFileChooser	PetriNet	Tool
6	PetriPlace	XmlBrowser	PetriNetEditor	XMLResourceBundle
7	IntHashtableEntry	PetriNet	EFileChooser	ToolFactory
8	MainWindow	FindAccessory	XmlBrowser	figures.Transition
9	PetriStatesEnumAnalysis	PetriNetEditor	AbstractJARPTool	Main
10	ImageEncoder	AdapterNode	FindAccessory	figures.Place



TRANSITION OF COMPONENT RANK - JARP(APPLICATION)

- Component Rank depends on the implemented function.
 - Ver. 1.0.0 : About **Petri-net**
 - Ver. 1.1.9- : About **Petri-net, Filter, Progress Callback, XMLBrowser**
- Based on the increase of rank, we can confirm
 - What functions are newly implemented?
 - What type of data is used for new function?



TRANSITION OF COMPONENT RANK - JHOTDRAW(FRAMEWORK)

	JH 5.1 in JARP 1.0.0	JH 5.1 in JARP 1.1.9
1	Figure	Figure
2	util.Storable	util.Storable
3	Connector	Connector
4	FigureEnumeration	FigureEnumeration
5	Locator	Locator
6	FigureChangeEvent	FigureChangeEvent
7	FigureChangeListener	FigureChangeListener
8	util.StorableInput	util.StorableInput
9	util.StorableOutput	util.StorableOutput
10	ConnectionFigure	ConnectionFigure

JH 5.3 in JARP 1.1.12	JH 5.4 in JARP 1.1.14
Figure	Figure
FigureEnumeration	DrawingView
Connector	FigureEnumeration
Locator	JHotDrawRuntimeException
FigureChangeEvent	Connector
FigureChangeListener	ConnectionFigure
util.Storable	Drawing
DrawingView	Handle
ConnectionFigure	util.CollectionsFactory
util.StorableInput	DrawingEditor

TRANSITION OF COMPONENT RANK - JHOTDRAW(FRAMEWORK)

- Component rank is useful for detecting core components in the framework.
 - Some components are joined into core components during development.
 - Half of core components are still on the list.
 - Because interface of framework is not changed.



RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



THE IMPACT OF FRAMEWORK UPGRADING

- If the framework is upgraded, some application component would be modified for adjusting.
- Research question:
 - What component's component rank is changed?
 - What kind of components should be reviewed?
- We list JARP components whose rank goes up when JHotDraw is upgraded.
 - affected components



THE IMPACT OF FRAMEWORK UPGRADING

Table 8. JARP Component rank's change between ver1.1.9 and 1.1.10 (129 components)

	Class	ver9	ver10	diff
1	PetriNetImpl	92	62	30
2	Crc32Hash	71	59	12
3	Hash	24	17	7
3	PetriSelectionTool	80	73	7
5	25 components	-	-	1

Table 9. JARP Component rank's change between ver1.1.11 and 1.1.12 (124 components)

	Class	ver11	ver12	diff
1	PNMLStorageFormat	107	59	48
2	FormatTool	94	54	40
3	AlignTool	94	55	39
4	EditionTool	94	60	34
5	Main	60	30	30
5	LoadTool	94	64	30
7	PrintTool	94	78	16
8	FileFilterImpl	107	93	14
9	SplashWindow	78	66	12
10	PetriNetMarking	21	13	8

Table 10. JARP Component rank's change between ver1.1.13 and 1.1.14 (130 components)

	Class	ver13	ver14	diff
1	LanguageTool	107	22	85
2	ChangeNetNameTool	117	84	33
2	FindPathAnalysis	117	84	33
4	SelectionTool	104	72	32
5	Main	31	6	25
6	PetriInvariantAnalysis	77	53	24
7	CommentTool	107	84	23
7	GridTool	107	84	23
7	NewTool	107	84	23
7	NewWindowTool	107	84	23

Tool

Main

Utility class



THE IMPACT OF FRAMEWORK UPGRADING

- Components which use framework class, such as **tool classes**, are affected.
 - Direct use relation changed.
 - Some of them have almost the class same structure.
- **Main** and **utility** classes are also affected.
 - Direct and Indirect use relation changed.



RESULTS OF EXPERIMENTS

- General Information
- Major evolution periods in JARP development
- The evolution of outgoing edges
- The evolution of incoming edges
- The transitions of component rank
- The impact of framework upgrading
- The impact of incoming edges from application classes



THE IMPACT OF INCOMING EDGES

Research question:

- Are frequently used components in framework different when external use relations are considered?
- We compared these two rankings:
 - Component rank based on **JHotDraw classes only**.
 - Component rank based on **both JHotDraw and JARP classes**.
 - Only JHotDraw classes are extracted.



THE IMPACT OF INCOMING EDGES

Table 11. JHotDraw Component rank's change in JARP ver1.1.9 (156 components)

	Class	Set3	Set3'	diff	Used
1	AlignCommand	95	133	38	1
1	ToggleGridCommand	95	133	38	1
3	ChangeAttributeCommand	92	125	33	1
3	DeleteCommand	75	108	33	2
5	DragTracker	81	111	30	1
6	ConnectionHandle	67	87	20	1
7	BringToFrontCommand	114	133	19	1
7	SendToBackCommand	114	133	19	1
9	BufferedUpdateStrategy	91	108	17	1
10	CopyCommand	119	133	14	1
10	CutCommand	119	133	14	1
10	PasteCommand	119	133	14	1
13	ChopEllipseConnector	77	88	11	1
14	GroupHandle	98	107	9	0
15	PolyLineHandle	52	60	8	1
15	SelectionTool	63	71	8	2
17	Clipboard	50	56	6	1
18	RadiusHandle	93	98	5	0
18	ShortestDistanceConnector	93	98	5	0
18	DrawingEditor	13	18	5	12

Table 12. JHotDraw Component rank's change in JARP ver1.1.12 (241 components)

	Class	Set3	Set3'	diff	Used
1	UndoableCommand	43	199	156	15
2	StorageFormatManager	48	201	153	4
3	AlignCommand	68	201	133	6
4	ChangeAttributeCommand	90	185	95	3
5	StandardDrawingView	61	147	86	6
6	UndoableTool	79	164	85	4
7	ToggleGridCommand	120	201	81	1
8	BringToFrontCommand	124	201	77	1
8	SendToBackCommand	124	201	77	1
10	RedoCommand	132	201	69	1
10	UndoCommand	132	201	69	1
12	DeleteCommand	102	167	65	1
13	CopyCommand	138	201	63	1
13	CutCommand	138	201	63	1
15	PasteCommand	159	201	42	1
16	UndoActivity	118	157	39	1
17	Alignment	55	89	34	6
18	StandardStorageFormat	49	79	30	7
19	ConnectionHandle	104	122	18	1
19	Clipboard	73	91	18	3

Handler

Command

Connector

THE IMPACT OF INCOMING EDGES

- Many **command** classes went up in ranking.
 - Some components are not used in the framework.
 - These components implements specific function.
 - Direct use relations from application
- **Handler** and **connector** also appears.
 - Indirect use relations from application
- There is a little difference between **components used in the framework** and **components used in the application**.
 - This information is useful for reorganization.



SUMMARY: THE EFFECTIVENESS OF USE RELATION ANALYSIS

- Use relation between components assists to grasp the entire of software.
 - The change of use relation is closely related to the activities in the development.
 - The change of use relation highlights other characteristics of software.
- Component rank is useful to assess the change.
 - To grasp newly implemented functions roughly.
 - To detect core component.
 - To confirm the affected components by the update.

CONCLUSION

- In this research, we observed the evolution of use relation between components between framework and application
- Metrics about use relation is useful to grasp the overview of software.
 - The change of use relation is closely related to the activities in the development.
 - Component rank is also a good sources of information.



FUTURE WORKS

- To observe an impact of application in another situation.
 - If the existing API of framework is changed?
 - If the framework itself is completely replaced?
- To estimate the cost of upgrading of framework.
 - Which application components are really modified?
 - Can we estimate needed effort roughly?
- Use relation analysis in an another theme.
 - evaluation of a refactoring method based on aspectization.



Thank you

