RT-205: Identifying and Measuring Modularity Violations in Cyber-physical Systems

Sponsor: DASD(SE)

By
Dr. Lu Xiao
Dr. Michael Pennock
10th Annual SERC Sponsor Research Review
November 8, 2018
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW, 8th Floor
Washington, DC 20009

www.sercuarc.org

Agenda

- Introduction
- Module decomposer
  - —Package view (development view)
  - —Dependency hierarchy view (sequential work allocation)
  - —Organizational view (vendor-lock in)
- Domain concept learner
- Next Steps

Agenda

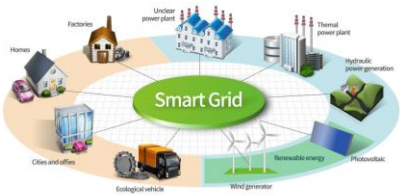
- Introduction
- Module decomposer
  - Package view (development view)
  - Dependency hierarchy view (sequential work allocation)
  - Organizational view (vendor-lock in)
- Domain concept learner
- Next Steps

Cyber-physical Systems (CPS)

- The term cyber-physical system was first coined in 2006. A cyber-physical system is an integration of computation with physical processes.

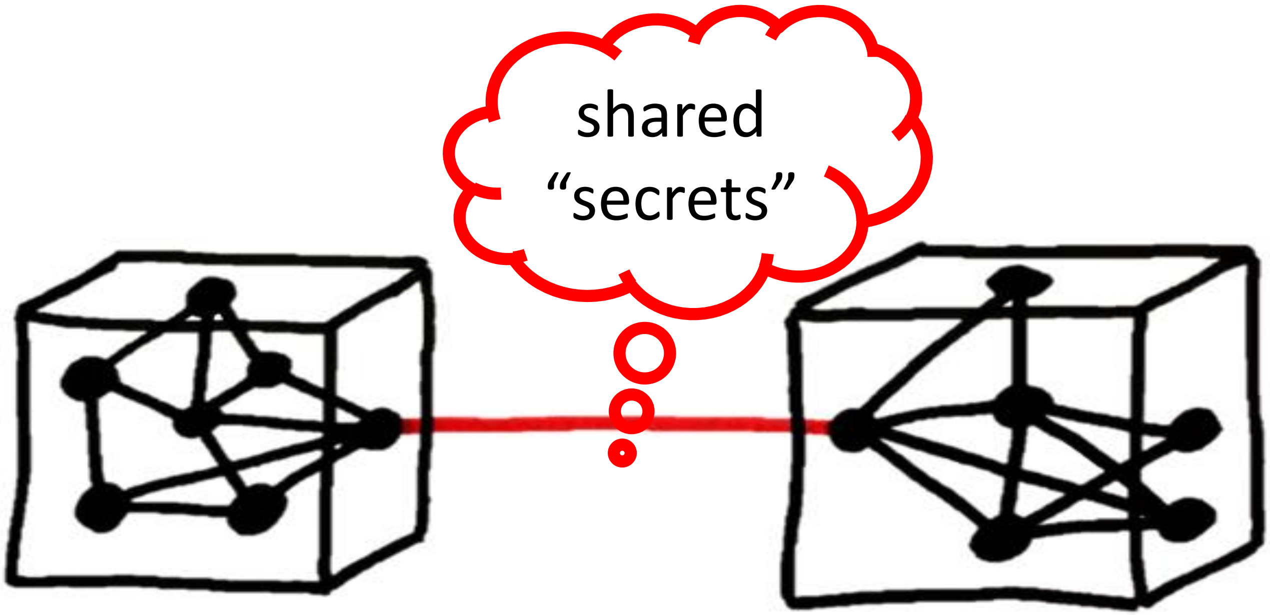
- *Physical* and *software* components are deeply *intertwined and interacting* with each other under changing context.

- Cyber-physical systems have gained wide-spread application in diverse areas including: civil infrastructure, energy, healthcare, transportation, automotive, smart appliances, and others



Modularity Violations on CPS

- Two supposedly independent modules are actually coupled in the evolution of the system.
  - For example, an update to one module requires a corresponding update to another module.



Prior Results from RT-180

- We conducted preliminary studies on OpenWrt and MD PnP to prove the feasibility of implying hardware related modularity violations using software data.

- Findings include:

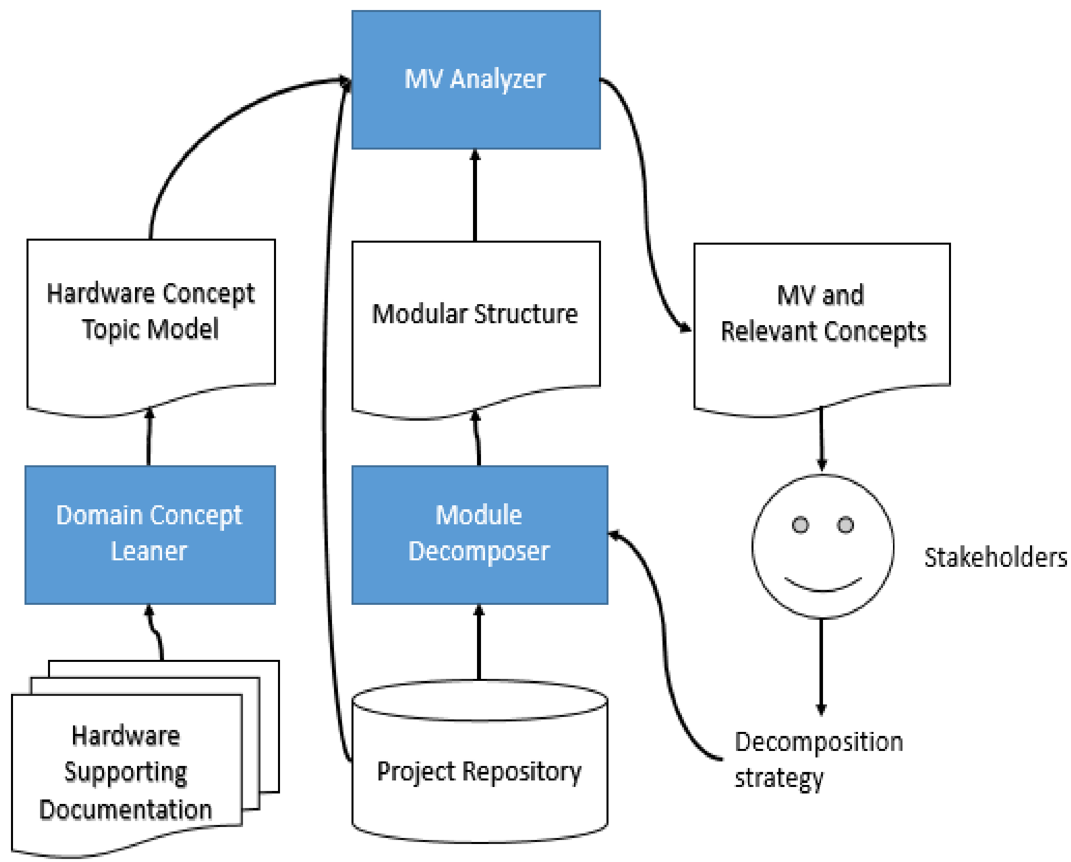
1. The OpenWrt is more modularized than about 85% of the 129 (commercial and open source) traditional software systems.
2. In OpenWrt, hardware concepts are the potential underlying causes of software-level modularity violations.
3. Using software side data help to imply hardware-related modularity violations in OpenWrt.

Limitations of Prior Work

- In the incubator phase, we treated source files as the granularity of a module. This is appropriate from the perspective of a low-level developer. However, a project owner views modules as cohesive functional components to deliver the product value and competitiveness.
- In the incubator phase, we used manually extracted hardware-related keywords as heuristics to identify modularity violations. However, the keywords developed for one project domain may not be applicable to another project domain.

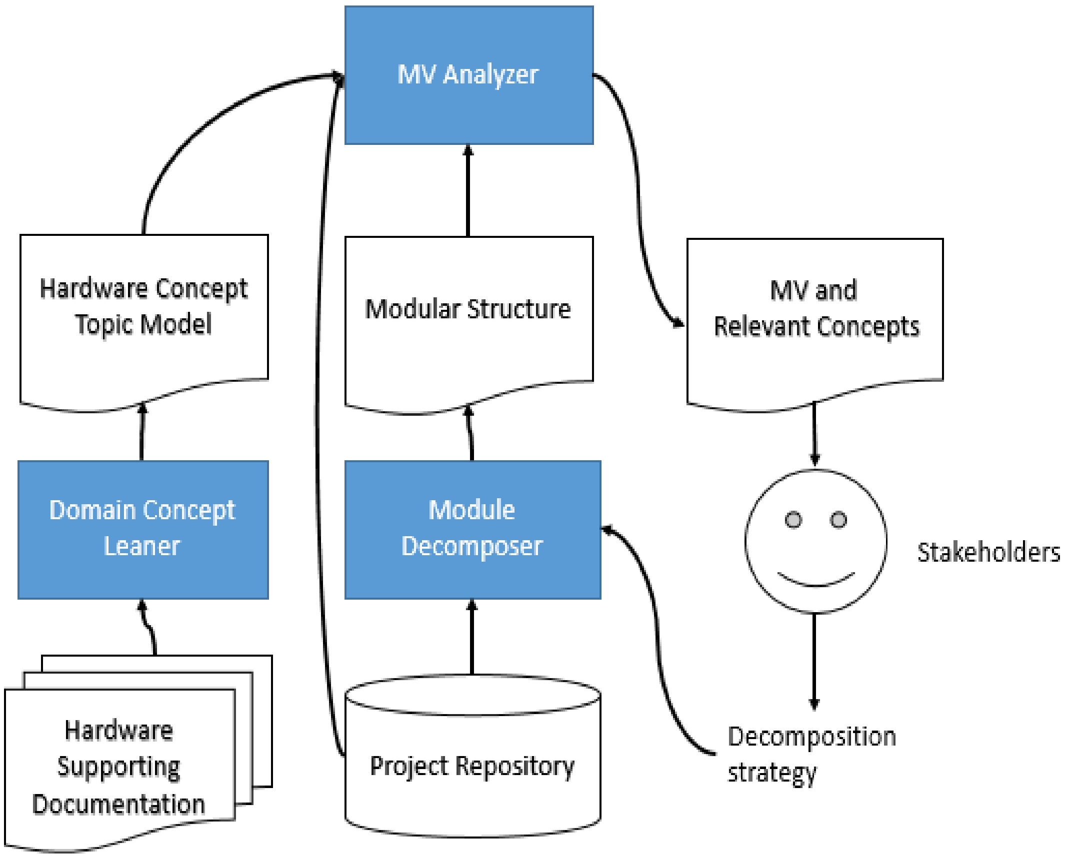
Proposed Work

- Examine the Criteria to Decompose a CPS into Modules
- Build a “Domain Concept Learner” to Identify Modularity Violations in Different Domains
- Build Decision Framework and Demonstrator



Proposed Work

- ✓ *Examine the Criteria to Decompose a CPS into Modules*
- ✓ *Build a "Domain Concept Learner" to Identify Modularity Violations in Different Domains*
- Build Decision Framework and Demonstrator



Agenda

- Introduction
- **Module decomposer**
  - —Package view (development view)
  - —Dependency hierarchy view (sequential work allocation)
  - —Organizational view (vendor-lock in)
- Domain concept learner
- Next Steps

Case Study Subjects

- OpenWrt: A Linux operating system targeting embedded devices. It frees you from the application selection and configuration provided by the vendor and allows you to customize the device through the use of packages to suit any application.
  - https://openwrt.org/
- MdPnP: The medical device "Plug-and-Play" interoperability program advancing safe and secure interoperability to improve patient care.
  - http://www.mdpnp.org/

Case Study Subjects

- OpenWrt
  - 1063 source files (in c language)
  - 80 developers
  - 42018 commits
    - 1996 commits include .h or .c files
    - 40107 commits not include any .h nor .c files
- MdPnp:
  - 808 source files (in java language)
  - 7 developers
  - 1611 commits
    - 993 commits include .java file,
    - 618 commits not include any .java file

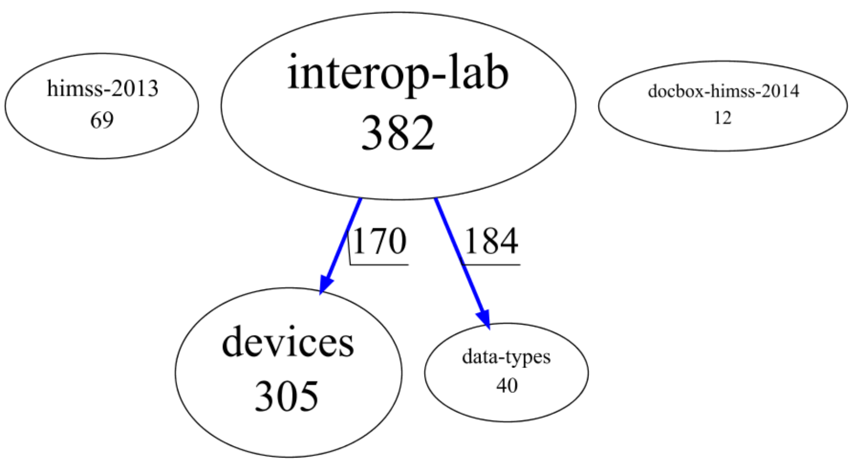
Criteria for Modular Decomposition

- There are different criteria to decompose a large-scale, complex system into modules based on different stakeholders concerns.
  1. The natural package decomposition (a.k.a. the development view).
  2. The dependency hierarchy decomposition that represents a system as layers.
  3. The modular structure based on the organizational structure of a system.
- There are two different dimensions of relationship among modules:
  - The static structural dependencies
  - The co-change relationship

Root Package: Structural Dependencies

Md PnP

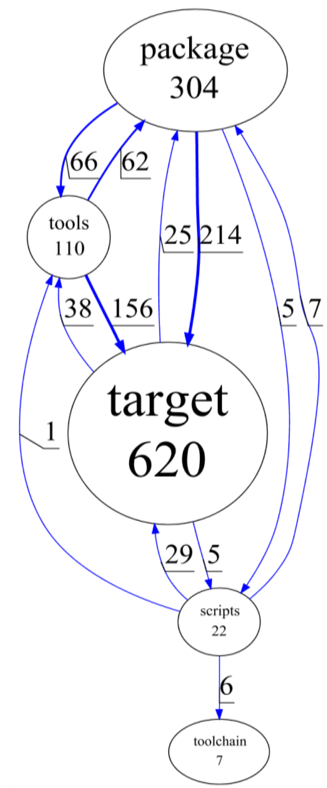
- “interop-lab” and “devices”
- “interop-lab” and “data-types”



- First impression: MD PnP has simpler modular structure

OpenWrt

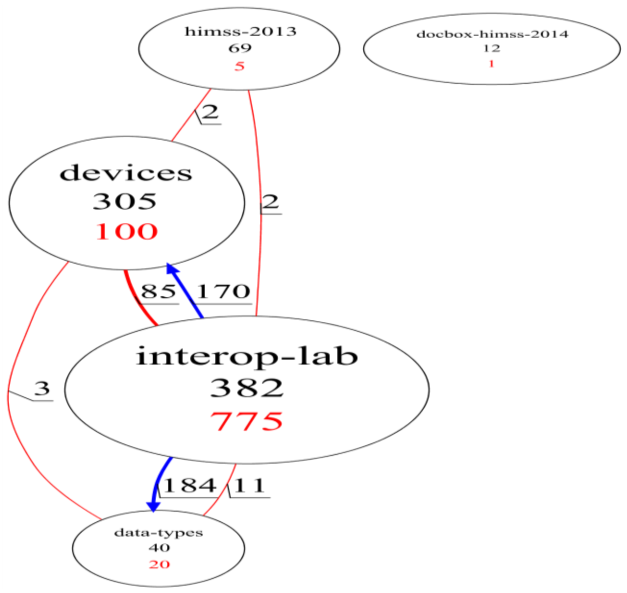
- Every module is connected with almost every other modules

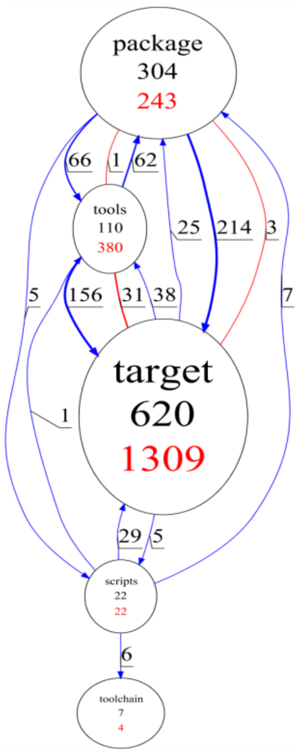


Root Package: Co-change Relationship

MD PnP **OpenWrt**

- The red lines: the number of cross-module changes
- The red characters: the number of inner module changes

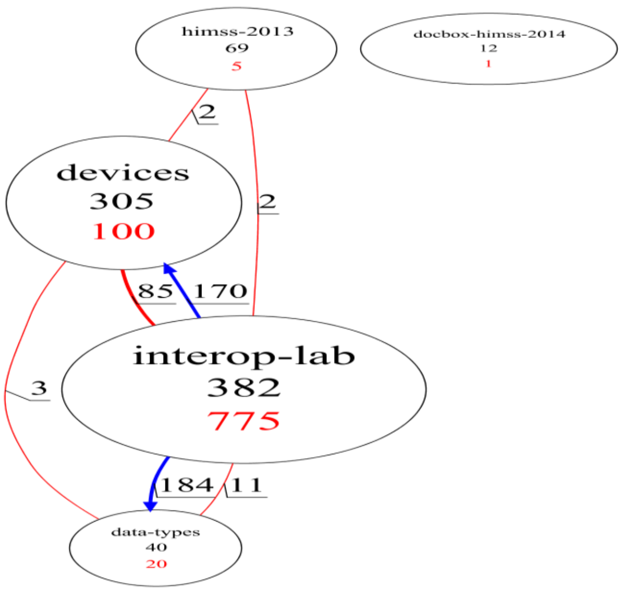




Root Package: Co-change Relationship

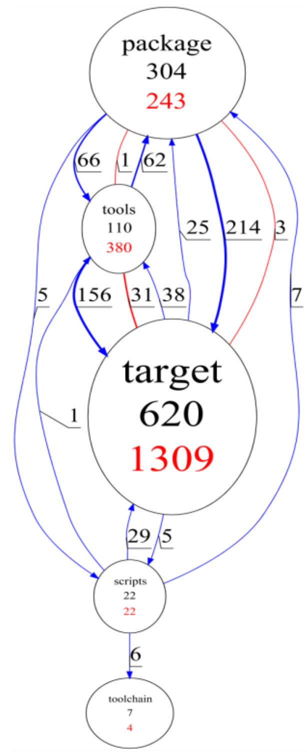
MD PnP

- 90% inner module changes
- 10% cross module co-changes



OpenWrt

- 98% inner module changes
- 2% cross module co-changes

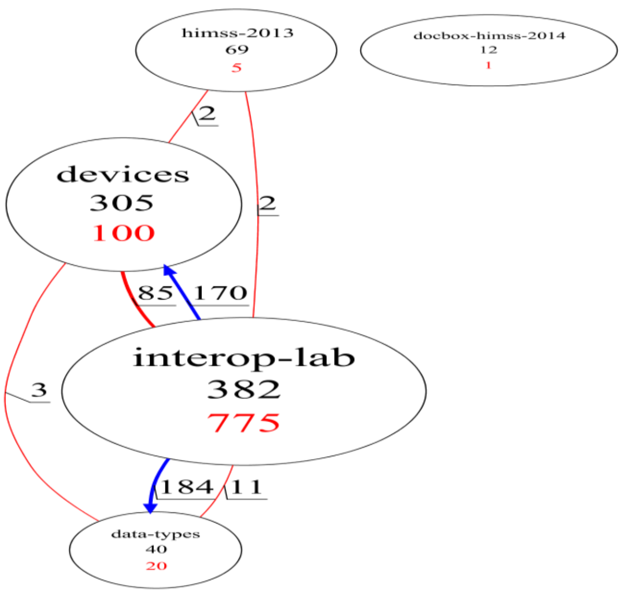


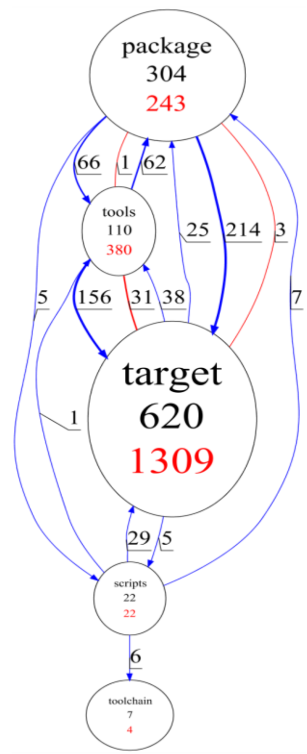
Root Package: Co-change Relationship

MD PnP

OpenWrt

- Modules in MD PnP are more likely to co-change with each other compared to OpenWrt.



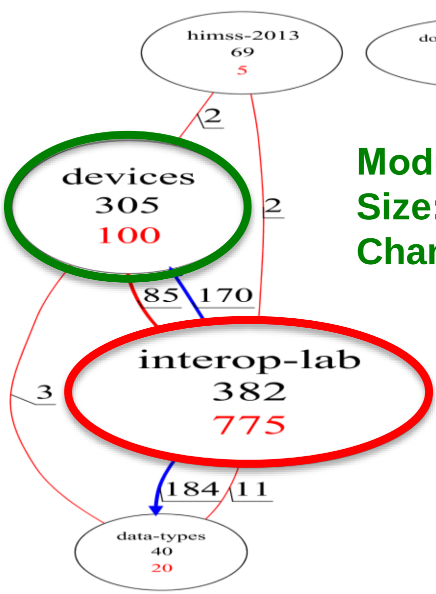


Root Package: Expensive Modules

MD PnP

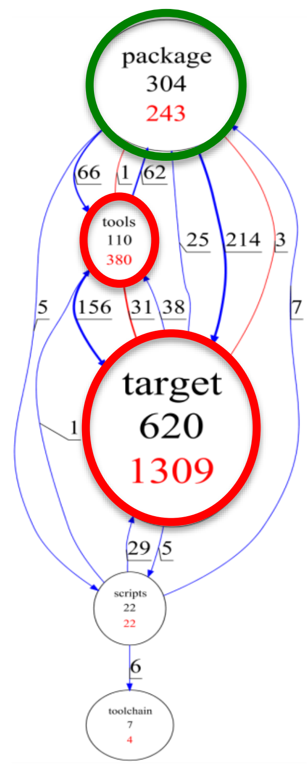
OpenWrt

- In both projects, some modules are more expensive to maintain or replace than other modules.



Module “devices”:
Size: 37%
Change: 11%

Module “interop-lab”:
Size: 47%
Change: 86%



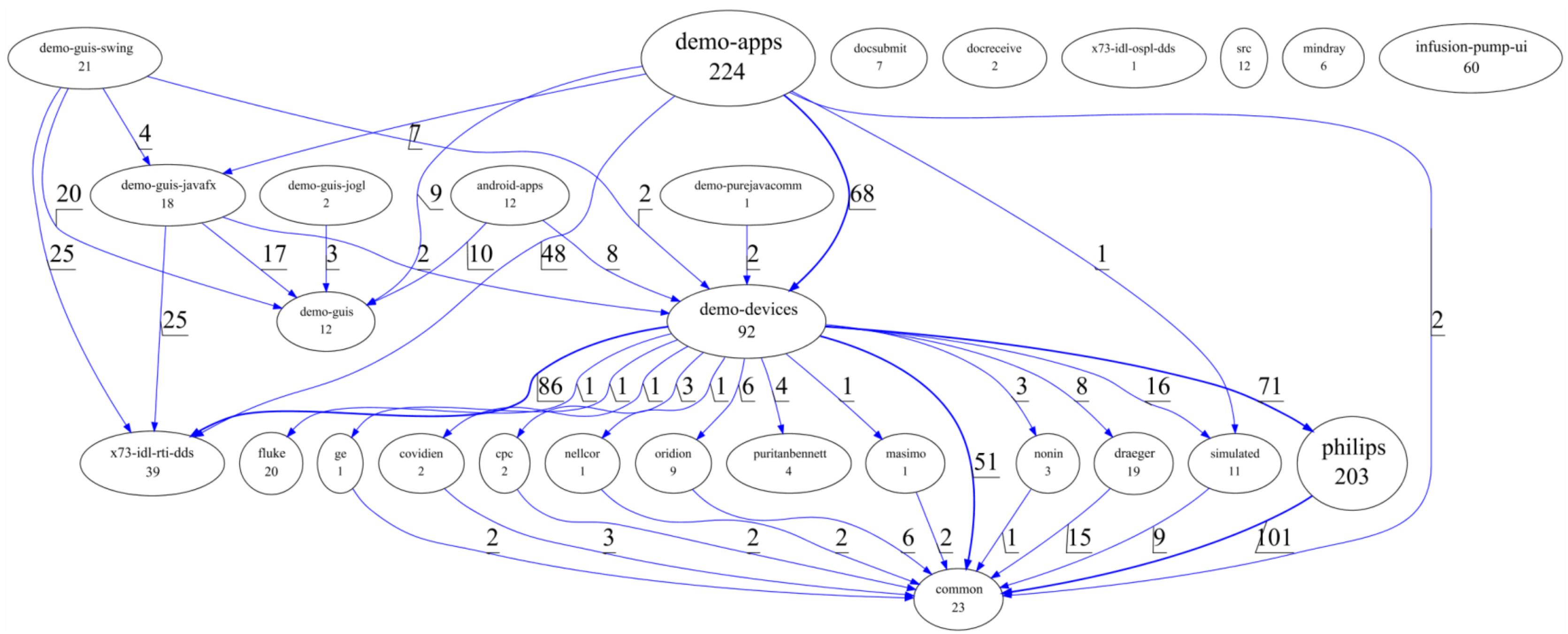
Module “package”:
Size: 29%
Change: 12%

Module “tools”:
Size: 10%
Change: 20%

Module “target”:
Size: 58%
Change: 67%

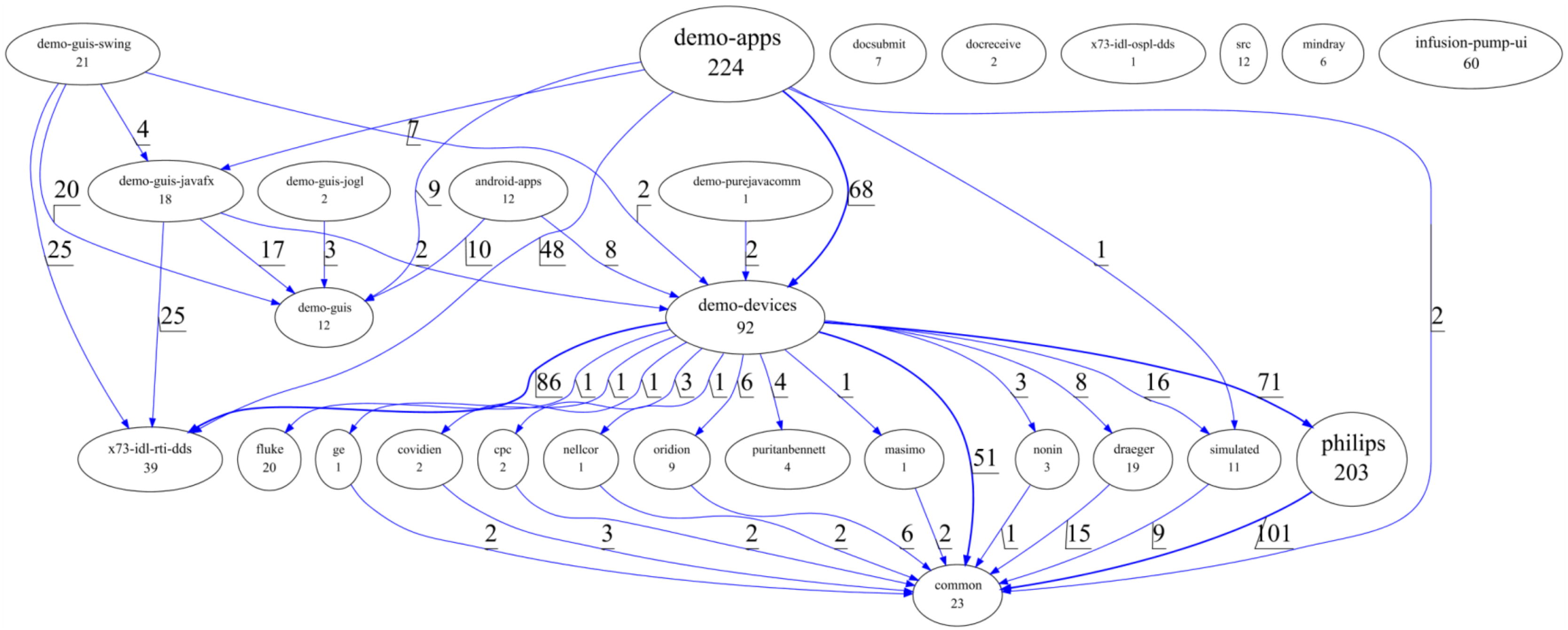
Fine-grained Package: Structural Dependencies

- Some stakeholders need more fine-grained modular structure for more detailed guidance.
- MD PnP: Fine-grained Package Structure Visualized:



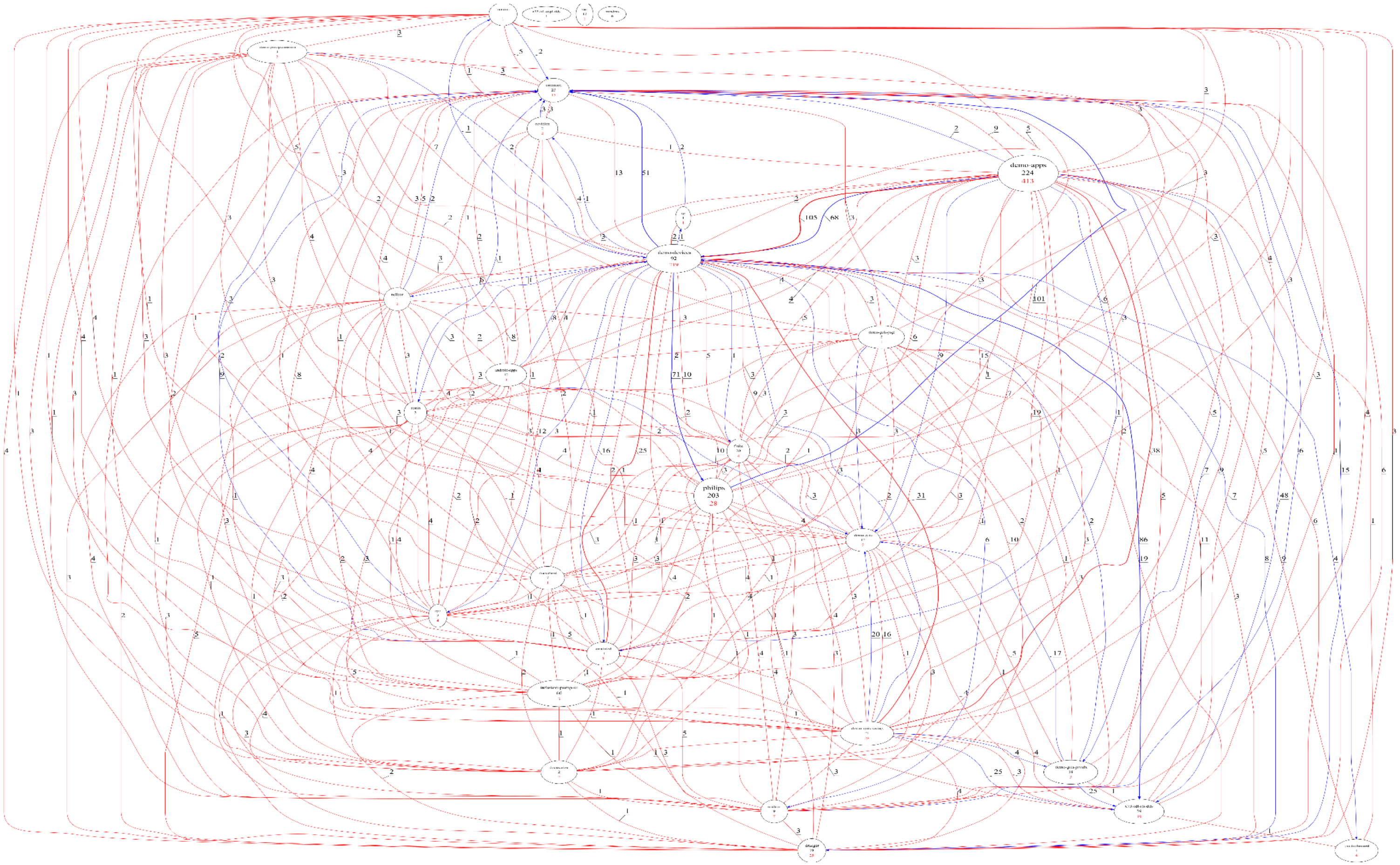
Fine-grained Package: Structural Dependencies

- First impression: the structure of MD PnP is clearly layered
  - module “demo-devices” plays a role of façade.



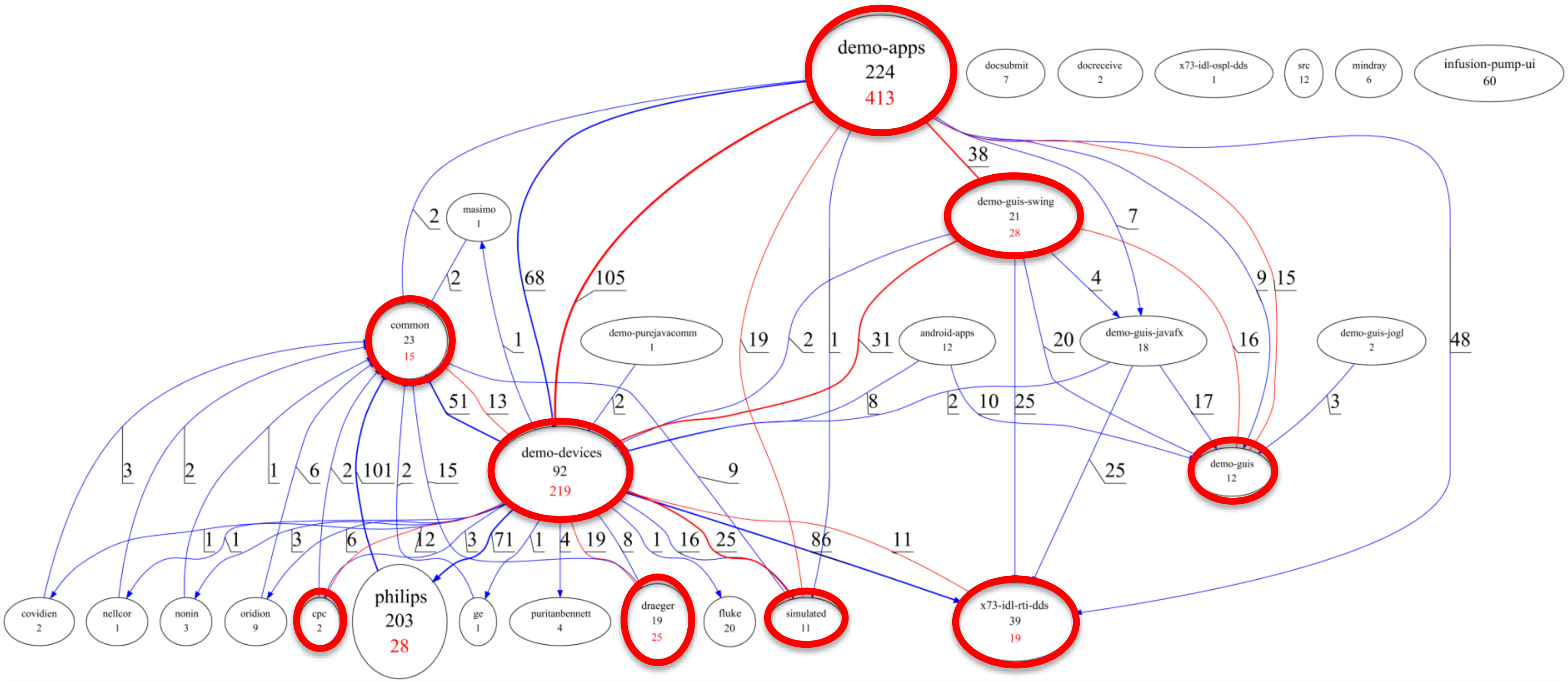
Fine-grained Package: Co-change Relationship

- Reality: everything changes with everything else!
- Thus, high maintenance costs, vendor lock-in, …



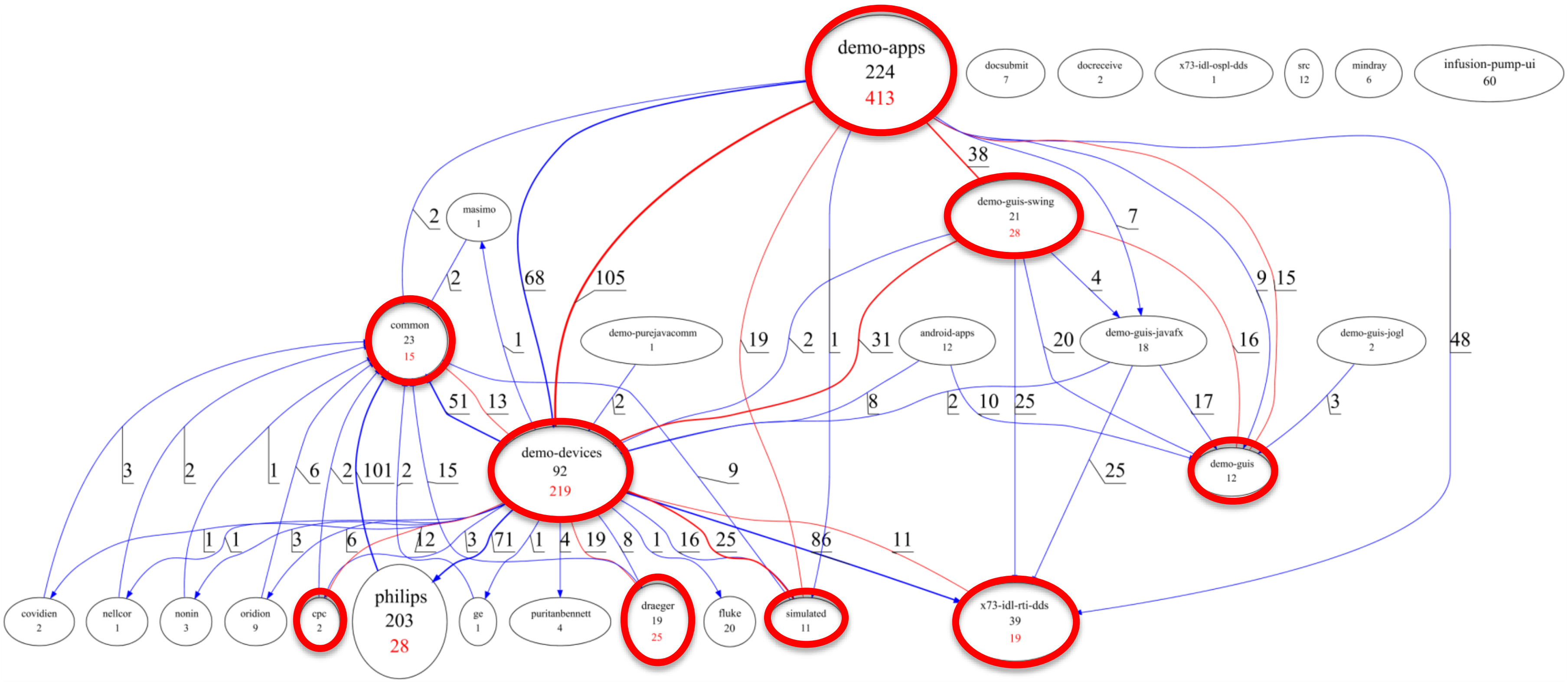
Fine-grained Package: Co-change Relationship

- We filter out co-change relationships whose weight is less than 10
- There are only 9 modules with a co-change weight of 10 or more



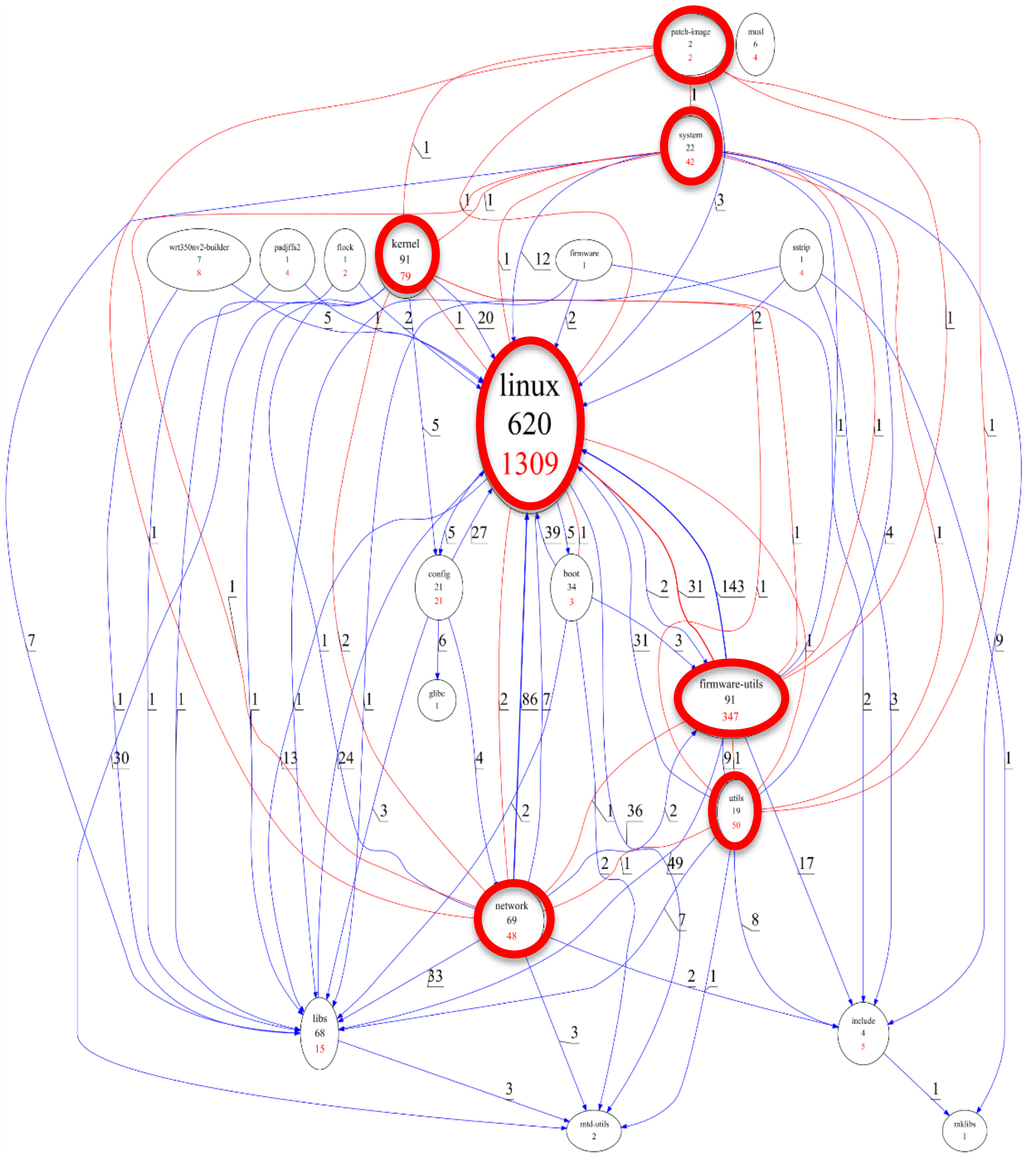
Fine-grained Package: Co-change Relationship

- Stakeholders should give top priority to these 9 modules



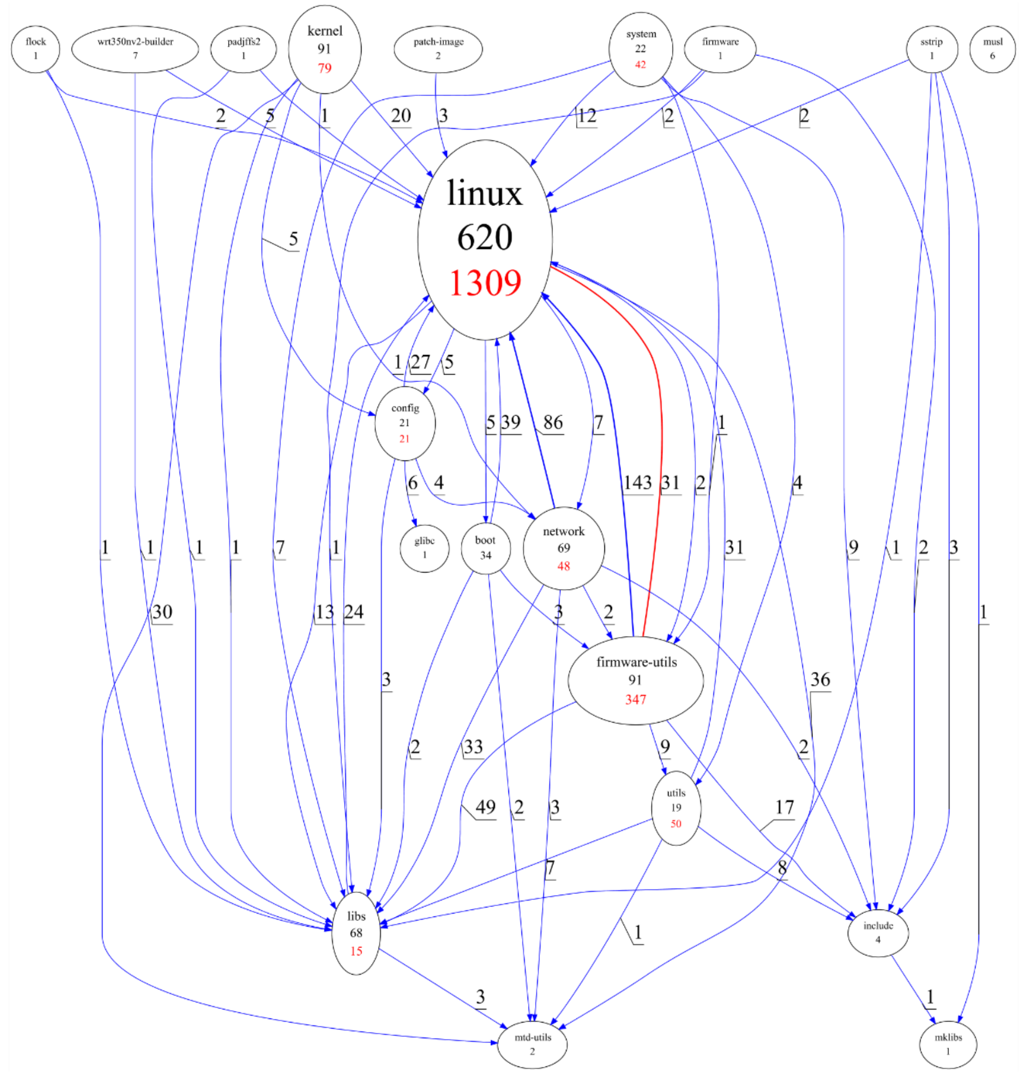
Fine-grained Package: Co-change Relationship

- OpenWrt: Structurally, the modules are highly dependent on each other. However, the co-changes are only among 7 modules
- OpenWrt is more decoupled in maintenance than is suggested by its structure.



Fine-grained Package: Co-change Relationship

- OpenWrt: we filter out co-changes less than 10
- Only “linux” and “firmware-utils” have co-changes more than 10.
- Modules in OpenWrt are almost perfectly decoupled from each other in maintenance!



Modules based on Dependency Hierarchy

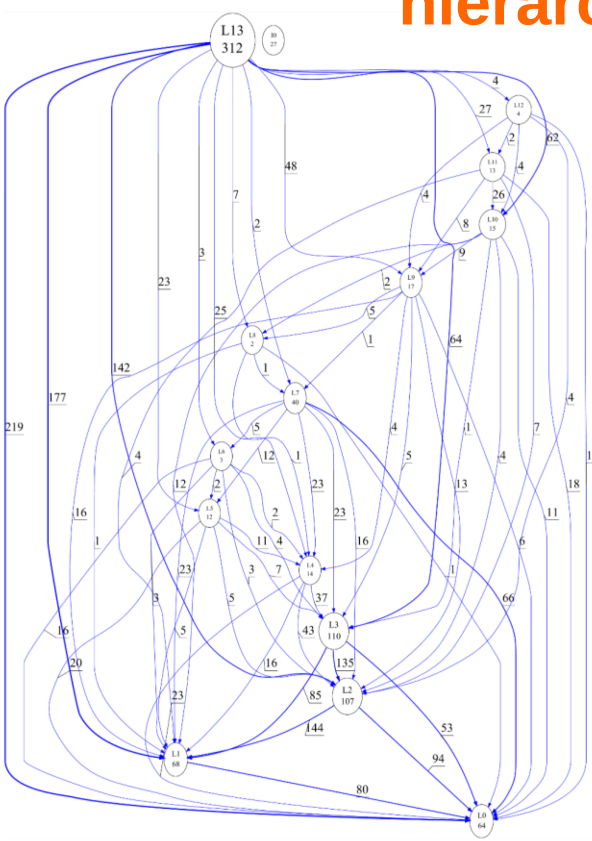
MD PnP

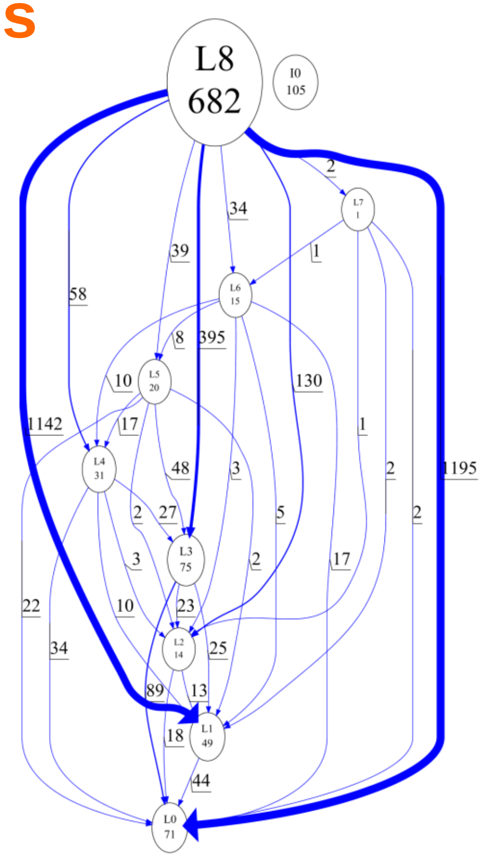
- The system contains 13 layers

OpenWrt

- The system contains 8 layers

Each layer is formed by the actual dependency hierarchy among files





Modules based on Dependency Hierarchy

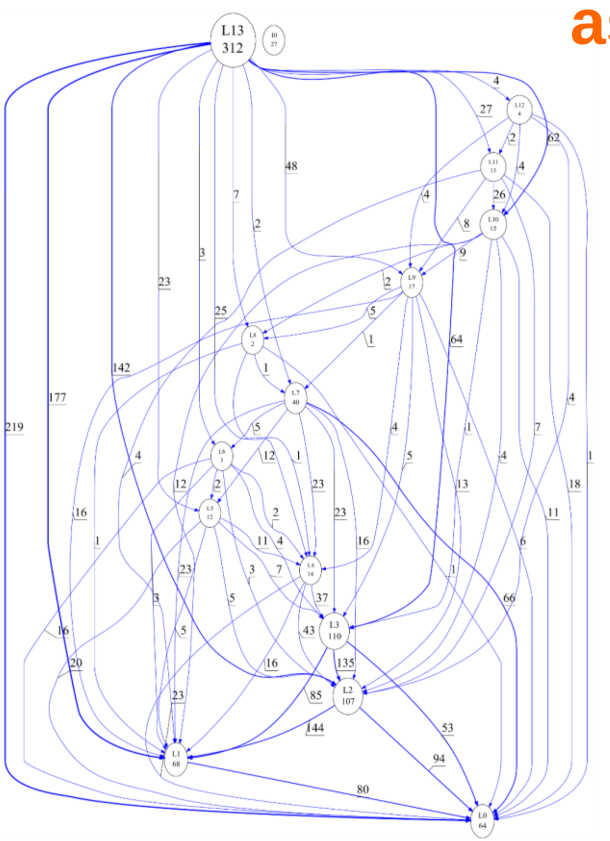
MD PnP

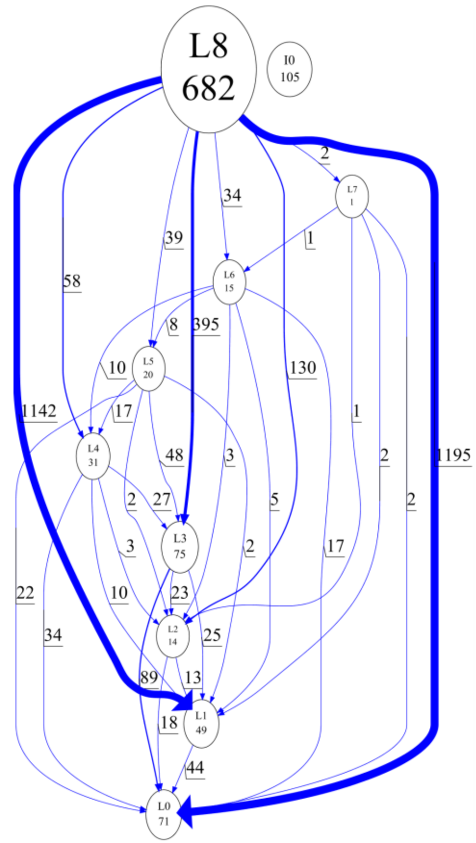
- The system contains 13 layers

OpenWrt

- The system contains 8 layers

The layers could be used for sequential task assignment

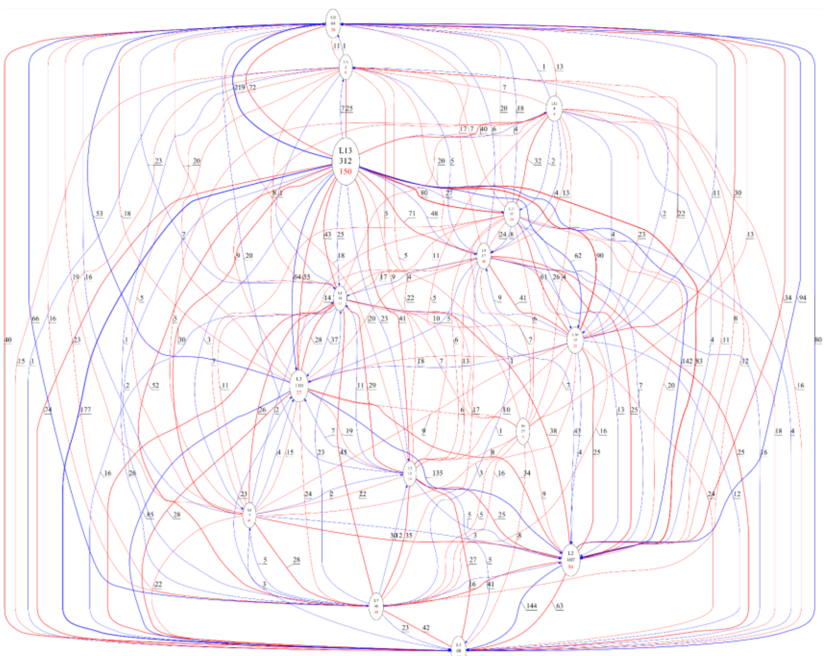


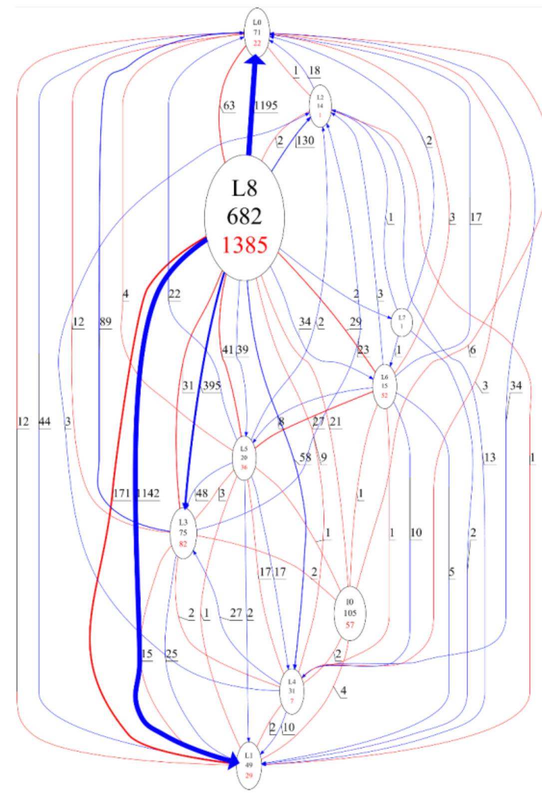


Modules based on Dependency Hierarchy

MD PnP | **OpenWrt**

Reality: everything change with everything else!



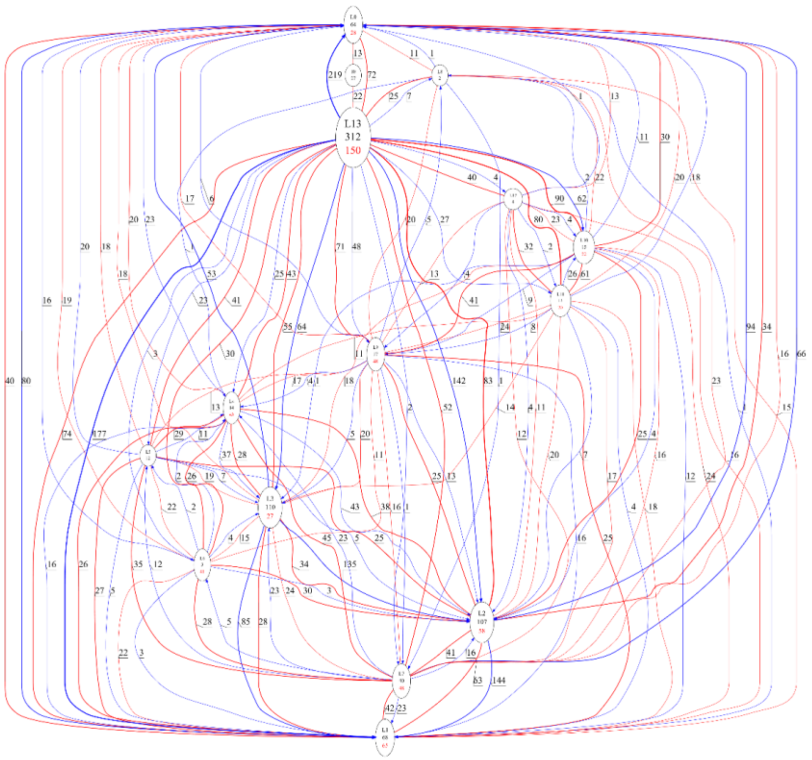


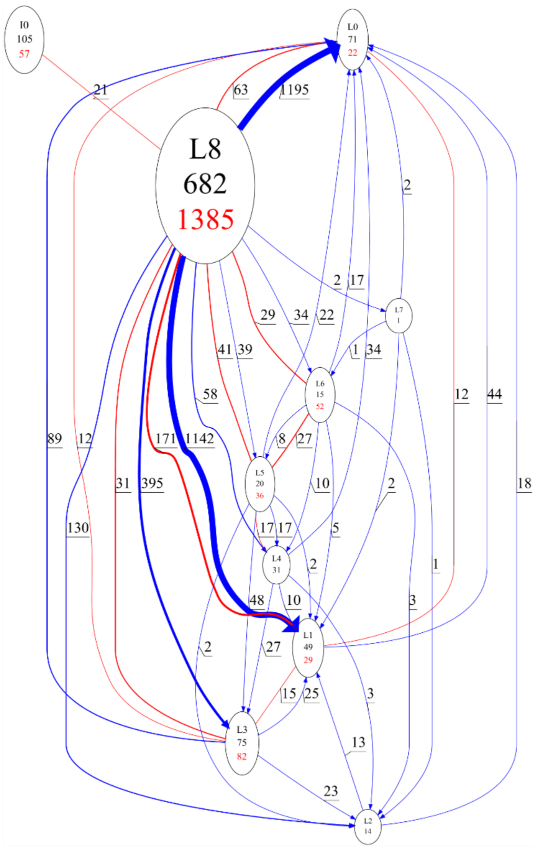
Modules based on Dependency Hierarchy

MD PnP

OpenWrt

Co-changes above 10 : still everything changes with almost everything else!

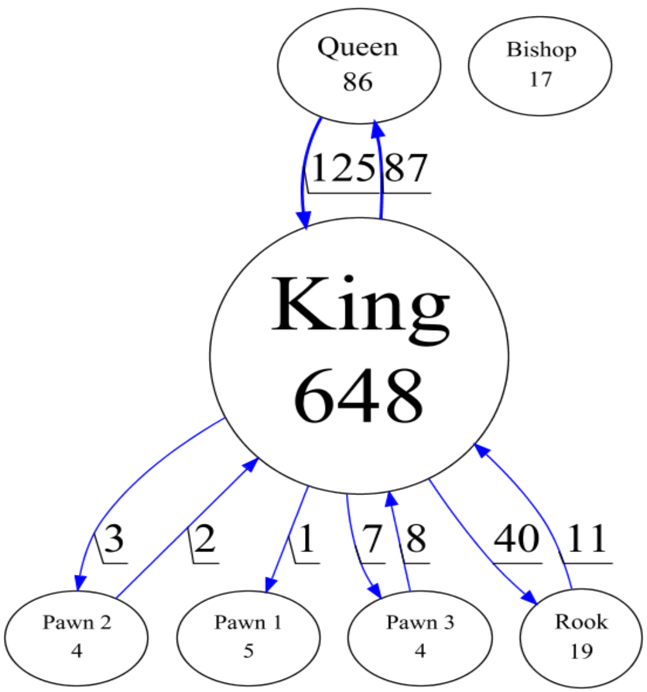




Modular Structure based on Organizational Structure

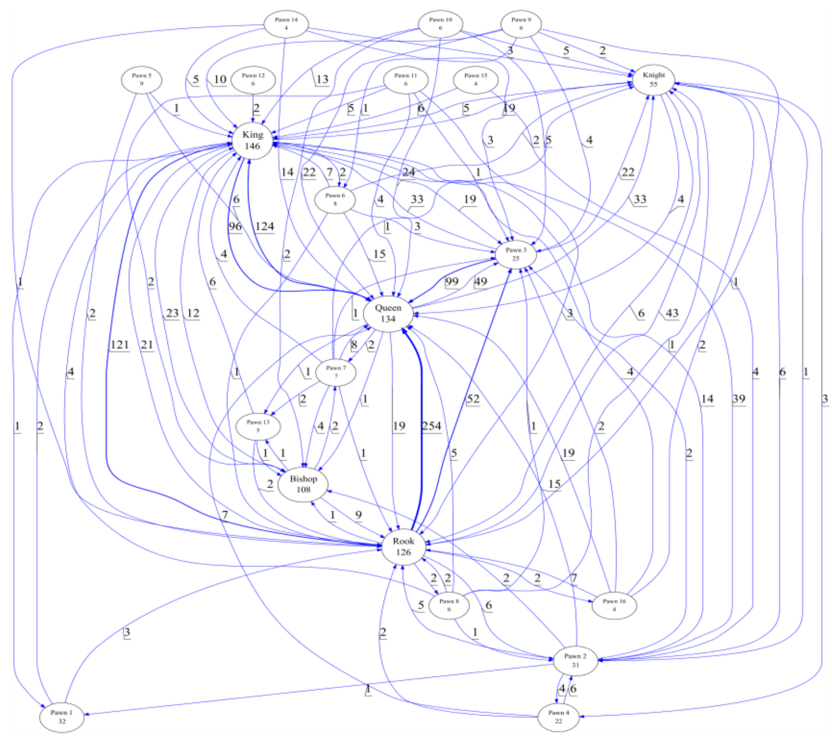
MD PnP

- 7 main contributors
- The “King” is the main contributor and his module decouples other contributors’ modules



Open Wrt

- 21 main contributes
- Everyone’s work is related to everyone else’s.

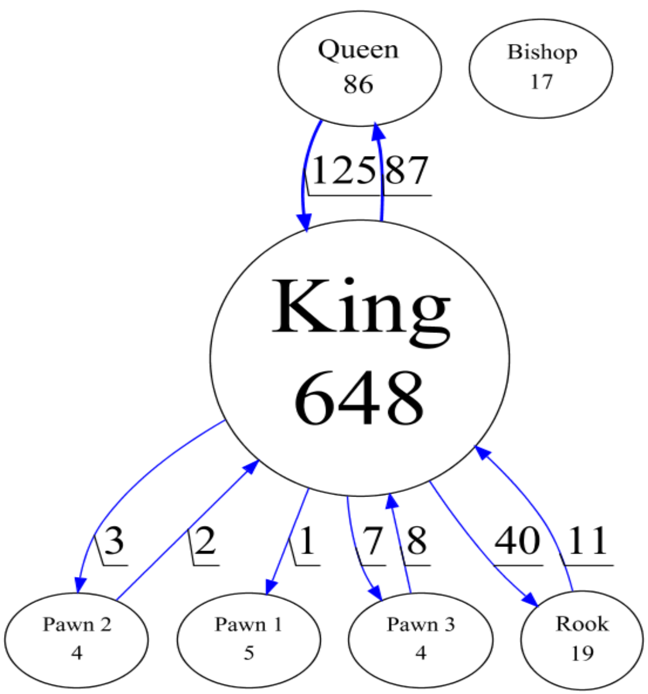


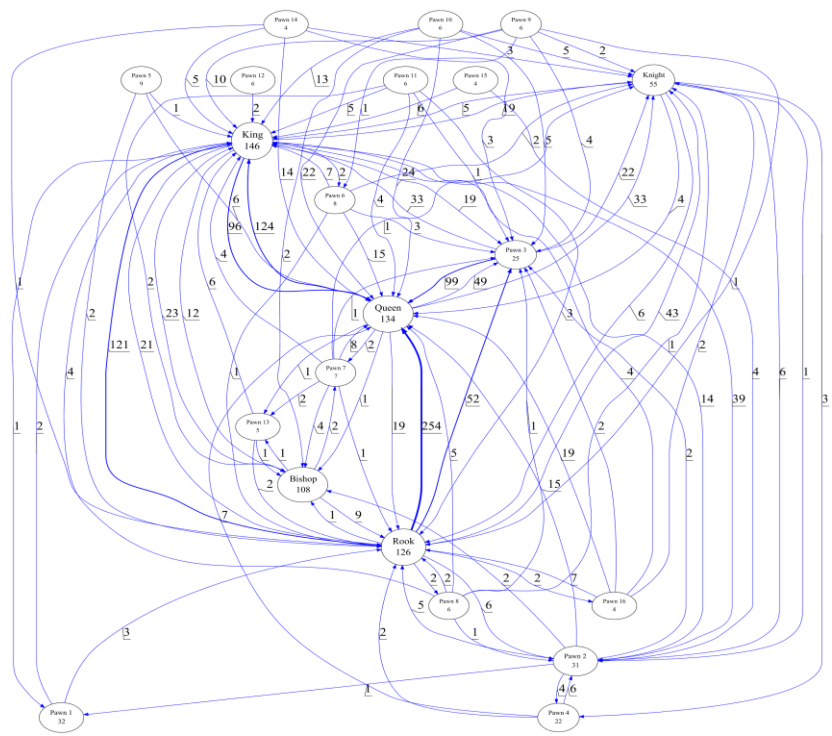
Modular Structure based on Organizational Structure

MD PnP

Open Wrt

OpenWrt looks far more complicated compared to MD PnP!

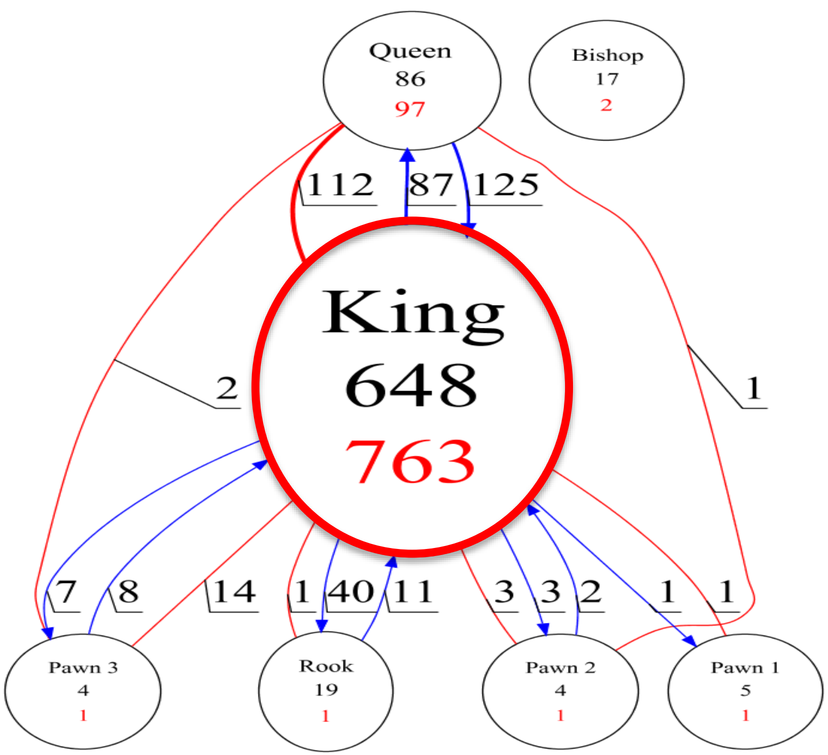




Modular Structure based on Organizational Structure

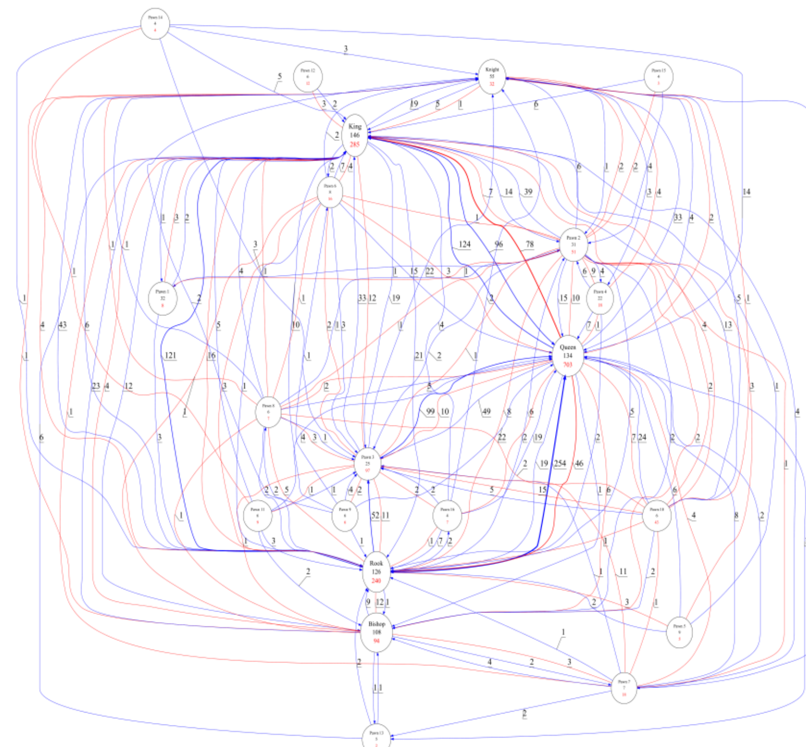
MD PnP

- Pros: 87% inner module changes, 12% cross module changes.
- Cons: King: 45% of files, but 77% of changes



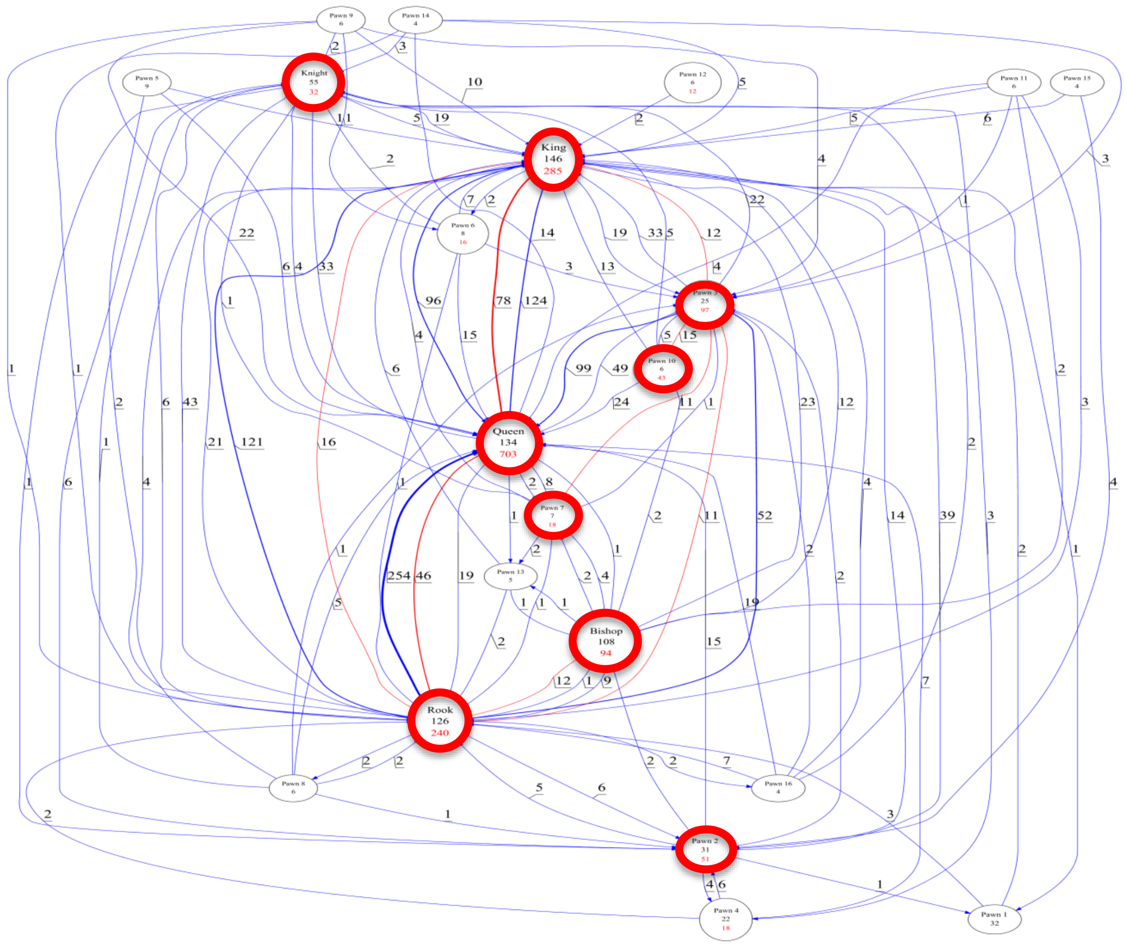
Open Wrt

- Pros: Everyone contributes relatively equally to the system.
- Cons: The modules are more coupled with each other.



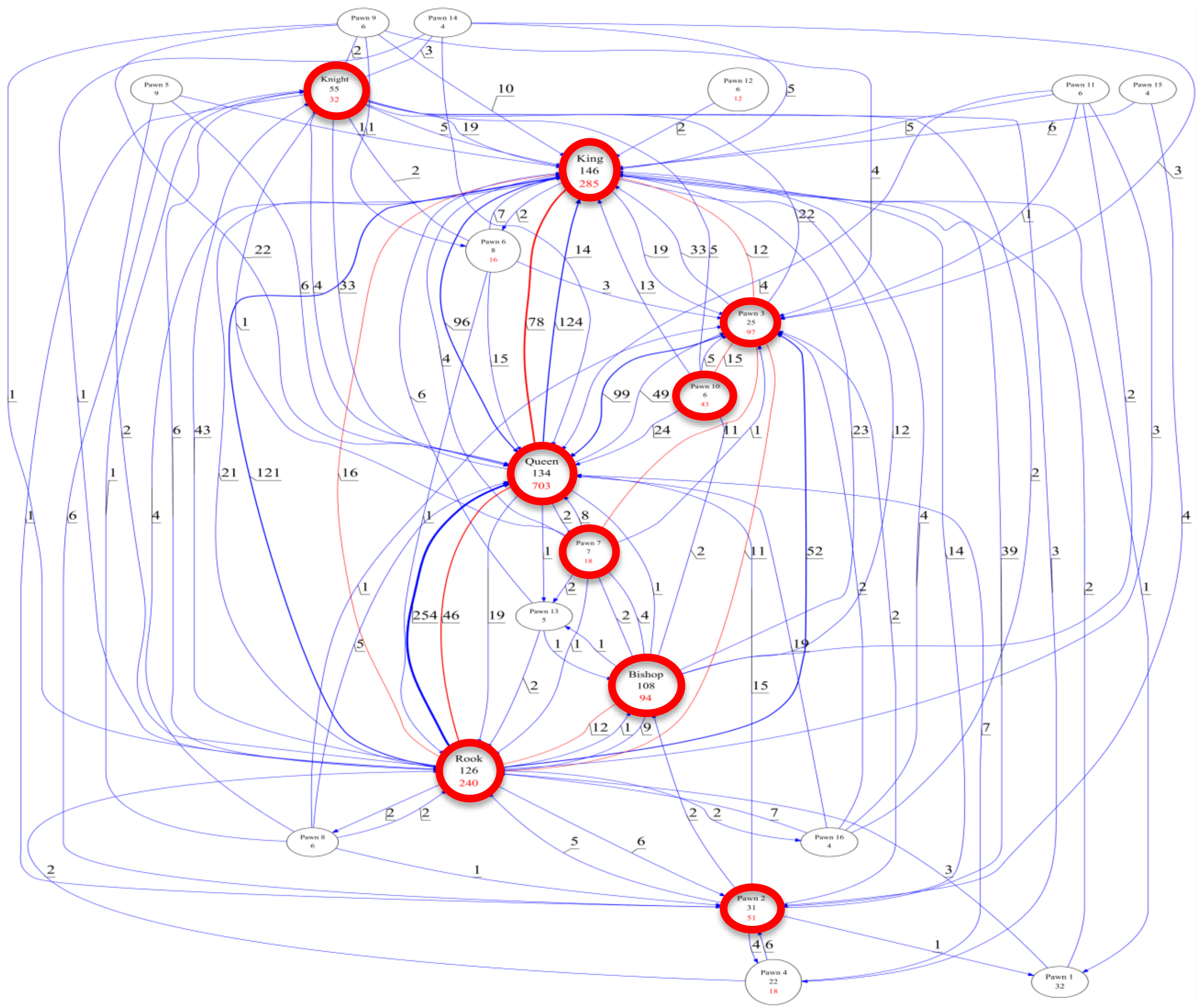
Modular Structure based on Organizational Structure

- We filter out co-change relationships whose weight is less than 10
- Only 9 contributors' modules have co-change weights of 10 or more



Modular Structure based on Organizational Structure

- Vendor lock-in is most likely to happen among these nine modules!



Agenda

- Introduction
- Module decomposer
  - Package view (development view)
  - Dependency hierarchy view (sequential work allocation)
  - Organizational view (vendor-lock in)
- ***Domain concept learner***
- Next Steps

Domain Concept Learner Overview

- Problem: Each cyber-physical system may use different keywords to identify hardware related components and concepts
- Objective: extract semantic relationships among keywords to identify system specific terms and concepts
- Accomplishing this objective will enable us to:

1. Extract hardware related terms for identifying hardware related modularity violations
2. Construct a view of the system based on the semantic structure to find additional candidate modularity violations
3. Enable assisted analysis of new cyber-physical systems through machine learning

SYSTEMS ENGINEERING RESEARCH CENTER

Approach

- Use natural language processing (NLP) techniques to analyze project documentation and organize keywords into topics
- Extract hardware related terms for use in co-change analysis
- Identify relationships among topic groups to extract a semantic structure for the project
- Map the semantic structure to the software architecture to identify potential modularity violations
- To evaluate the efficacy of the approach, OpenWRT will serve as the training set for any developed algorithms, and MDPnP will serve as the test set

Progress to Date

- Scraped OpenWRT project documentation from project wiki pages as well as github change logs to serve as the corpus
- Applied LDA and related algorithms to the corpus but results were not useful
- Applied word2vec skip-gram algorithm to corpus and obtained a useful model
- Extracted hardware related terms using cosine distance metrics applied to the fit word2vec model
- Applied agglomerative clustering to the word2vec model but did not obtain useful clusters
- Applied k-means clustering to the word2vec model and obtained useful clusters of terms by concept

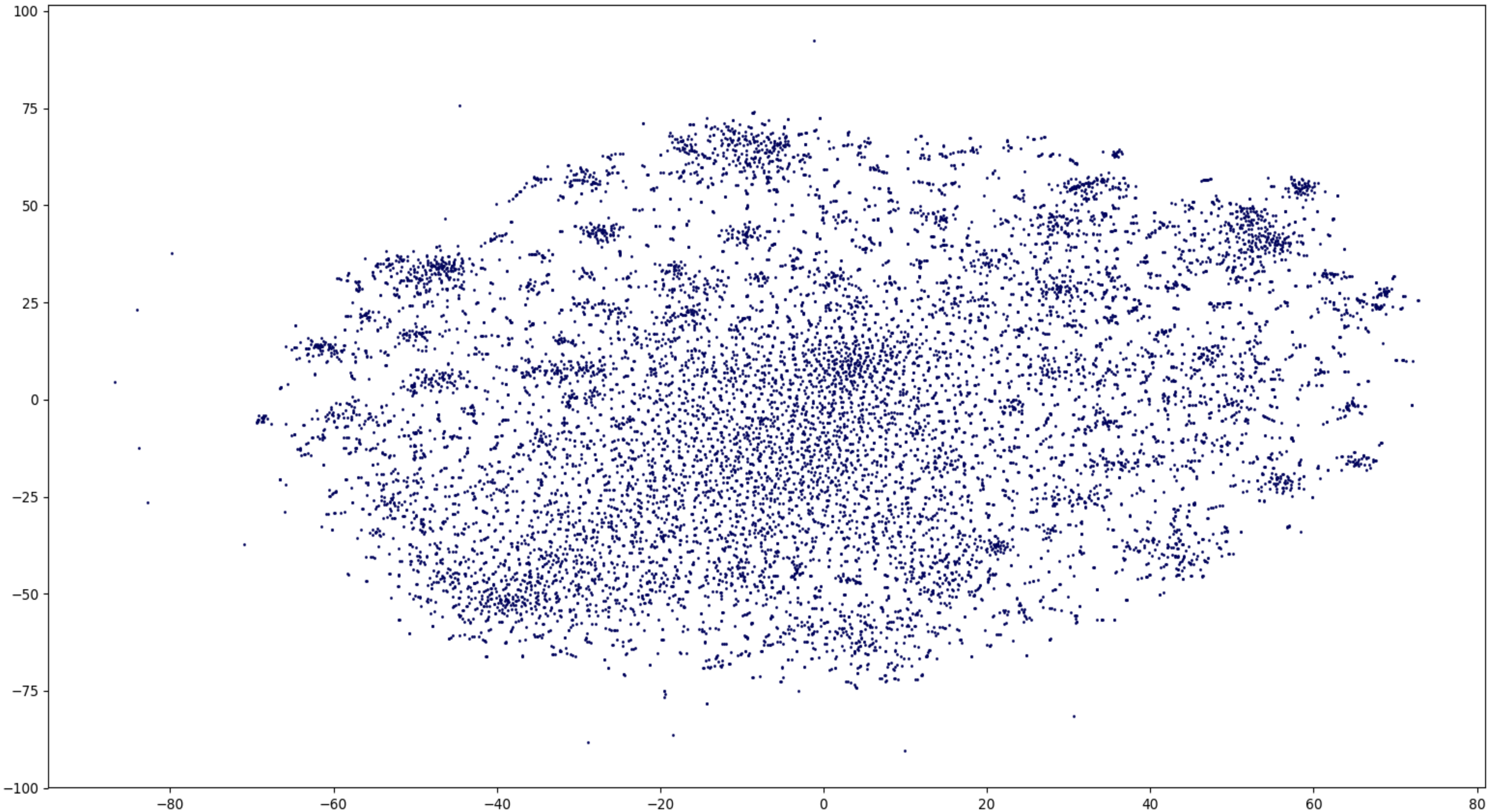
OpenWRT Data

- After pre-processing:
  - 2,318,673 raw words in the corpus
  - 409,283 sentences in the corpus
  - 28,135 unique word types in the vocabulary
- After applying minimum count of 5 occurrences:
  - 11,952 unique word types in the vocabulary
  - 2,289,887 words remain in the corpus

Word2vec

- Word2vec is shallow neural network that attempts to model the probability that words will occur near each other in text
- A consequence of the training process is that each word in the vocabulary is represented by multi-dimensional vector
- Applying a distance metric to a pair of vectors can quantify the degree of similarity among the words that the vectors represent

Fit word2vec Model



t-SNE plot of the trained word2vec model with 500 hidden nodes using cosine distance

Extracting Hardware Related Terms

- Comparisons of the trained vectors enable us to “query” the model for keywords of interest
- We can include both positive and negative words in the query
- For example:

```
word_distances = model.most_similar(positive = ["hardware", "device","router",
"radio", "wifi", "mips", "ramips", "mtd",  "broadcom", "routerboot", "router",
"firmware", "bluetooth", "energy", "power", "soc", "chip" ], negative = ["api",
"call", "class", "code", "readability", "style", "data", "function", "gdb",
"infinite", "loop" , "bug" , "json", "kernel", "leak", "method", "null",
"parameter", "plugin", "process", "recursive", "script", "string", "syscall",
"variable"], topn = False)
```

Extracting Hardware Related Terms

board	cf	plus	verdex
profile	ep93xx	fi	qca9563
at91	zyxel	rt5350	dk01
ehci	techdata	udc	agl300nh
netgear	compex	ar9331	k330
linksys	mt7620a	imx23	305x
cns21xx	omap35xx	routerboard	rb750up
mikrotik	qualcomm	wi	u7623
rt3883	pro	extender	ls1043ardb
rt288x	ata	amcc	aga
ppc40x	openmesh	meraki	awake
apm821xx	gumstix	dlan	sc16is752
pxa	alice	pirelli	mx60w
buffalo	huawei	gate	mt7621a
avila	rb1xx	devolo	7links

Top 60 words from query

Topic Clustering

- Clustering has resulted in reasonably coherent groupings

processors and chipsets	communications	project infrastructure	software management	code organization	storage
0	1	2	3	4	5
bcm2708	patch	project	package	config	mtd
orion	generic	guide	utils	base	data
ppc40x	kernel	http	crypto	etc	flash
mx	kmod	documentation	ltq	sh	nand
timer	pending	lede	atm	lib	info
pxa	backport	welcome	dsl	init	partition
smp	f	wiki	yaffs2	ipkg	size
asoc	hack	org	iwinfo	uci	nvram
cf	v4	forum	swconfig	diag	block
pi	fo	downloads	fw	preinit	chip
jz4740	filter	git	ar6000	conf	map
fsl	optional	com	libpcap	hotplug	m25p80
compatible	ledtrig	binding	libnl	share	rootfs
cpufreq	reduce	release	tiny	bin	squashfs
ipq8064	increase	doc	e100boot	sbin	mount
ipq4019	sched	www	app	usr	write
arm64	netdev	cortex	vdsl	skeleton	parser

Example k-means clustering run for 30 clusters, normalized vectors

Refining the Clusters

- We are currently running parametric experiments and testing various clustering approaches to refine the results
- We are also mapping the clusters to the software architecture

2	3	4	5	6	7
linux	image	file	package	add	patch
target	support	default	openwrt	fix	es-3
generic	device	use	makefile	update	es-4
es-2	board	user	config	build	-default
ar71xx	usb	data	network	remove	pending-4
ramips	driver	option	base-files	version	international
brcm63xx	profiles	configuration	etc	make	submitting
lantiq	platform	interface	src	change	alike
brcm2708	switch	http	lib	documentation	pagesource
adm5120	ethernet	server	control	upgrade	attribution-share
brcm47xx	wifi	set	ipkg	content	lzma-loader
mtd	phy	port	net	lede	swconfig
ath79	code	rule	modules	page	backport-4
ixp4xx	register	using	services	missing	coldfire
ipq806x	gpio	start	init	new	map
s3c24xx	wireless	address	scripts	enable	uml

Extracting the Semantic Structure

- We are tracing how words are grouped as the number of clusters is varied in order to extract relationships among them

Clusters	0	1	2	3	4	5	6	7	8	9
Module terms	6rd	uboot-ar71xx	linux			package		swconfig	kernel	include
	gpio-button-hotplug	w1-gpio-custom	target			config		map	tools	boot
	button-hotplug	rbcfg	generic			network			utils	
	otrx	trelay	ar71xx			scripts			system	
	rssileds	owipcalc	ramips			ppp			hostapd	
	flock	resolveip	brcm63xx			broadcom-wl			libs	
		padjffs2	lantiq						openssl	
		rtc-rv5c386a	adm5120						firmware-utils	
		iwcap	brcm47xx						musl	
		usbreset	mtd						glibc	
		rotary-gpio-custom	ath79						ead	
		ixp4xx-microcode	ar7						libnl-tiny	
		avila-wdt	firmware						mtd-utils	
		leds-apu2	at91						px5g	
		spidev_test	cns3xxx						mklibs	
		oseama	oxnas						gettext	
		fbtest	nvram						uboot-oxnas	
		fritz-tools	mcs814x						libiconv	
		maccalc	adm8668						sstrip	
		fwtool	mpc85xx							
		spi-gpio-custom	apm821xx							
		patch-image								
		osafeloader								

Clusters	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Module terms	map	fritz-tools		tools		rssileds	linux		package	broadcom-wl		lantiq	config	generic	
		maccalc		utils		resolveip	target		network	swconfig			kernel	ramips	
		fwtool		hostapd		padjffs2	ar71xx		scripts	ead			include	brcm63xx	
		spi-gpio-custom		libs		flock	adm5120		ppp	6rd			mtd	brcm47xx	
		patch-image		openssl		rotary-gpio-custom	ath79			gpio-button-hotplug			boot	at91	
		osafeloader		musl		ixp4xx-microcode	adm8668			uboot-oxnas			system	cns3xxx	
				glibc		leds-apu2				libiconv			ar7	oxnas	
				libnl-tiny		spidev_test				uboot-ar71xx			firmware	mcs814x	
				mtd-utils		oseama				button-hotplug			nvram	mpc85xx	
				px5g						otrx			firmware-utils	apm821xx	
				mklibs						w1-gpio-custom					
				gettext						rbcfg					
				sstrip						trelay					
										owipcalc					
										rtc-rv5c386a					
										iwcap					
										usbreset					
										avila-wdt					
										fbtest					

Agenda

- Introduction
- Module decomposer
  - Package view (development view)
  - Dependency hierarchy view (sequential work allocation)
  - Organizational view (vendor-lock in)
- Domain concept learner
- ***Next Steps***

Next Steps

- Short-term:
  - Cross reference learned domain concepts to modules.
  - Identify and measure modularity violations at different levels of decomposition for different stakeholders.
  - Build proof-of-concept demonstrator.

- Long-term:
  - Prioritize and visualize modularity violations for restructuring decision-making for stakeholders.
  - Provide in-depth interpretation of the root causes of modularity violations for restructuring insights.

Thank You!

Lu Xiao;
lxiao6@stevens.edu
Michael Pennock;
mpennock@stevens.edu

School of Systems and Enterprises
Stevens Institute of Technology