SYSTEMS ENGINEERING RESEARCH CENTER

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

Sponsor: DASD(SE)

By
Dr. Lu Xiao and Dr. Michael Pennock
11th Annual SERC Sponsor Research Review
November 19, 2019
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW, 8th Floor
Washington, DC 20009

www.sercuarc.org

Agenda

- Introduction
- Dataset: OpenWrt and MD PnP
- Module Decomposition
- Domain Concepts Extraction
- Modularity Violation Analysis
- Future Plan

Agenda

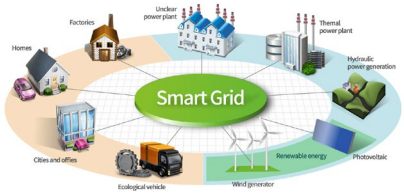
- **Introduction**
- Dataset: OpenWrt and MD PnP
- Module Decomposition
- Domain Concepts Extraction
- Modularity Violation Analysis
- Future Plan

Cyber-physical Systems (CPS)

- A cyber-physical system is an integration of computation with physical processes

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

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



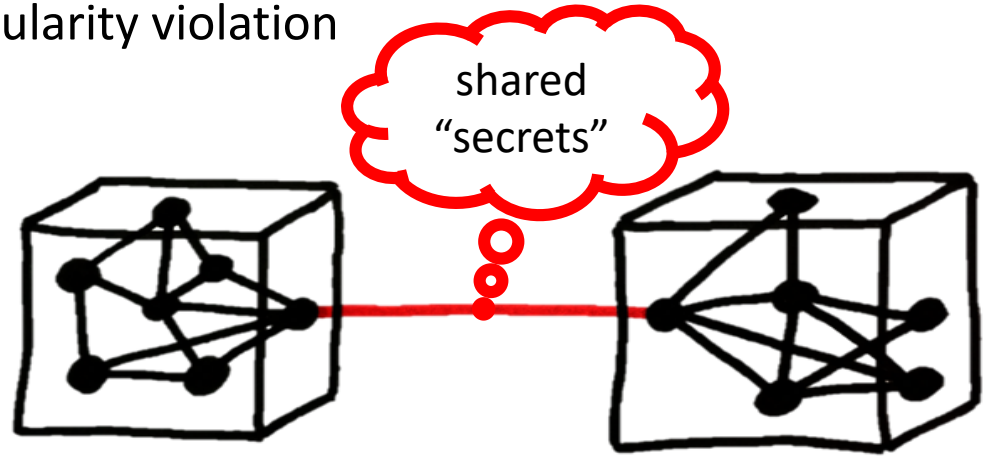
Modularization

- Given that cyber-physical systems evolve over time, they need to be robust to changes in their extent and composition
- One way to achieve this is to employ a modular design
  - —Ideally, the system can be assembled from “plug and play” components
  - —Improve interoperability
  - —Facilitate system evolution and technology insertion
  - —Avoid vendor lock-in

- When companies and governments acquire or use a cyber-physical system, how do they know that they are getting what they are paying for?

Modularity Violations

- Poor design and development practices may lead to latent dependencies among modules called “Modularity Violations”
  - —e.g., an implicit assumption about time units in multiple modules could be a modularity violation



- Furthermore, vendors may be incentivized to make it difficult to swap their components with those from other vendors
- Customers and stakeholders need a way to measure whether a cyber-physical system is as modular as advertised

Research Motivation

- The DoD has emphasized modularity in the systems it acquires
  - E.g., Modular Open Systems Approach (MOSA)
- The challenge is that it is extremely difficult to determine if acquired cyber-physical systems are modular
  - A system with nominally modular architecture can have latent modularity violations in the real system
  - This inhibits system evolution and maintenance as well as promotes vendor lock-in
- Previous research has resulted in methods for detecting modularity violations in pure software systems
- The motivation behind this research is to determine if those techniques could be used to infer modularity violations for cyber-physical systems

Approach: Data-Driven and Intelligent

1. Use the repositories, change histories, and documentation of two open source cyber-physical projects as test cases

2. Develop and apply alternative methods for extracting system architectures from different perspectives from the repositories

3. Adapt prior research from the software engineering domain to develop and apply metrics to detect, measure, and prioritize modularity violations

4. Apply *natural language processing (NLP)* to analyze project documentation and change histories and extract key project related terms and concepts

5. Apply *NLP* models to assess the feasibility of detecting hardware related issues through the software change repositories

6. Merge the NLP models, decompositions, and metrics to analyze the two projects and identify key hardware and software related modularity issues

Agenda

- Introduction
- **Dataset: OpenWrt and MD PnP**
- Module Decomposition
- Domain Concepts Extraction
- Modularity Violation Analysis
- Future Plan

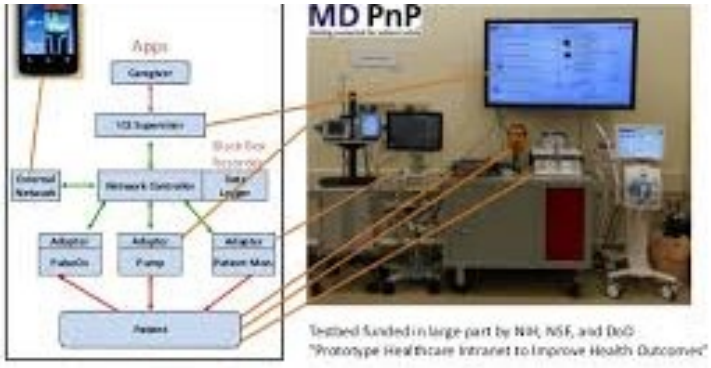
Case Study Subjects

- 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/

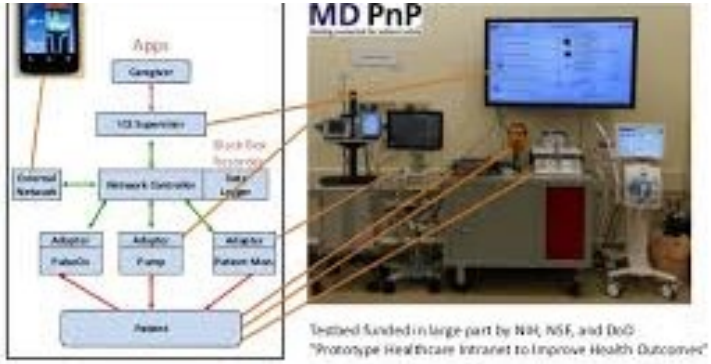
- Md PnP: The medical device "Plug-and-Play" interoperability program advancing safe and secure interoperability to improve patient care.

  —http://www.mdpnp.org/



Case Study Subjects

- OpenWrt
  - —1063 source files (in c)
  - —80 developers
  - —42018 commits
    - 1996 commits on source files
    - 40k commits on non-source files
- Md PnP:
  - —808 source files (in java)
  - —7 developers
  - —1611 commits
    - 993 commits on source files
    - 618 commits on non-source files



Agenda

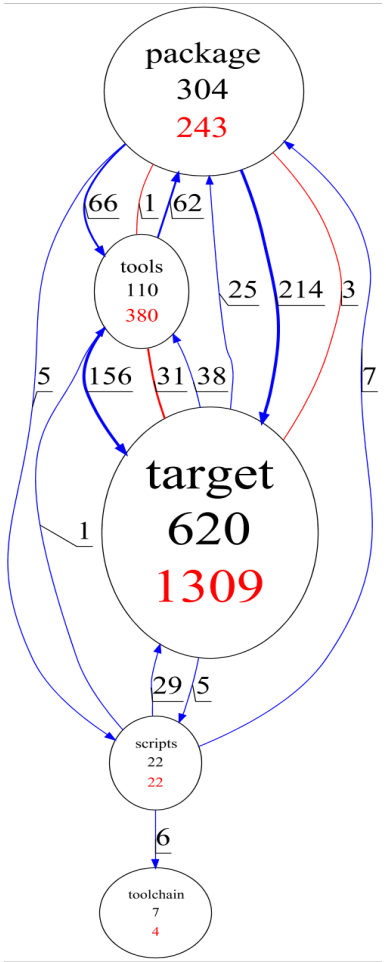
- Introduction
- Dataset: OpenWrt and MD PnP
- **Module Decomposition**
- Domain Concepts Extraction
- Modularity Violation Analysis
- Future Plan

Module Decomposition

- In this study, we explored three different criteria to decompose a large-scale, complex system into modules.
  - —The development modular structure
  - —The dependency hierarchy structure
  - —The organizational structure

- Modularity violations identified based on different modular decompositions reveal different perspective of concerns.

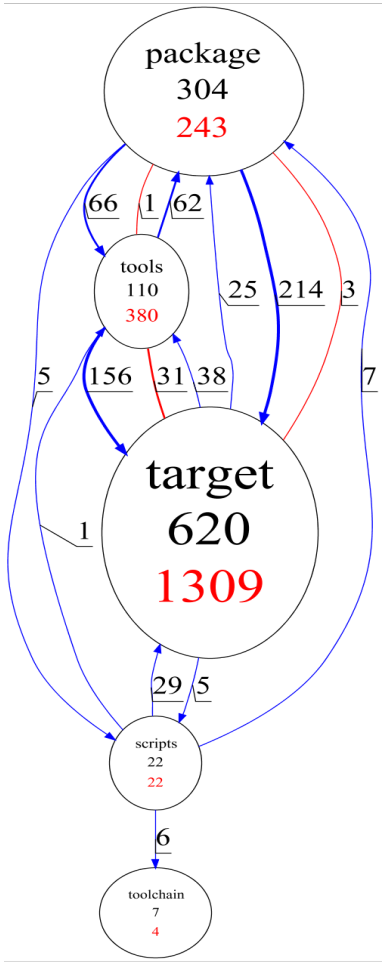
Illustration of Modular View



- Each oval represents a module based on a selected decomposition criterion.
  - The number in black indicates the number of source files
  - The number in red indicates the number of inner module changes
- The edge represents the relationship between two modules
  - Edge in blue indicates the number of structural dependencies by implementation (directed).
  - Edge in red indicates the number of cross module changes (bi-directional).

OpenWrt- Root Development View

Measurement of the Modular Structure



OpenWrt- Root Development View

- We propose a new set of measurements to evaluate the modular structure of a system:
  - **Cross Module Change (CMC):** the percentage of changes happened cross different modules. The higher the value, the more severe modularity violations.
    - CMC = 2%: *Only 2% changes imply modularity violations.*
  - **Inner Module Change (IMC):** the percentage of changes happened within modules. The higher the value, the better the modular structure.
    - IMC = 98%: *The modules mostly (98%) evolve independently!*
  - **Value for Cost (VfC)** for a Module: the number of changes on a module divided by the size of a module. The higher the value, the more expensive it is to maintain a particular module.
    - VfC_tools = 19%/10% = 2: *Module "tools" is very expensive to maintain.*

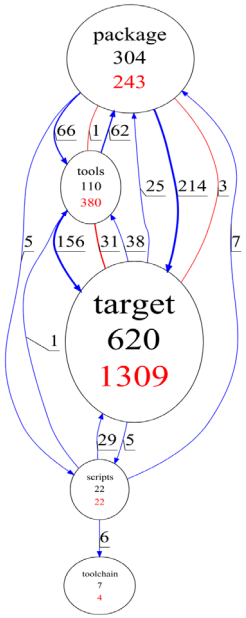
Measurement of the Modular Structure

- The measurements applied based on different decomposition criteria reveal different aspects of modularity violation concerns.

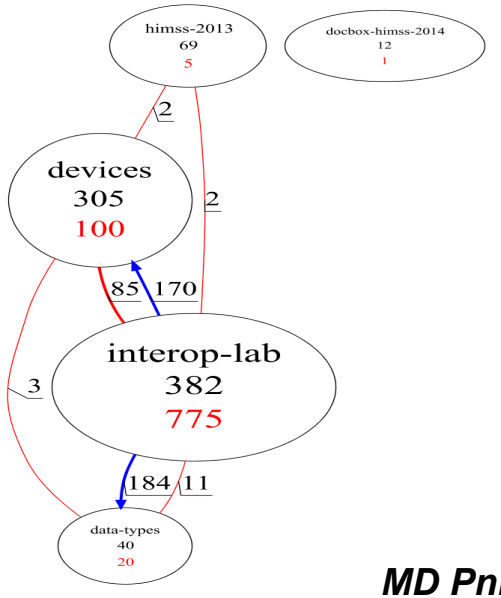
Criterion	CMC and IMC	VfC
Development	How the system can evolve as truly independent modules/subsystems?	Which module/subsystem is the most expensive to maintain?
Hierarchical	How likely the system can be developed as sequential tasks without rework?	Which design elements form the largest blocking points?
Organizational	How likely vendor lock-in will happen?	Which vendor takes the largest amount of maintenance costs?

Root Development Modular Structure

- The modules mostly (>90%) evolve truly independently!
- Each project has modules that are expensive to maintain given its size.



OpenWrt

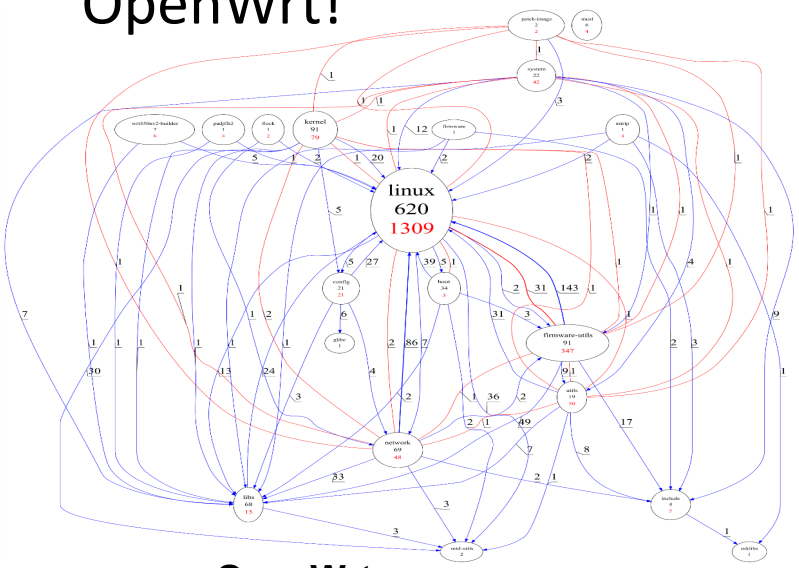


MD PnP

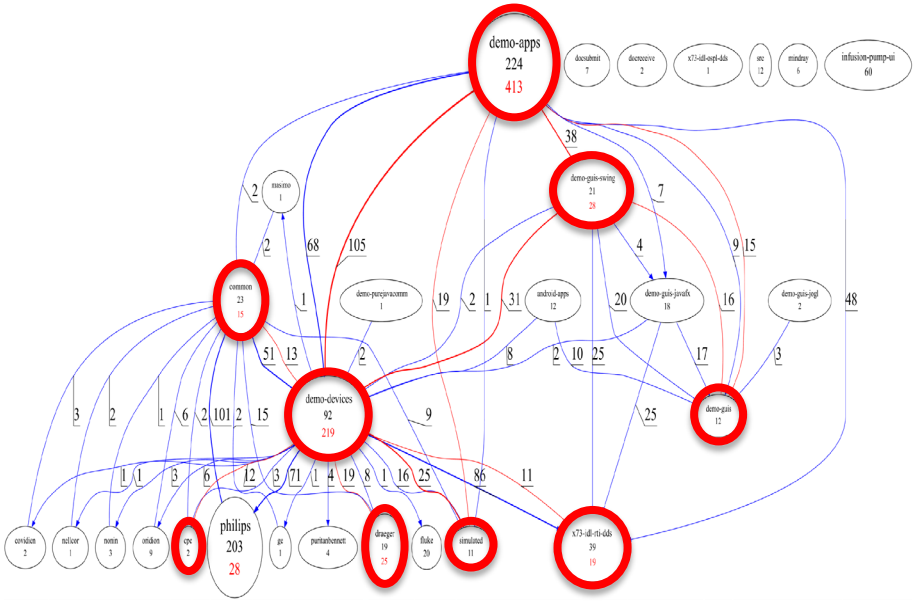
Metric	OpenWrt	MD PnP
CMC	2%	10%
IMC	98%	90%
VfC_max	"Tools": 19%/10% = 2	"interop-lab": 86%/47% = 1.8

Fine-grained Development Modular Structure

- In the fine-grained development view, MD PnP seems to suffer a lot more from modularity violations compared to OpenWrt!



OpenWrt

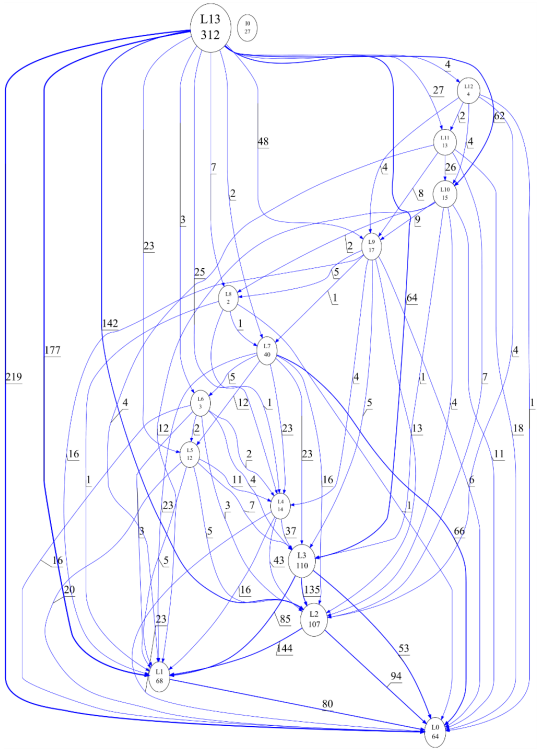


MD PnP

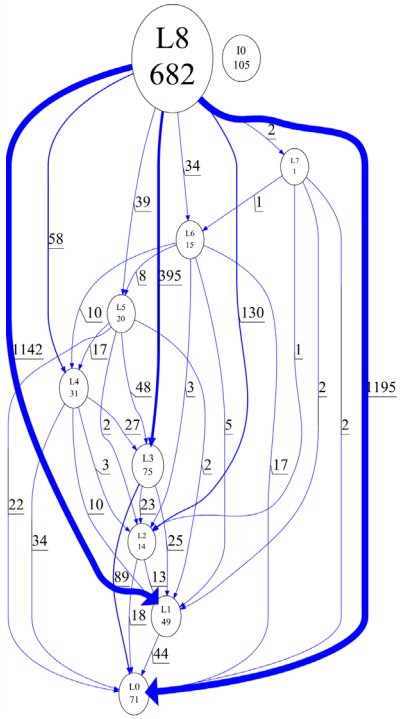
Metric	OpenWrt	MD PnP
CMC	3%	**52%**
IMC	97%	**48%**
VfC_max	“firmware-utils”: 18%/9% = 2	“demo-apps”: 51%/28% = 1.8 “demo-devices”: 27%/11% = 2.5

Hierarchical Modular Structure

- Modules can be decomposed based on the dependency hierarchy. Design elements in the upper modules depends on elements in the lower modules
- This view implies sequential/parallel work assignment.



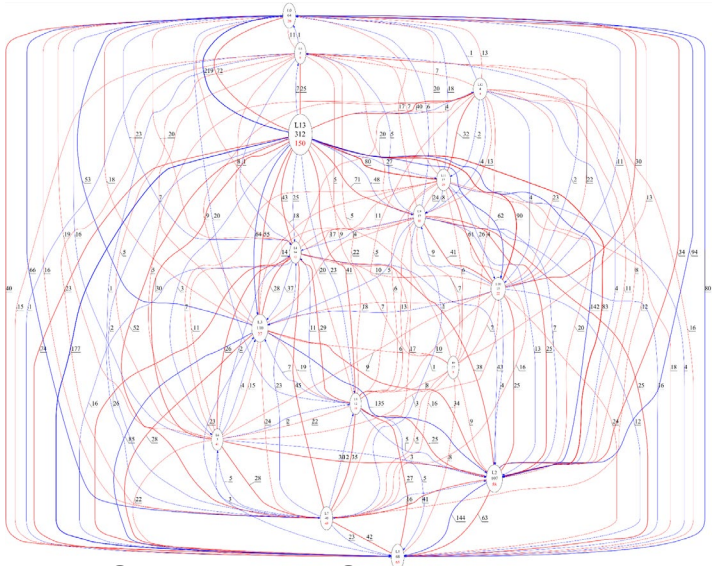
OpenWrt: 13 Sequential Layers



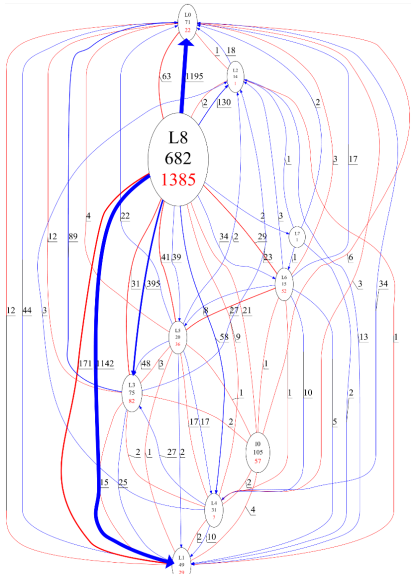
MD PnP: 8 Sequential Layers

Hierarchical Modular Structure

- **Reality:** modules cannot be truly sequential, but requiring non-trivial (20% and 16%) rework!
- The VfC measurement points to the bottleneck!



OpenWrt: 13 Sequential Layers

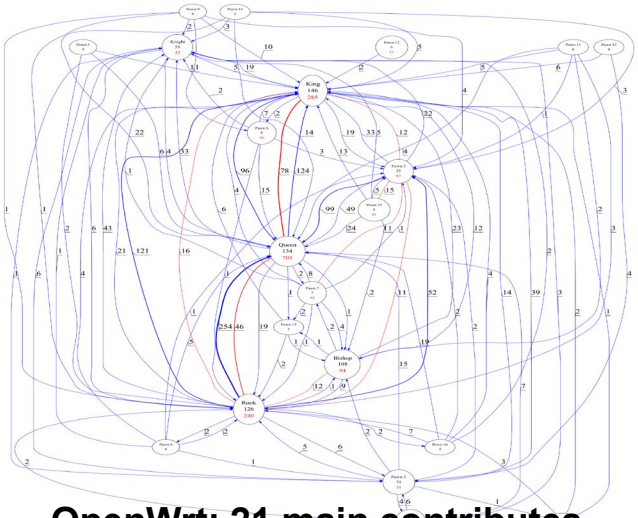


MD PnP: 8 Sequential Layers

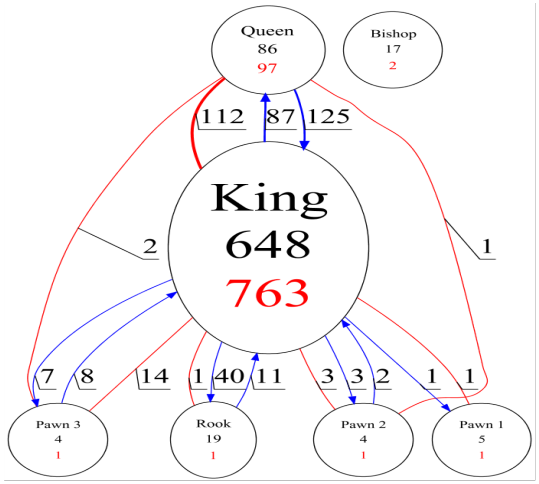
Metric	OpenWrt	MD PnP
CMC	16%	20%
IMC	84%	80%
VfC_max	"L8": 82%/64% = 1.3	Always below 1

Organizational Modular Structure

- Non-trivial (32% and 17%) number of chances require collaboration among different vendors.
- The points of risk are the “King” and “Queen” modules.



OpenWrt: 21 main contributes



MD PnP: 7 main contributors

Metric	OpenWrt	MD PnP
CMC	17%	32%
IMC	83%	68%
VfC_max	“Queen”: 42%/18% = 2.3	“Queen”: 13%/10% = 1.25

Agenda

- Introduction
- Dataset: OpenWrt and MD PnP
- Module Decomposition
- **Domain Concepts Extraction**
- Modularity Violation Analysis
- Future Plan

Domain Concept Learner

- The modular decomposer helps us to understand the structure and changes from different perspectives.
- To find out what hardware concepts trigger the software changes, we need to analyze the **hardware concepts** embedded in the documentation, change logs, and source code.
- Therefore, we developed a concept learner to automatically learn and extract hardware related concepts in CPS.

Domain Concept Learner Approach

- Part 1: Training: From Project Data to Word2Vector
  - Extract and the learn the relationships among project specific terms using the word2vec algorithm
  - Measure the relevance of terms to software and hardware concepts using cosine distances computed from the fit word2vec model
- Part 2: Application: Word2Vector to Relevance Score
  - Calculate scores and categorize source files/changes based on the measured similarity to hardware and software concepts
  - The higher the score, the more likely that an artifact is hardware related
  - Categorization thresholds are based on an analysis of empirical data

Domain Concept Learner

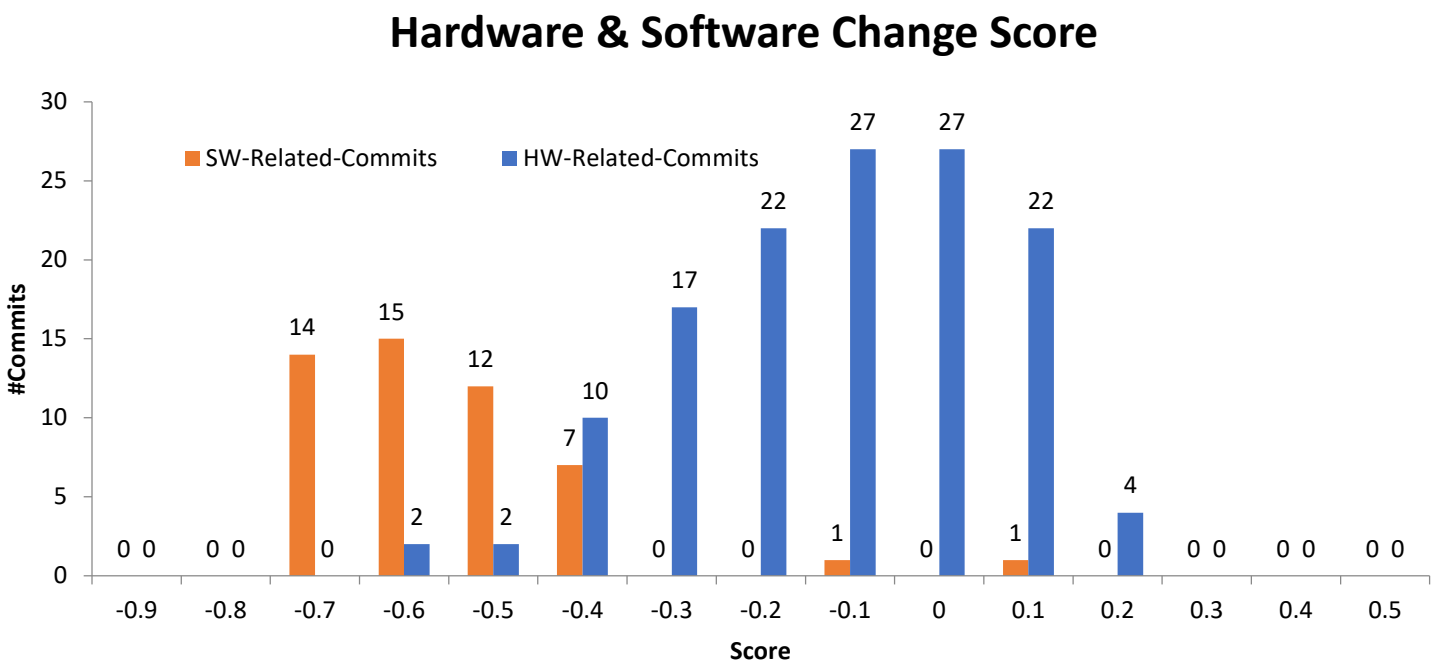
- We applied the domain concept learner to 42,250 total change messages in OpenWrt
  - —"get rid of NVRAM_SPACE Now we support NVRAM whatever its size is"
- This was a test to determine if the concept learner could accurately assess if an item of interest was hardware or software related
- If the domain concept learner works, it should be able to group the evaluated items as hardware related, software, related or unknown/mixed.

Domain Concept Learner Verification

- We randomly selected and verified 264 messages from the 42250 change messages.
- Without looking at the score, we manually categorized each message into three types:
  - **Software related (19%):** *scripts/symlink-tree.sh: print warnings/errors to stderr Makes warnings/errors visible when building with V=w/V=1.*
  - **Hardware related (50%):** nvram: *get rid of NVRAM_SPACE Now we support NVRAM whatever its size is*
  - **Unknown (31%)**:
    - Mixing terms: running an ebtables executable linked this way just crashes with a segmentation fault at runtime on program startup, e.g. on ARM architectures.
    - Lacking information: *hopefully fix duplicate ppp instances (#895)*
    - Cannot understand: *ipt-extra not broken on kernel 2.6.28 any more as CHAOS, TARPIT and DELUDE references were removed in r14461 SVN-Revision: 14779*

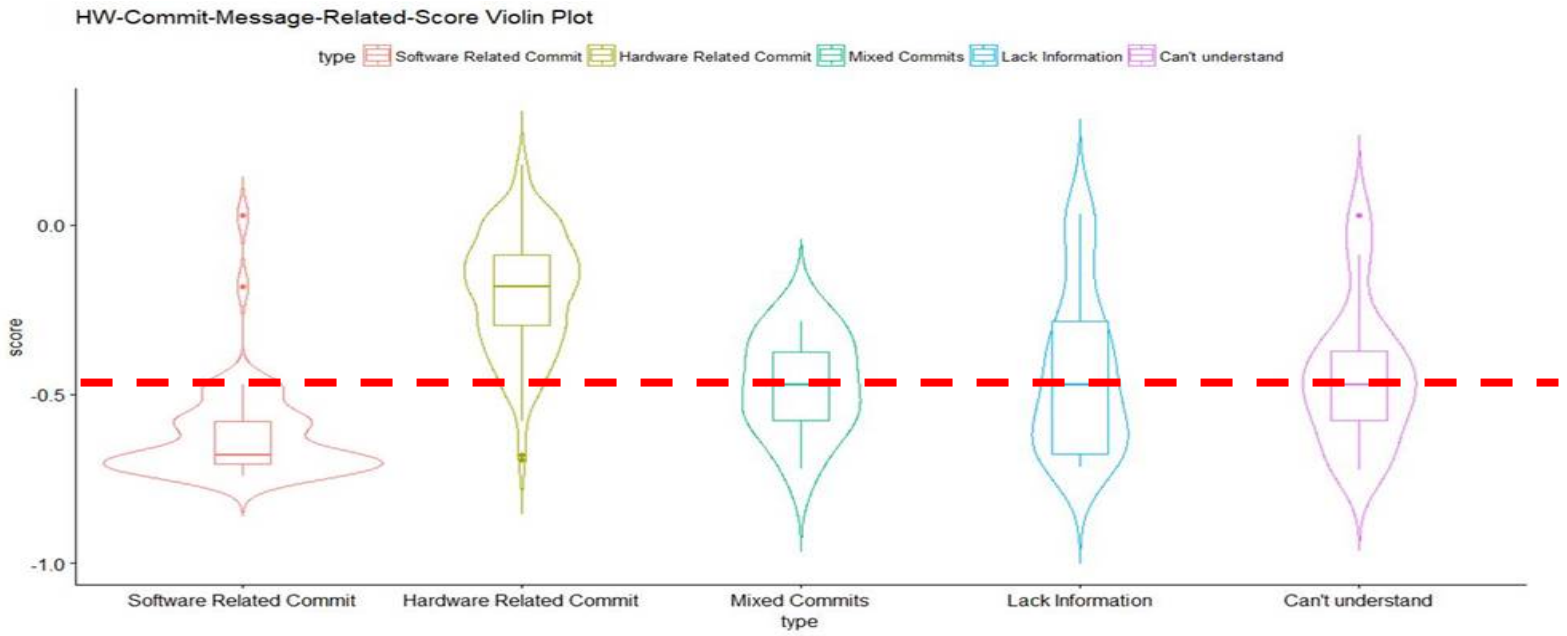
Score for Change Messages

- The relative value of scores effectively separated hardware and software terms in OpenWrt change messages.



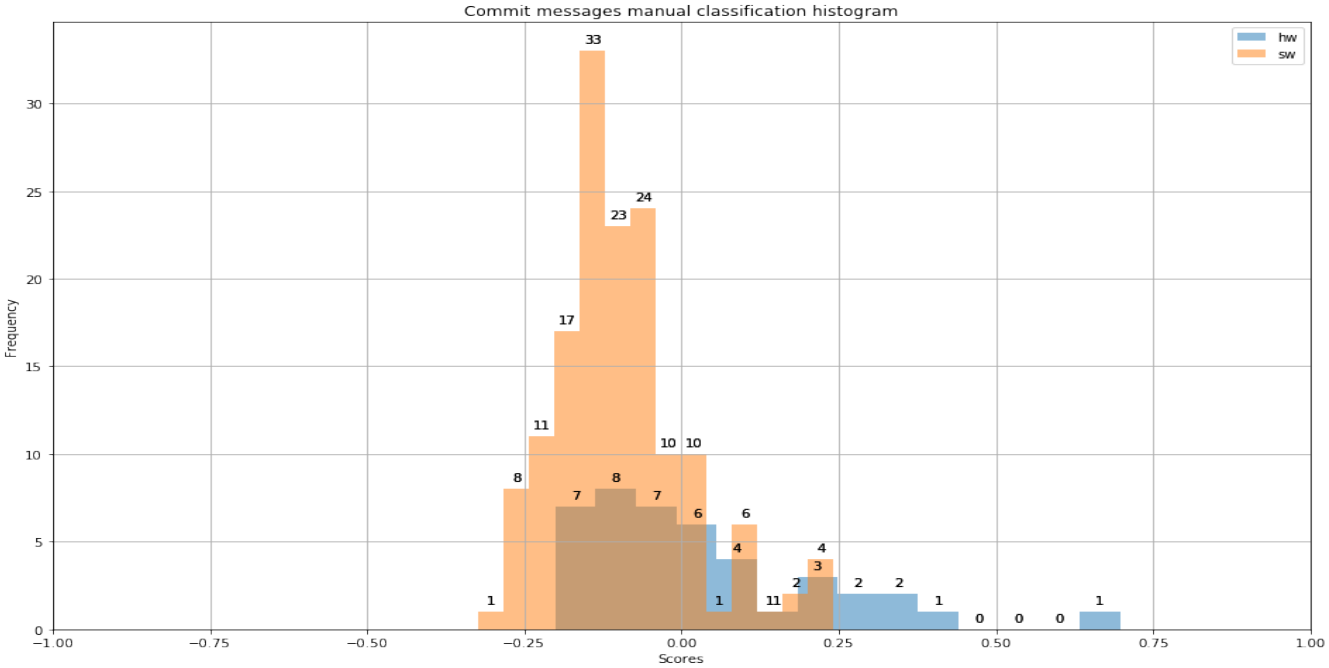
Categorizing Changes

- The domain concept learner is effective at categorizing changes.
- Based on empirical data, we use **threshold of -0.47** to separate software and hardware concepts.



Limitations for MD PnP

- The domain concept learner was **not** effective at categorizing file names: many file names are terse.
- We also applied the same approach on MD PnP, the software and hardware/domain concepts are not as clearly separated (264 messages manually verified):



Manual Inspection of MD PnP

- We manually examined all the 1600 change messages from MD PnP and categorized the messages into the following categories:
  - —48% changes are related to software concepts
  - —Only 16% changes are related to hardware concepts
  - —7% changes happened due to combined software and hardware concepts

Number of selected commit messages	1600	100%
Hardware	255	16%
Mixed-Hardware and Software	104	7%
Software	762	48%
Unsure	120	8%
Administrative	162	10%
Lacking Information	197	12%

Limitations

- Why Concept Learner is not working very well in MD PnP ?
  - —OpenWrt has very clear separation of hardware and software concepts
  - —MD PnP's information structure is more complicated for representing the problem domain: software, hardware, and domain related.
  - —MD PnP has much less documentation compared to OpenWrt.

CATEGORY	EXAMPLE
HW	db205f49: adds nellcor, masimo, philips
MIXED	d017d82f: Pass an optional transitionNote with StateMachine transitions. Also do not throw RuntimeException from Serial Intellivue when serial port unavailable. Instead connect now returns false and the process terminates gracefully.
SW	a572d2c3: Adds support for builtin topics to InstanceModelImpl and also support for iterating without registering for callbacks.
NOT SURE	1dae724d: Loosens constraints on participant liveliness. Refines QoS policies for heartbeat samples.
ADMINISTRATIVE	55d09c9d: Merge branch 'master' into patient
MORE INFORMATION REQUIRED	2d594835: placeholders for the future

Agenda

- Introduction
- Dataset: OpenWrt and MD PnP
- Module Decomposition
- Domain Concepts Extraction
- **Modularity Violation Analysis**
- Future Plan

Modularity Violation Analysis (OpenWrt)

- Goal: What are the latent concepts causing changes across modules and across software/hardware components?
- Approach: Combine the results from module decomposer and concept learner
- Data: 542 Changes between June-2016 and July-2018 from OpenWrt
  - —121 (22%) Software Changes (<=-0.47)
  - —421 (78%) Hardware Changes (>0.47)

Category	True Positive	False Positive
Software Changes (12 cases)	**11**	1
Hardware Changes (53 cases)	**50**	3

Fine-grained Development View (OpenWrt)

- Most changes (both hardware and software triggered) happen within modules in the development view.

Organizational	Change Type	Change Count	Change Percentage
Software Change	Inner Module Changes	104	**82%**
	Cross Module Changes	23	8%
Hardware Change	Inner Module Changes	395	**96%**
	Cross Module Changes	16	4%

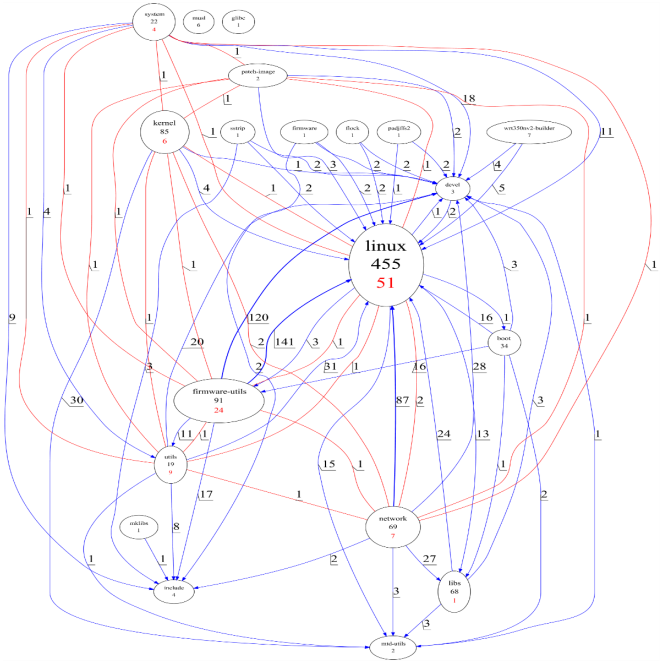
- Some modules are more hardware related and some are more software related.

Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VfC Ratio
linux	455	0.9	1.34	0.67
firmware-utils	91	2.13	1.47	1.45
kernel	85	0.57	0.54	1.06
network	69	0.82	0.48	1.71
libs	68	0.12	0.03	4.00

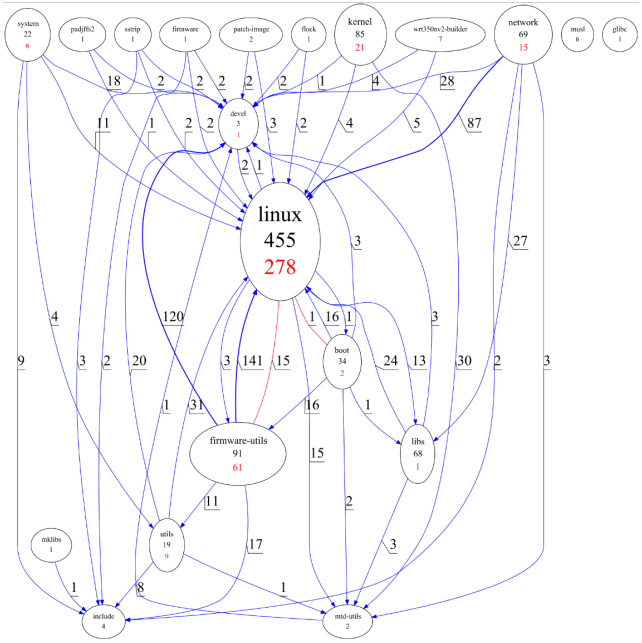
Fine-grained Development View (OpenWrt)

- All of the 16 hardware cross-module changes are between “Linux” and “firmware-utils” or “boot”!!

- 11 of them explicitly mentioned the change based on other chips e.g. TP-Link Archer C7 v4 is a dual-band AC1750 router, and Qualcomm/Atheros QCA9561+QCA9888.



SW-Related Commits



HW-Related Commits

Organizational View (OpenWrt)

- Hardware and software concepts triggered changes are evenly distributed within and across organizational modules.

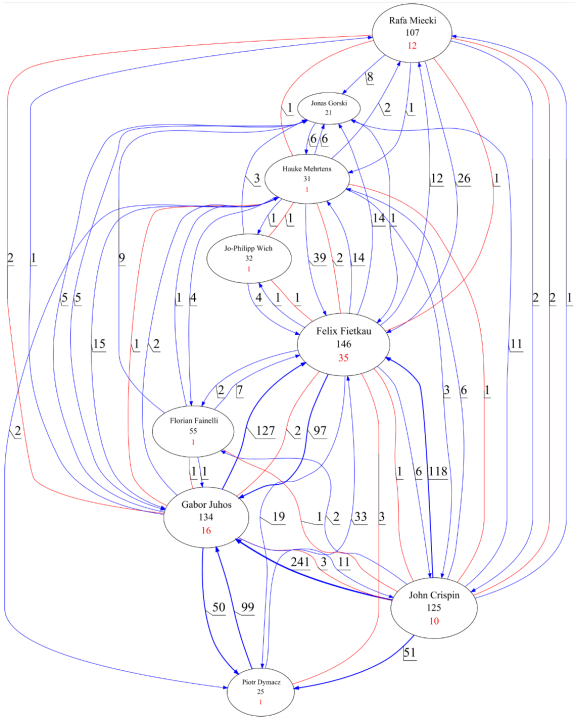
Organizational	Change Type	Change Count	Change Percentage
Software Change	Inner Module Changes	104	59.43%
	Cross Module Changes	71	40.57%
Hardware Change	Inner Module Changes	309	59.65%
	Cross Module Changes	209	40.35%

- There are separate hardware and software expert vendors in the organizational structure.

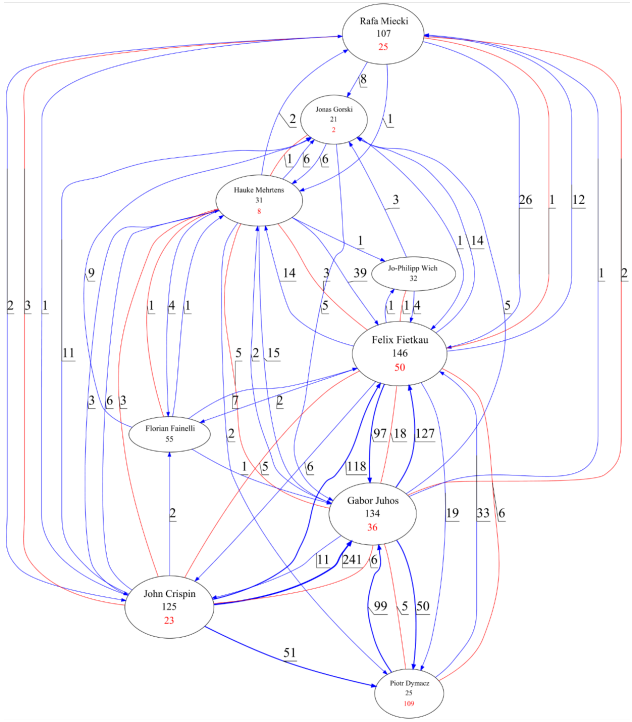
Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VFC Ratio
Vendor 1	151	1.4	0.8	1.75
Vendor 2	130	0.67	0.6	1.12
Vendor 3	107	0.46	0.48	0.96
Vendor 4	32	0.17	0	Null
Vendor 5	31	0.17	0.42	0.40

Organizational View (OpenWrt)

- Shared knowledge underlying modularity violations:
  - Software: project naming and management conventions, e.g. change email addresses, fixing code clone.
  - Hardware: fixing bugs related to chips or add kernel support for more hardware chips.



SW-Related Changes



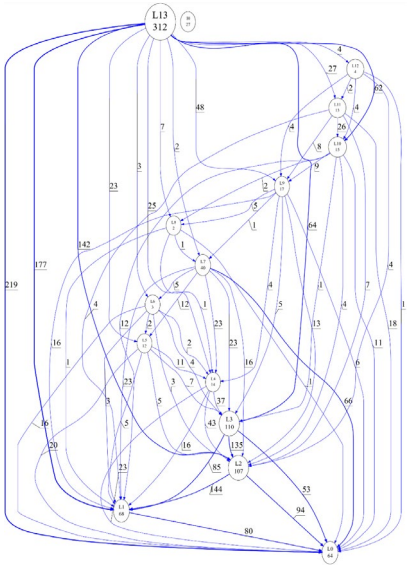
HW-Related Changes

Hierarchical View (OpenWrt)

- Hardware and software concepts are both likely (36% and 25%) to cause changes across organizational modules.

Hierarchical	Change Type	Change Count	Change Percentage
Software Change	Inner Module Changes	104	74.82%
	Cross Module Changes	35	25.18%
Hardware Change	Inner Module Changes	286	63.84%
	Cross Module Changes	162	36.16%

- Lower layers (the most fundamental layer) seems to be more relevant to hardware concepts.



Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VFC Ratio
I0	88	0.1	0.28	0.36
L0	58	0.15	0.32	0.47
L1	23	1.12	1.63	0.69
L2	90	0.48	0.38	1.26
L3	11	1.56	0.85	1.84
L4	20	1.29	0.94	1.37
L5	10	0	1.25	0.00
L6	1	0	0	N/A
L7	594	1.29	1.24	1.04

Hierarchical View (OpenWrt)

- Software changes happens across sequential layers due to house-keeping, bug fixing, and other specific quality goals.
- Hardware concepts trigger changes across sequential layers due to the updates in hardware supports.

SW Cross Module Changes	
b6c8bc0981855b50b7bf22f074	Changing Naming Convention
37ab944ac5752a23fef63b56f6	Changing Naming Convention
fedc826c320d27759ac551c842	Changing Naming Convention
c66658441b78736e5537c10ed	Fix Error Message
7eeb254cc49c4e9f837c980826	Revise Email Address
7a4930b27c2f4d336b4f69a5f4	Clean Code
810659a22c11ea0d2d94f3e09(	Fix Bug
17eb826a703d996d12004b68c	Fix Bug
d40a358136fdc19e6af1392186	Fix Bug
c47a1a3527d988b637c1daee5	Fix Bug
cb7ab730c76c47bbcb1f8ba4cc	Fix Bug
f9e7ffe73be9a45c1b44f8899d9	Fix Bug
3bf35126733a4b4073c1d3388(	Fix Bug
3d58d7f053eae83b59baf5c8b8	Fix Bug
798497c66e6a67d9dbad096c0	Improve Performance
c30a70fc9fbc968fb0563feef0e(	Improve Performance
99a188828713d6ff9c541590b0	Remove Feature

House keeping

Fix bug

Specific goals

HW Cross Module Changes	
4caab808300ad75575d778bae	Add cross chip support
53a45020135b504cb4bee0fa8(	Add cross chip support
ac03d51a3f4daa2f6a2a83f041(	Add cross chip support
b74f63f81d6121b5eace3f0c0c8	Add cross chip support
5c5bf8b8658a588423f6ec445d	Add cross chip support
bf39d5594b3c8f9409e6d9408a	Add cross chip support
a789c0f49159c618d990541c23	Add cross chip support
c0742307a18178ee7d2cee1fc6	Add cross chip support
d265a8f2ca49a67a46493cf57d	Add cross chip support
c2d2647c099679d64c8d5ef52(	Add single chip support
81d446b045176e3e25bb0ef74	Add single chip support
b4bf43c66767648f98494dd5f4	Add single chip support
e36f8b3f3980903d5cefc51fe27	Add single chip support
f7aa08595db07f4c56484b3cef(	Add single chip support
b6a4c674684585902ee07a4ba	Add single chip support
b834487203996e71c5ea25f83(	Add single chip support
dcc34574efba524cf75608c3b6(	Bring Oxnas Target
17511a7ea86a34b5a5cfd4831(	Kill Oxnas Target
bf136c637c02c154c14bbf48ac(	Remove Linux workarounds
bbc2e1d919c635e2c9fd0cd34a	Update chips' name

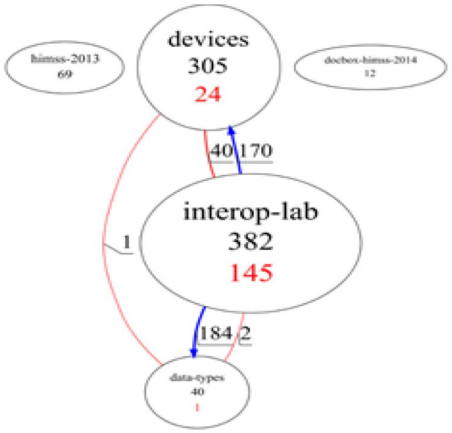
Add cross chip support

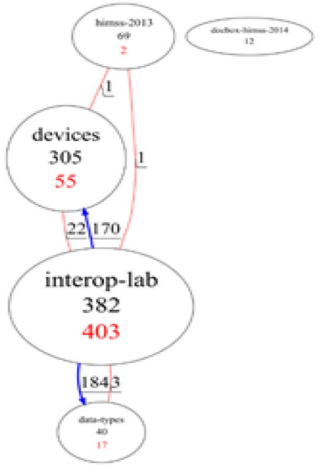
Add single support

Remove Hardware Support

Coarse Development View (MD PnP)

- 20% hardware related changes happen cross modules (left)
- 5% software related changed happen cross modules (right)





- Module “data-types” mostly change due to hardware triggers!

Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VfC Ratio
interop-lab	382	1.76	1.78	0.99
devices	305	0.3	0.37	0.82
himss-2013	69	0.05	0	N/A
data-types	40	0.71	0.12	6.06

Fine-grained Development View (MD PnP)

- The modules “demo-apps”, “philips”, “x73-idl-rti-dds” and “common” are more likely to be driven by software changes;
- The module “demo-devices”, “demo-guis-swing”, and “fluke” are more likely to be triggered by hardware changes.

Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VfC Ratio
demo-apps	224	1.93	1.3	1.48
philips	203	0.17	0.07	2.35
demo-devices	92	1.72	2.91	0.59
infusion-pump-ui	60	0.06	0	N/A
x73-idl-rti-dds	39	0.78	0.12	6.32
common	23	0.77	0.21	3.72
demo-guis-swing	21	1.02	1.37	0.74
fluke	20	0.09	0.24	0.37

Organizational View (MD PnP)

- Vendor 1 is more likely to make hardware related changes.
- Vendor 5 exclusively makes hardware changes.
- Vendor 2 is 4.45 times more likely to make software related changes.
- Vendors 3 and 4 exclusively make software changes.

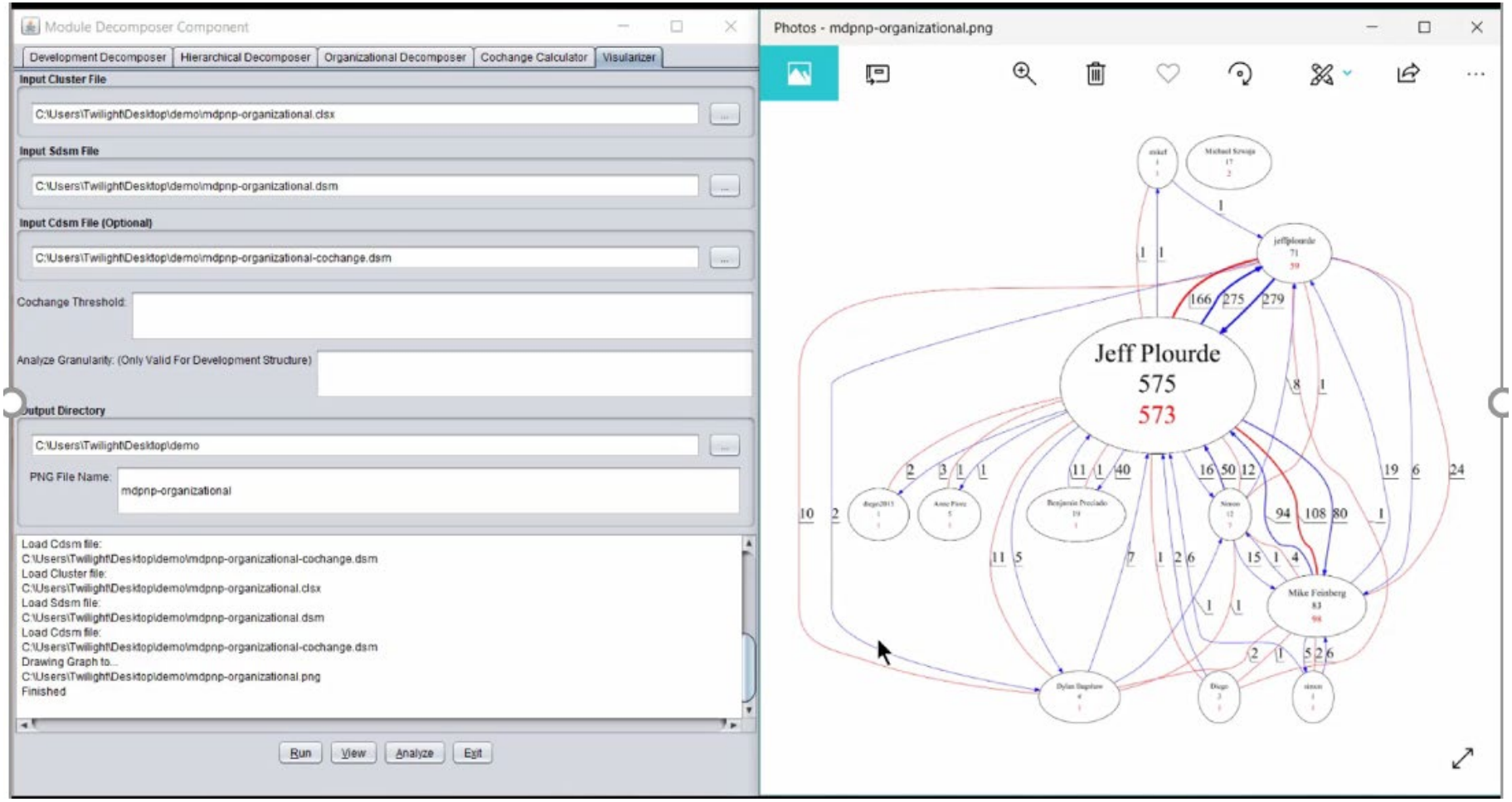
Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VFC Ratio
Vendor 1	648	1	1.12	0.89
Vendor 2	86	1.48	0.33	4.45
Vendor 3	19	0.1	0	N/A
Vendor 4	5	0.36	0	N/A
Vendor 5	4	0	1.02	0.00

Hierarchical View (MD PnP)

- L4, L5, L6, and L7 are more likely to be driven by hardware related changes.

Inner Module	Module Size	SW Change VFC	HW Change VFC	SW/HW VfC Ratio
I0	27	0.38	0	N/A
L0	63	0.9	0.24	3.83
L1	68	1.33	1.2	1.11
L10	15	2.75	0.99	2.79
L11	13	2.78	0.57	4.88
L12	4	1.94	1.85	1.04
L13	313	0.67	0.66	1.01
L2	107	0.89	0.28	3.22
L3	110	0.45	0.13	3.31
L4	14	2.03	9.53	0.21
L5	12	1.08	3.09	0.35
L6	3	13.77	29.65	0.46
L7	40	1.03	3.15	0.33
L8	2	9.04	3.71	2.44
L9	17	4.1	2.18	1.88

Proof-of-Concept Demonstrator



Agenda

- Summary of Project Status
- Background Introduction
- Dataset: OpenWrt and MD PnP
- Module Decomposition
- Domain Concepts Extraction
- Modularity Violation Analyzer
- **Future Plan**

Future Plan

- Overarching Goal: Develop leading indicators and CPS taxonomies to enable managers and engineers to predict, prevent, and minimize problems caused by latent modularity violations.
- Research Problems:

1. How can we build a domain/hardware concept model to represent the domain modular view of a complicated CPS?
2. How can discrepancies among different modular views predict modularity violations in CPS?
3. Can we use the information of latent modularity violations to suggest system reconstructing?

Thank You!

PI: Lu Xiao;
lxiao6@stevens.edu
Co-PI: Michael Pennock
mpennock@stevens.edu

Students:
Xiao Wang
xwang97@stevens.edu
Joana Cardoso
jcardoso@stevens.edu

School of Systems and Enterprises
Stevens Institute of Technology

Backup

Possible Approaches

1. Increase the depth of analysis on the current cases, OpenWrt and MD PnP
   - —Build the domain/hardware model for each project instead of a binary separation, i.e. hardware vs. software
   - —Perform time series analysis to identify leading indicators as early warning sign of modularity violations
   - —Analyze potential restructuring opportunities
2. Increase the scope of analysis on other projects
   - —Other open source projects
   - —Corporate or government cyber-physical project

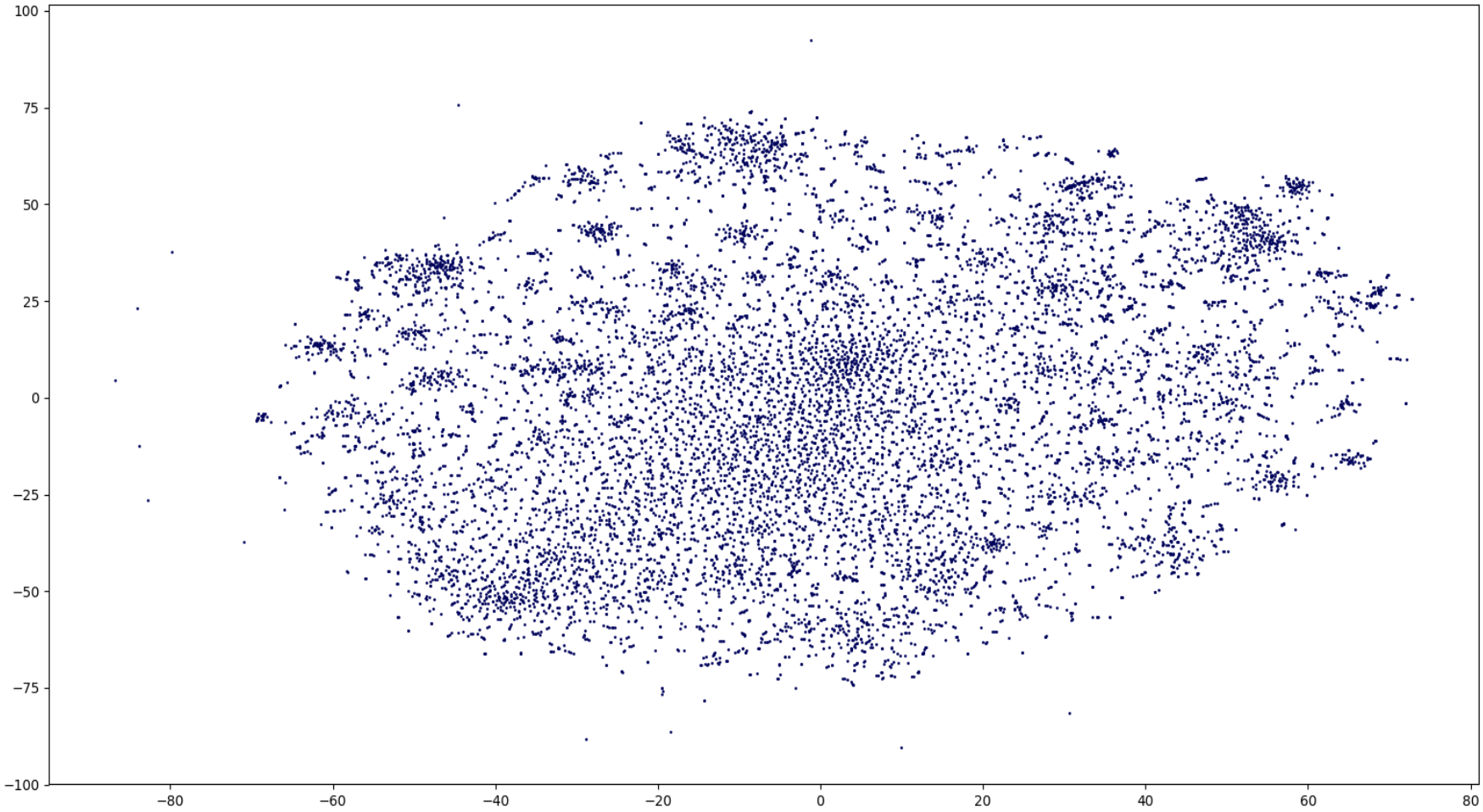
OpenWRT Data

- 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					