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
Md PnP

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

OpenWrt

- Every module is connected with almost every other modules

- First impression: MD PnP has simpler modular structure
Root Package: Co-change Relationship

MD PnP

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

OpenWrt

package
304
243

target
620
1309

devices
305
100

interop-lab
382
775

data-types
40
20

himss-2013
69
5

docbrr-himss-2014
12
1

tools
110
380

scripts
22
22

switches
2
4
**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

- Modules in MD PnP are more likely to co-change with each other compared to 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%
Some stakeholders need more fine-grained modular structure for more detailed guidance.

MD PnP: Fine-grained Package Structure Visualized:
• 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
• Stakeholders should give top priority to these 9 modules
• 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*
**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

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

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

**Open Wrt**
- 21 main contributors
- Everyone’s work is related to everyone else’s.
Modular Structure based on Organizational Structure

MD PnP

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
• 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
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:

```python
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

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

**Top 60 words from query**
• Clustering has resulted in reasonably coherent groupings

<table>
<thead>
<tr>
<th>processors and chipsets</th>
<th>communications</th>
<th>project infrastructure</th>
<th>software management</th>
<th>code organization</th>
<th>storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bcm2708</td>
<td>patch</td>
<td>project</td>
<td>package</td>
<td>config</td>
</tr>
<tr>
<td>1</td>
<td>orion</td>
<td>generic</td>
<td>guide</td>
<td>utils</td>
<td>base</td>
</tr>
<tr>
<td>2</td>
<td>ppc40x</td>
<td>kernel</td>
<td>http</td>
<td>crypto</td>
<td>etc</td>
</tr>
<tr>
<td>3</td>
<td>mx</td>
<td>kmod</td>
<td>documentation</td>
<td>ltq</td>
<td>sh</td>
</tr>
<tr>
<td>4</td>
<td>timer</td>
<td>pending</td>
<td>lede</td>
<td>atm</td>
<td>lib</td>
</tr>
<tr>
<td>5</td>
<td>pxa</td>
<td>backport</td>
<td>welcome</td>
<td>dsl</td>
<td>init</td>
</tr>
<tr>
<td></td>
<td>smp</td>
<td>f</td>
<td>wiki</td>
<td>yaffs2</td>
<td>ipkg</td>
</tr>
<tr>
<td></td>
<td>asoc</td>
<td>hack</td>
<td>org</td>
<td>iwinfo</td>
<td>uci</td>
</tr>
<tr>
<td></td>
<td>cf</td>
<td>v4</td>
<td>forum</td>
<td>swconfig</td>
<td>diag</td>
</tr>
<tr>
<td></td>
<td>pi</td>
<td>fo</td>
<td>downloads</td>
<td>fw</td>
<td>preinit</td>
</tr>
<tr>
<td></td>
<td>jz4740</td>
<td>filter</td>
<td>git</td>
<td>ar6000</td>
<td>conf</td>
</tr>
<tr>
<td></td>
<td>fsl</td>
<td>optional</td>
<td>com</td>
<td>libpcap</td>
<td>hotplug</td>
</tr>
<tr>
<td></td>
<td>compatible</td>
<td>ledtrig</td>
<td>binding</td>
<td>libnl</td>
<td>share</td>
</tr>
<tr>
<td></td>
<td>cpufreq</td>
<td>reduce</td>
<td>release</td>
<td>tiny</td>
<td>bin</td>
</tr>
<tr>
<td></td>
<td>ipq8064</td>
<td>increase</td>
<td>doc</td>
<td>e100boot</td>
<td>sbin</td>
</tr>
<tr>
<td></td>
<td>ipq4019</td>
<td>sched</td>
<td>www</td>
<td>app</td>
<td>usr</td>
</tr>
<tr>
<td></td>
<td>arm64</td>
<td>netdev</td>
<td>cortex</td>
<td>vdsl</td>
<td>skeleton</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>linux</td>
<td>image</td>
<td>file</td>
<td>package</td>
<td>add</td>
<td>patch</td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>support</td>
<td>default</td>
<td>openwrt</td>
<td>fix</td>
<td>es-3</td>
<td></td>
</tr>
<tr>
<td>generic</td>
<td>device</td>
<td>use</td>
<td>makefile</td>
<td>update</td>
<td>es-4</td>
<td></td>
</tr>
<tr>
<td>es-2</td>
<td>board</td>
<td>user</td>
<td>config</td>
<td>build</td>
<td>-default</td>
<td></td>
</tr>
<tr>
<td>ar71xx</td>
<td>usb</td>
<td>data</td>
<td>network</td>
<td>remove</td>
<td>pending-4</td>
<td></td>
</tr>
<tr>
<td>rambps</td>
<td>driver</td>
<td>option</td>
<td>base-files</td>
<td>version</td>
<td>international</td>
<td></td>
</tr>
<tr>
<td>bcm63xx</td>
<td>profiles</td>
<td>configuration</td>
<td>etc</td>
<td>make</td>
<td>submitting</td>
<td></td>
</tr>
<tr>
<td>lantiq</td>
<td>platform</td>
<td>interface</td>
<td>src</td>
<td>change</td>
<td>alike</td>
<td></td>
</tr>
<tr>
<td>bcm2708</td>
<td>switch</td>
<td>http</td>
<td>lib</td>
<td>documentation</td>
<td>pagesource</td>
<td></td>
</tr>
<tr>
<td>adm5120</td>
<td>ethernet</td>
<td>server</td>
<td>control</td>
<td>upgrade</td>
<td>attribution-share</td>
<td></td>
</tr>
<tr>
<td>bcm47xx</td>
<td>wifi</td>
<td>set</td>
<td>ipkg</td>
<td>content</td>
<td>lzma-loader</td>
<td></td>
</tr>
<tr>
<td>mtd</td>
<td>phy</td>
<td>port</td>
<td>net</td>
<td>lede</td>
<td>swconfig</td>
<td></td>
</tr>
<tr>
<td>ath79</td>
<td>code</td>
<td>rule</td>
<td>modules</td>
<td>page</td>
<td>backport-4</td>
<td></td>
</tr>
<tr>
<td>ixp4xx</td>
<td>register</td>
<td>using</td>
<td>services</td>
<td>missing</td>
<td>coldfire</td>
<td></td>
</tr>
<tr>
<td>ipq806x</td>
<td>gpio</td>
<td>start</td>
<td>init</td>
<td>new</td>
<td>map</td>
<td></td>
</tr>
<tr>
<td>s3c24xx</td>
<td>wireless</td>
<td>address</td>
<td>scripts</td>
<td>enable</td>
<td>uml</td>
<td></td>
</tr>
</tbody>
</table>
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.
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