RT-180: Identifying and Measuring Modularity Violations in Hybrid Cyber-Physical Systems

Lu Xiao, Mike Pennock, Joana Cardoso, Xiao Wang
School of Systems and Enterprises, Stevens Institute of Technology

Research Overview

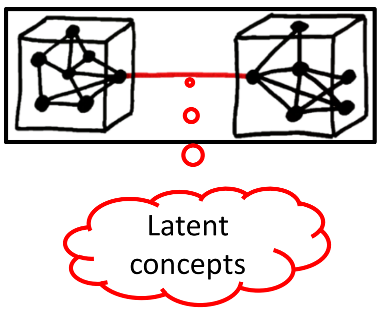
- Cyber-physical systems are composed of diverse subsystems consisting of both *physical* and *software* components developed by different vendors.
- Stakeholders, such as DoD, have increasingly emphasized modular and open approaches to system development to improve interoperability, facilitate system evolution and technology insertion, and foster competition.
- With the advance of technology, the recognition of new consumer needs, and the detection of deficiencies in the current systems, can the upgrade, replacement, or problem fix happen quickly in a plug-and-play manner? In other words, is a cyber-physical system truly modular?

CPS = Hardware + Software modules.

Modularization = Interdependence within + Independence across modules [1]

Goals & Objectives

- In the traditional software engineering field, it has been observed that modules without any explicit structural dependencies were found to share latent connections (e.g. evolutionary co-changes). We call this phenomenon a *Modularity Violation (MV) [2]*



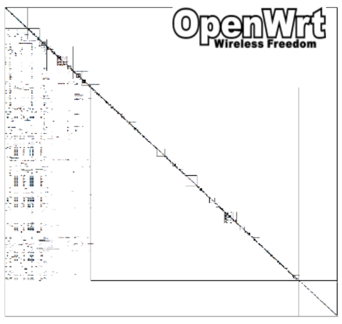
- The objective of this research is to develop techniques, metrics, and models that would allow the stakeholders to detect, measure, and understand *Modularity Violations* in developed and acquired cyber-physical systems.
- **Incubation Phase: Is it possible to identify modularity violations in cyber-physical systems?**
  - 1) **Software** vs. **Software**
  - 2) **Software** vs. **Hardware**
  - 3) **Hardware** vs. **Hardware**

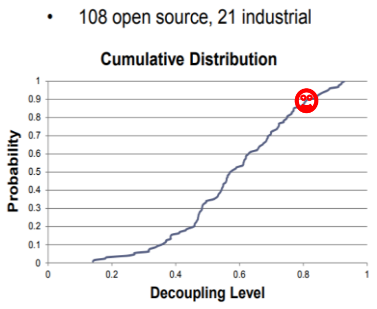
Data & Analysis

- Feasibility study subjects in Incubation Phase

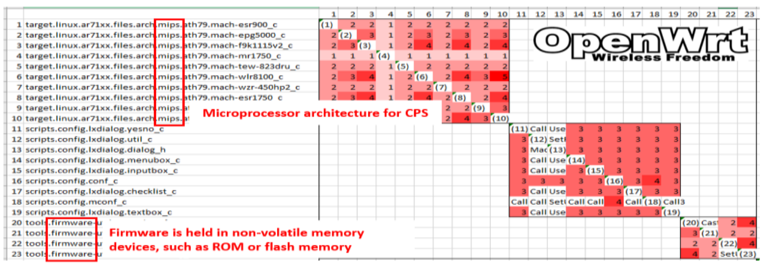
Project	#Files	#Methods	LOC	#Developers	#Revisions	History
OpenWrt	1052	6061	163114	137	38099	Since 2004
MD PnP	866	7872	73616	12	1605	Since 2013

- Observation 1: the OpenWrt is more modularized than about 90% of the 129 (commercial and open source) traditional software systems we studied before.





- Observation 2: Software-Software modularity violations in OpenWrt include hardware related information in the naming conventions of the involved source files.


1 target.linux.ar71xx.files.arch.mips.ath79.mach-esr900_c
2 target.linux.ar71xx.files.arch.mips.ath79.mach-epg5000_c
3 target.linux.ar71xx.files.arch.mips.ath79.mach-f9k1115v2_c
4 target.linux.ar71xx.files.arch.mips.ath79.mach-mr1750_c
5 target.linux.ar71xx.files.arch.mips.ath79.mach-tew-823dru_c
6 target.linux.ar71xx.files.arch.mips.ath79.mach-wlr8100_c
7 target.linux.ar71xx.files.arch.mips.ath79.mach-wzr-450hp2_c
8 target.linux.ar71xx.files.arch.mips.ath79.mach-esr1750_c
9 target.linux.ar71xx.files.arch.mips.a
10 target.linux.ar71xx.files.arch.mips.a
11 scripts.config.lxdialog.yesno_c
12 scripts.config.lxdialog.util_c
13 scripts.config.lxdialog.dialog_h
14 scripts.config.lxdialog.menubox_c
15 scripts.config.lxdialog.inputbox_c
16 scripts.config.conf_c
17 scripts.config.lxdialog.checklist_c
18 scripts.config.mconf_c
19 scripts.config.lxdialog.textbox_c
20 tools.firmware-u
21 tools.firmware-u
22 tools.firmware-u
23 tools.firmware-u

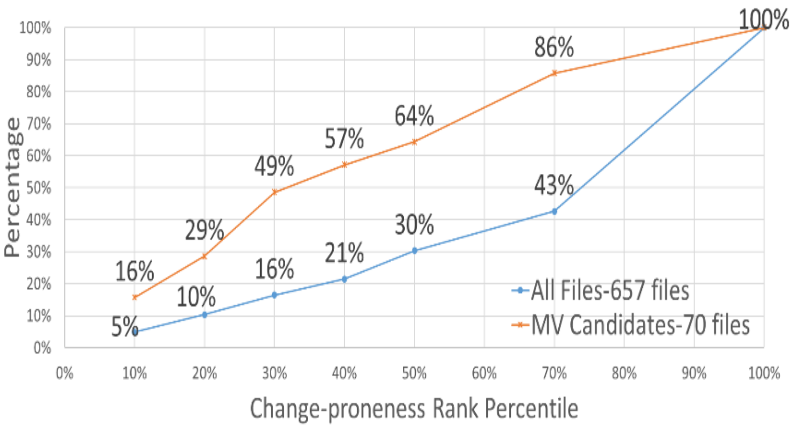
OpenWrt
Wireless Freedom
Microprocessor architecture for CPS
Firmware is held in non-volatile memory devices, such as ROM or flash memory

We found 23 source files that exhibit evolutionary coupling above 4, but are structurally independent from each other.

- Observation 3: Hardware-related concepts are the main contributing factors to Software-Hardware MV in OpenWrt.

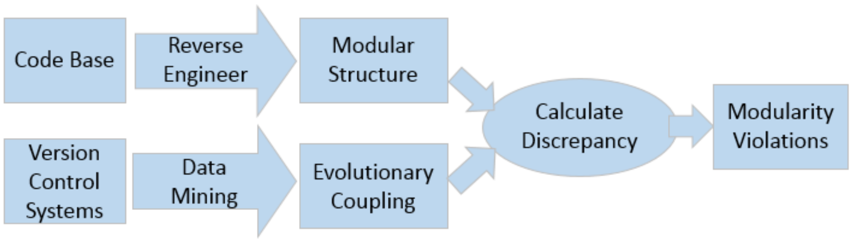
> *{"radio", "WiFi", "zigbee", "btle", "mips", "ramips", "mtd", "broadcom", "routerboot", "router", "firmware", "bluetooth", "energy", "power", "soc", "chip"}*

71 source files are involved in hardware-related modularity violations using key-word based heuristics. The figure below shows that these 71 files are twice as likely to change compared to average files. Therefore, the hardware related concepts could be the main contributing factors of changes made to software modules.

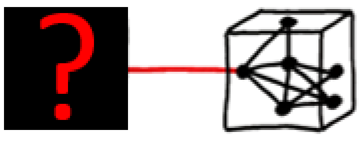


Methodology

- Identify **Software** vs. **Software** Modularity Violations [3]



- How to identify modularity violations involving hardware components in cyber-physical systems? How to track changes made to / propagate from hardware components?



The records for the maintenance activities (e.g. revisions) on the hardware side are usually not available or do not exist at all.

- Identify software maintenance activities that imply hardware-related concepts. This helps point to Modularity Violations involving hardware components.

When developers make changes to a software module/entity, he/she says something about hardware components. E.g. "set *chip* type directly in ar8216_id_chip."	The naming of a software entity contains important hardware related key words, e.g. "mips", "firmware". openwrt.tools.*firmware*-utils.src.mktplinkfw_c

Future Research

- Examine the Criteria to Decompose CPS into Modules: in the incubator phase, we treated source files as the granularity of 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.
- Build "Concept Learner " to Identify Modularity Violations in Different Domains: in the incubator phase, we realized that the keyword heuristics developed for one project domain cannot be applied to another project domain.
- Build Decision Framework and demonstrator. The ultimate goal is to provide decision making support to improve modularity where appropriate and achieve the associated benefits.

Contacts/References

[1] Carliss Y. Baldwin and Kim B. Clark. 1999. Design Rules: The Power of Modularity Volume 1. MIT Press, Cambridge, MA, USA.

[2] S. Wong, Y. Cai, M. Kim and M. Dalton, "Detecting software modularity violations," 2011 33rd International Conference on Software Engineering (ICSE), Honolulu, HI, 2011, pp. 411-420. doi: 10.1145/1985793.1985850

[3] Lu Xiao, Yuanfang Cai, and Rick Kazman, "Titan: a Toolset that Connects Software Architecture with Quality Analysis", FSE 2014 Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. Pages 763-766. Hong Kong, China, Nov 16-21, 2014.

- Lu Xiao, Assistant Professor, lxiao6@stevens.edu
- Mike Pennock, Assistant Professor, mpennock@stevens.edu
- Joana Cardoso, Ph.D. Student, jcardoso@stevens.edu
- Xiao Wang, Ph.D. Student, xwang97@stevens.edu