Overview

• Background
• Research findings on OSSD
• Case study: OSSD, open architectures, and software licenses for Command and Control
• Case study: Securing the development and evolution of an OA C2 system within an agile, adaptive software ecosystem
• Discussion, challenges, new practices, and conclusions
Background
Personal history of OSS Development studies

- 2000-2016 (60+ publications)
  - Computer games, defense, X-ray astronomy, Internet/Web infrastructure, bioinformatics, higher education, e-commerce, virtual reality, IP licenses.
- Discovering requirements practices and processes across OSS communities of practice.
- Participant role sets, role migration, and social movements within/across (global) OSS projects.
- Open architecture (OA) systems with heterogeneously licensed components.
What is open source?

- *Open source* denotes specifications, representations, socio-technical processes, and multi-party coordination mechanisms in *human readable, computer processable* formats.
  - Open Source Software (OSS)
- *Socio-technical control* of OSS is elastic, negotiated, and amenable to decentralization.
- OSS development *subsidized* by contributors and participants.
When did software become OSS?

- **IBM Share** (started late 1950s – open sharing of application software)
- **DECUS** (1960s – open exchange of user developed software via magnetic tape)
- **Academic/research software sharing** (1970-80s) –
  - Internet, language environments (*InterLisp*, *UCI Lisp*), operating systems (*Tenex*, *Unix*), *X Windows*, VLSI CAD tools, Ingres Relational DBMS, and many more.
  - Most early Interactive Development Environments
- **USC System Factory** (1981-90) – Iterative open source software life cycle engineering
“Open Source Software” declared

- *Free Software* movement (1983) and GPL software
- “Open Source Software” declared in 1998 and later!
  - GCC, X Windows, BSD Unix, Mosaic, WWW, BRL CAD, etc.
- Linux OS → Linux Kernel development goes *global*
  - Linux Kernel today involves >4K developers
- OSS and OSSD become “engines of innovation” and basis for start-up ventures (VA Linux, RedHat, MySQL) and corporate OSS (HP, IBM, Sun)
“Open Source Software” declared

• But soon after, many software engineers and software development firms decry OSS as:
  – Cancer (will kill software products industry)
  – Communism (no for-profit corporations will use)
  – Untrustworthy and insecure (not for Defense)
  – Never going to be used on mission-critical software systems
What is free/libre/open source software development?

- Free/Libre (as in “freedom” or liberty, not cost) vs. OSS
  - Freedom to access, browse/view, study, modify and redistribute the source code: the basis of the GPL
    - Free is about *IP rights and obligations* (licenses)!
  - Free is always open, but open source is not always free

- OSSD is not (in general) “Software Engineering”
  - *Different*: OSSD can be faster, better, and cheaper than SE in some circumstances
    - OSSD teams use 10-500+ OSSD tools (versions) and communications applications to support their development work
# International Sites

LibreOffice has a number of teams working on localization into different languages. For software, support and documentation in your preferred language, please check below to see if we currently have a site serving your locale. If you don't see a site in the language you are looking for, please consider joining our community and getting involved in working to fill the gap.

<table>
<thead>
<tr>
<th>Link</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://ar.libreoffice.org/">http://ar.libreoffice.org/</a></td>
<td>Arabic</td>
</tr>
<tr>
<td><a href="http://bo.libreoffice.org/">http://bo.libreoffice.org/</a></td>
<td>Tibetan</td>
</tr>
<tr>
<td><a href="http://cs.libreoffice.org/">http://cs.libreoffice.org/</a></td>
<td>Czech</td>
</tr>
<tr>
<td><a href="http://da.libreoffice.org/">http://da.libreoffice.org/</a></td>
<td>Danish</td>
</tr>
<tr>
<td><a href="http://de.libreoffice.org/">http://de.libreoffice.org/</a></td>
<td>German</td>
</tr>
<tr>
<td><a href="http://el.libreoffice.org/">http://el.libreoffice.org/</a></td>
<td>Greek</td>
</tr>
<tr>
<td><a href="http://eo.libreoffice.org/">http://eo.libreoffice.org/</a></td>
<td>Esperanto</td>
</tr>
<tr>
<td><a href="http://es.libreoffice.org/">http://es.libreoffice.org/</a></td>
<td>Spanish</td>
</tr>
<tr>
<td><a href="http://et.libreoffice.org/">http://et.libreoffice.org/</a></td>
<td>Estonian</td>
</tr>
<tr>
<td><a href="http://fa.libreoffice.org/">http://fa.libreoffice.org/</a></td>
<td>Persian</td>
</tr>
<tr>
<td><a href="http://fi.libreoffice.org/">http://fi.libreoffice.org/</a></td>
<td>Finnish</td>
</tr>
<tr>
<td><a href="http://fr.libreoffice.org/">http://fr.libreoffice.org/</a></td>
<td>French</td>
</tr>
<tr>
<td><a href="http://gl.libreoffice.org/">http://gl.libreoffice.org/</a></td>
<td>Galician</td>
</tr>
<tr>
<td><a href="http://he.libreoffice.org/">http://he.libreoffice.org/</a></td>
<td>Hebrew</td>
</tr>
<tr>
<td><a href="http://hi.libreoffice.org/">http://hi.libreoffice.org/</a></td>
<td>Hindi</td>
</tr>
<tr>
<td><a href="http://hu.libreoffice.org/">http://hu.libreoffice.org/</a></td>
<td>Hungarian</td>
</tr>
<tr>
<td><a href="http://it.libreoffice.org/">http://it.libreoffice.org/</a></td>
<td>Italian</td>
</tr>
<tr>
<td><a href="http://ja.libreoffice.org/">http://ja.libreoffice.org/</a></td>
<td>Japanese</td>
</tr>
<tr>
<td><a href="http://lo.libreoffice.org/">http://lo.libreoffice.org/</a></td>
<td>Lao</td>
</tr>
<tr>
<td><a href="http://lt.libreoffice.org/">http://lt.libreoffice.org/</a></td>
<td>Lithuanian</td>
</tr>
</tbody>
</table>
Most downloads over all time

<table>
<thead>
<tr>
<th>Rank</th>
<th>Project Name</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Microsoft's TrueType core fonts</td>
<td>1.4B</td>
</tr>
<tr>
<td>2</td>
<td>Notepad++ Plugin Manager</td>
<td>993.6M</td>
</tr>
<tr>
<td>3</td>
<td>VLC media player</td>
<td>897.1M</td>
</tr>
<tr>
<td>4</td>
<td>eMule</td>
<td>682.1M</td>
</tr>
<tr>
<td>5</td>
<td>Vuze - Azureus</td>
<td>542.1M</td>
</tr>
<tr>
<td>6</td>
<td>MinGW - Minimalist GNU for Windows</td>
<td>435.2M</td>
</tr>
<tr>
<td>7</td>
<td>FileZilla</td>
<td>413.6M</td>
</tr>
<tr>
<td>8</td>
<td>7-Zip</td>
<td>409.1M</td>
</tr>
<tr>
<td>9</td>
<td>Ares Galaxy</td>
<td>406.5M</td>
</tr>
<tr>
<td>10</td>
<td>PortableApps.com: Portable Software/USB</td>
<td>361.5M</td>
</tr>
</tbody>
</table>

Most downloads last week

<table>
<thead>
<tr>
<th>Rank</th>
<th>Project Name</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Microsoft's TrueType core fonts</td>
<td>13.1M</td>
</tr>
<tr>
<td>2</td>
<td>egbucket</td>
<td>8.0M</td>
</tr>
<tr>
<td>3</td>
<td>Notepad++ Plugin Manager</td>
<td>7.4M</td>
</tr>
<tr>
<td>4</td>
<td>MinGW - Minimalist GNU for Windows</td>
<td>2.0M</td>
</tr>
<tr>
<td>5</td>
<td>Apache OpenOffice</td>
<td>1.0M</td>
</tr>
<tr>
<td>6</td>
<td>FileZilla</td>
<td>966.1K</td>
</tr>
<tr>
<td>7</td>
<td>PortableApps.com: Portable Software/USB</td>
<td>961.5K</td>
</tr>
<tr>
<td>8</td>
<td>Ubuntuzilla: Mozilla Software Installer</td>
<td>624.1K</td>
</tr>
<tr>
<td>9</td>
<td>bucket</td>
<td>537.5K</td>
</tr>
<tr>
<td>10</td>
<td>Scrollout F1</td>
<td>405.4K</td>
</tr>
</tbody>
</table>
GitHub OSS Repositories (2015 sample)

Users Filter
Languages Filter

Click to Highlight

GitHubViz visualization (c) by Donne Martin, 2016.
How do various corporations populate the Apache projects?

Submitted by megan on July 21, 2010 - 2:01pm

With the FLOSSmole Apache Project/Contributor/Roles data we updated earlier today, we thought an interesting initial analysis would be to figure out how various corporations populate the Apache projects (at least according to the official lists of contributors posted on each Apache project page).

Here is a list of the Apache projects with the highest density of participation by a single corporation:

<table>
<thead>
<tr>
<th>project_name</th>
<th>organization</th>
<th>org dev count</th>
<th>all devs</th>
<th>pct of team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synapse</td>
<td>WSO2</td>
<td>15</td>
<td>24</td>
<td>62.50</td>
</tr>
<tr>
<td>Ignite</td>
<td>GridGain</td>
<td>15</td>
<td>25</td>
<td>60.00</td>
</tr>
<tr>
<td>Jackrabbit</td>
<td>Adobe</td>
<td>27</td>
<td>49</td>
<td>55.10</td>
</tr>
<tr>
<td>Knox</td>
<td>Hortonworks</td>
<td>15</td>
<td>28</td>
<td>53.57</td>
</tr>
<tr>
<td>Geronimo</td>
<td>IBM</td>
<td>33</td>
<td>63</td>
<td>52.38</td>
</tr>
<tr>
<td>Ambarni</td>
<td>Hortonworks</td>
<td>68</td>
<td>131</td>
<td>51.91</td>
</tr>
<tr>
<td>Falcon</td>
<td>InMobi</td>
<td>11</td>
<td>22</td>
<td>50.00</td>
</tr>
<tr>
<td>Calcite</td>
<td>Hortonworks</td>
<td>7</td>
<td>16</td>
<td>43.75</td>
</tr>
<tr>
<td>REEF</td>
<td>Microsoft</td>
<td>14</td>
<td>32</td>
<td>43.75</td>
</tr>
<tr>
<td>Sentry</td>
<td>Cloudera</td>
<td>14</td>
<td>33</td>
<td>42.42</td>
</tr>
<tr>
<td>Flume</td>
<td>Cloudera</td>
<td>11</td>
<td>26</td>
<td>42.31</td>
</tr>
<tr>
<td>Nutch</td>
<td>NASA JPL</td>
<td>6</td>
<td>15</td>
<td>40.00</td>
</tr>
<tr>
<td>OpenJPA</td>
<td>IBM</td>
<td>12</td>
<td>30</td>
<td>40.00</td>
</tr>
<tr>
<td>AsterixDB</td>
<td>UC Irvine</td>
<td>11</td>
<td>28</td>
<td>39.29</td>
</tr>
<tr>
<td>Airavata</td>
<td>Indiana University</td>
<td>12</td>
<td>32</td>
<td>37.90</td>
</tr>
<tr>
<td>Hive</td>
<td>Hortonworks</td>
<td>21</td>
<td>56</td>
<td>37.50</td>
</tr>
<tr>
<td>Phoenix</td>
<td>Salesforce</td>
<td>10</td>
<td>27</td>
<td>37.04</td>
</tr>
<tr>
<td>Spark</td>
<td>Databricks</td>
<td>16</td>
<td>44</td>
<td>36.36</td>
</tr>
<tr>
<td>Avro</td>
<td>Cloudera</td>
<td>5</td>
<td>14</td>
<td>35.71</td>
</tr>
<tr>
<td>BVal</td>
<td>IBM</td>
<td>5</td>
<td>14</td>
<td>35.71</td>
</tr>
<tr>
<td>Aries</td>
<td>IBM</td>
<td>19</td>
<td>54</td>
<td>35.19</td>
</tr>
<tr>
<td>Mesos</td>
<td>Mesosphere</td>
<td>9</td>
<td>26</td>
<td>34.02</td>
</tr>
<tr>
<td>Karaf</td>
<td>JBoss</td>
<td>7</td>
<td>23</td>
<td>30.43</td>
</tr>
<tr>
<td>OpenMeetings</td>
<td>Unipro</td>
<td>6</td>
<td>20</td>
<td>30.00</td>
</tr>
<tr>
<td>Syncoppe</td>
<td>Terasa</td>
<td>6</td>
<td>20</td>
<td>30.00</td>
</tr>
</tbody>
</table>
Research findings on OSSD
OSSD Research Surveys


OSS Development Models
(and IP licensing regimes)

- Free Software (GPLv2, GPLv3, AGPL)
- Permissive Open Source (BSD/MIT, FreeBSD, APL, MPL)
- Non-profit foundations (Apache, Mozilla, Gnome, Perl, Eclipse)
- Government Open Source Software (Ozone/OWF)
- Consortium/Alliance (OSDL, SugarCRM)
- Corporate-Sponsored (Google, HP, IBM, Microsoft, Oracle)
- Corporate/Inner Source (Hewlett-Packard)
- Open Modding Extensions and domain-specific scripting (DSLs) to Closed Source (many games and many game studios)
- Gated Community Source (Sakai, Westwood)

----------------------------- not OSSD models below -----------------------------

- Shared Source via Non-Disclosure Agreement
- Open Systems (open APIs, closed components)
OSSD Project Characteristics

- Operational code early and often--actively improved and continuously adapted
  - Short-cycle (OSS) vs. long-cycle (SLC) time processes

- *Post-facto* software system requirements and design
  - OSSD has its own “-ilities” which differ from those for SE

- *Caution*: the vast majority (>90%) of OSSD projects fail to grow or produce a viable, sustained software release.
OSSD Project Characteristics

- OSS developers are typically end-users of what they build
  - Few OSS users (~1%) are also OSS developers
  - These developers know their “functional requirements”

- Requires critical mass of contributors and OSS components connected through socio-technical interaction networks
  - Can be <1% of user community for large OSS projects
  - Sustained commitment and contribution critical

- OSSD projects can emerge/evolve via bricolage
  - Unanticipated architectural (de)compositions
  - Multi-project component integrations
Individual participation in OSSD projects: motives and consequences

- OSS developers want to:
  - learn about new tools, techniques, skills, etc.
  - have fun building software
  - exercise their technical skill
  - try out new kinds of systems to develop
  - interconnect multiple OSSD projects

- OSS developers frequently:
  - build trust and reputation with one another
  - achieve “geek fame” (for project leaders)
  - spend more time reading online documents and communicating with one another than writing code
OSSD resources/capabilities

- Personal software development resources
- Beliefs supporting F/OSSD
- OSSD informalisms
- Skilled, self-organizing developers
- Discretionary time and effort
- Trust and social accountability
Personal software development resources

- Sustained commitment of personal or enterprise resources helps *subsidize* OSSD projects via:
  - Personal computer(s)
  - Internet access
  - Hosting personal Web site
  - Hosting project repositories (now on GitHub)
  - Personal choice of software development tools or tool set
Beliefs supporting F/OSSD

• *Freedom of expression*
  • What to develop or work on
  • How to develop it
  • What tools to employ

• *Freedom of choice*
  • When to release work products
  • Expressing what can be said to whom with or without reservation

• Observation: **Beliefs shape software architecture**
OSSD Informalisms

- Software *informalisms*—artifacts participants use to describe, proscribe, or prescribe what’s happening in a project.
- Informalisms capture detailed rationale and debates for what changes were made in particular development activities, artifacts, or source code files.
- Outside the world of software development, informalisms are sometimes called, “para-texts.”
Benefits of Qt3?

What are the benefits of moving to Qt3?

- Support for Arabic and Hebrew
- RichText classes
- Database support
- Component model
- No more cut/paste problems (but only between Qt3 apps)

One of the most complained about aspects of X is the darn clipboard, so getting KDE based on Qt3 will solve a lot of headaches. But this is from a user perspective.

From a developer perspective, KDE-DB is going to utilize Qt3’s database support, and this can’t happen until they make the switch. KWord currently uses a backported richtext for use with Qt2. So you can see that there is a drive/need in KDE to use the new Qt3 features.

Re: Benefits of Qt3?

What is the purpose of database support in a *widget toolkit*? Isn’t this just like placing TCP/IP support in /etc/passwd or another similarly unrelated place?

Re: Benefits of Qt3?

there is often a need to access data from a database and display it in a GUI, or vice versa. in those cases having a db API that abstracts the details of the actual data access away (connecting, sending queries, retrieving results, details specific to a given db implementaiton, etc) that works nicely with your widgets (even so far as to make the widgets aware of the database) is very very nice.

making such things simple and convenient opens the door to making more applications database aware (e.g. financial packages, email apps, contact information systems)
**OSSD Informalisms: Artifacts and repositories enable dispersed/distant collaboration in OSS development**

<table>
<thead>
<tr>
<th>Email lists</th>
<th>Discussion forums</th>
<th>News postings</th>
<th>Project digests</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM/Internet Relay Chat</td>
<td>Use cases or scenarios</td>
<td>How-to guides</td>
<td>Screenshots</td>
</tr>
<tr>
<td>FAQs; to-do lists: item lists</td>
<td>Project Wikis</td>
<td>System documentation</td>
<td>External publications</td>
</tr>
<tr>
<td>Copyright licenses</td>
<td>Architecture diagrams</td>
<td>Intra/inter-app scripting</td>
<td>Plug-ins</td>
</tr>
<tr>
<td>Code from other projects</td>
<td>Project Web site</td>
<td>Multi-project portals</td>
<td>Project source code web</td>
</tr>
<tr>
<td>Project repositories</td>
<td>Software bug reports</td>
<td>Issue tracking databases</td>
<td>Blogs, videos, social media.</td>
</tr>
</tbody>
</table>
Skilled, self-organizing developers

- Successfully developing an open architecture system requires prior experience
- Organizing project work as a *virtual organization*
  - Skill-based meritocracy
  - Informal rules of governance and control, but rules are readily recognized by participants
  - Social control incorporated into software and informalisms
    - How, where, and when to access data via APIs, UIs, and other architectural features
Discretionary time and effort

- Self-determination
  - work on what’s interesting
- Peer recognition
  - becoming a social gateway
- Project affiliation or identification
- Self-promotion
  - How to realize career advancement
- Belief in inherent value of OSS
Trust and social accountability

- **Social capital** accrues via:
  - Assuming ownership of an OSS module, fork, or project
  - Voting on approval of other’s actions
  - Shared peer reviewing
  - Contributing “gifts” that are (re)usable

- Accrued social capital is used to mitigate conflicts and accommodate resolutions
- Sustained social capital enables social networking externalities
- Shared investment of social capital as basis for trust
Cooperation, coordination, and control in OSSD projects
FIGURE 1. Bubble-plot overview of what we know about global software engineering (GSE). Results are based on a systematic review of the GSE literature available from 2000 to 2007. The left side classifies the 59 relevant studies thematically in terms of success or failure, and the right side classifies them according to globalization type.

Software version control

- Enables stabilization and synchronization of dispersed, invisible OSSD work
- SVC tools (CVS, SVN, GitHub, etc.) are:
  - Central mechanism coordinating development
  - Online venue for mediating control over what changes will be accommodated
  - Gentle but sufficient socio-technical control mechanism that constrains overall project complexity
Implicit project management

- OSSD projects self-organize as a meritocratic role-hierarchy and virtual project management
  - Meritocracies embrace incremental innovations over radical innovations
  - VPM requires people to act in leadership roles based on skill, availability, and belief in project community
- Reliance on evolving web of software informalism content constrains collective action within OSSD projects.
A *meritocratic* role sets, role hierarchy, and role migration paths for OSSD

![Diagram of role hierarchy](image)

**Figure 2.** An “onion” pyramid representation of a generic OSSD project organizational hierarchy with multiple role-sets and advancement tracks.

Alliances, social networking, and community development
Community networking

- Becoming a central node in a network of OSS developers increases social capital
  - *Linchpin developers* as social gateways
  - Sharing beliefs, tools, artifacts enables shared experience, camaraderie, collective learning

- Multi-project clustering enables small projects to merge into sustainable projects

- *Intellectual property* (licensing) regime fosters alignment and alliance with other IP compatible projects and organizations
OSS Social Networking across projects

Developers are nodes / Projects are links
24 Developers
5 Projects
2 Linchpin Developers
1 Cluster

Source: G. Madey, et al., 2005
OSSD as multi-project software ecosystems
What is a (software) ecosystem?

- *Ecology of systems* with diverse species juxtaposed in adaptive prey-predator food chain relationships.

- *Software supply network* of component producers, system integrators, and consumers.

- *Socio-technical network of processes* that transform the flow of resources, enacted by actors in different roles, using tools, to produce products, services, or capabilities.
Multi-project software ecosystem

- Mutually dependent OSS development and evolution propagate architectural styles, dependencies, and vulnerabilities
- Architectural bricolage arises when autonomous OSSD projects, artifacts, tools, and systems co-mingle or merge
  - Enables discontinuous or exponential growth of OSS source code, functionality, complexity, contributions
Software ecosystem of producers and the (licensed) software components for Web-compatible enterprise systems
Legend: Boxes are activities (using informalisms); Ellipses are resources required or provided; Actor roles in boldface; flow dependencies as arrows.
Project Issue Tracking: netbeans.org

### Issue

#### Tracking:

- **Query**: This page lets you search the database for recorded issues.

<table>
<thead>
<tr>
<th>Issue type</th>
<th>Component</th>
<th>Subcomponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFECT</td>
<td><strong>UNCATEGORIZED</strong></td>
<td><em>ALL</em></td>
</tr>
<tr>
<td>ENHANCEMENT</td>
<td>3rd-party</td>
<td>accessibility</td>
</tr>
<tr>
<td>FEATURE</td>
<td>ally</td>
<td>actions</td>
</tr>
<tr>
<td>TASK</td>
<td>accelerators</td>
<td>annotations</td>
</tr>
<tr>
<td>PATCH</td>
<td>ant</td>
<td>ant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Resolution</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNCONFIRMED</td>
<td>FIXED</td>
<td>P1</td>
</tr>
<tr>
<td>NEW</td>
<td>INVALID</td>
<td>P2</td>
</tr>
<tr>
<td>STARTED</td>
<td>WONTFIX</td>
<td>P3</td>
</tr>
<tr>
<td>REOPENED</td>
<td>LATER</td>
<td>P4</td>
</tr>
</tbody>
</table>

Submit query
Summary of OSSD Research

- OSS Development is a long-standing approach to building software applications and infrastructure systems.
- Most successful OSSD projects are now global.
- OSSD is inherently a socio-technical approach to continuous software development.
- Unresolved issues in OSSD:
  - Gender diversity
  - Cross-cultural diversity
  - OSSD in low-access countries
Case study: OSSD, open architectures, and software licenses for command and control (C2) systems
What is a command and control system?

- System whose operation entails planning, monitoring and time-critical control of resource flows
- Systems whose operation entails consequential human-centered decisions for what to do next
- Systems that manage networked infrastructure
- Systems that can be managed through a multi-panel/-guage dashboard and wall-sized displays
- C2 domains: military, police/fire, public utilities, air traffic control, airline operations, telecommunications and broadcasting networks, *Formula 1*, agile manufacturing in Pharma, etc.
Virtual world for C2: **DECENT**
Under-explored topics for DECENT*

- Most VW software technologies, including OpenSim, offer little/no ready support for integration of external application programs or other software components.
  - Though OSS, it does not (yet) provide an open architecture capability
- Securing a VW for C2 applications is a major concern in advancing this line of research and deployment.

Recurring vision for C2 systems [c. 2010]
What is an Open Architecture?

- U.S. Dept of Defense has announced policies and initiatives that commit to the acquisition of software-intensive systems that require or utilize an *Open Architecture*, and Open Technology.
- OA systems may include components with open APIs, OSS technology or development processes.
- Air Force, Army, and Navy each have their own reasons for adoption OA systems.
- But what happens when there are conflicts across the services regarding what an OA is?
- Therefore, is it clear what an OA is?
Open Architecture (OA) Software Concepts

- Software source code components
  - Standalone programs
  - Libraries, frameworks, or middleware
  - Inter-application script code (e.g., for mash-ups)
  - Intra-application script code (e.g., for Rich Internet Apps.)
- Executable software components (binaries)
- Application program interfaces (APIs)
- Software connectors
- Configured sub-system or system
Legend: Grey boxes are components; ellipses are connectors; white boxes are interfaces; arrows are data or control flow paths; complete figure is architectural design configuration.
OA software ecosystems

Software supply network for OA system components: Component IP license and cybersecurity requirements propagate from/to Producers, Integrators, and Consumers.
Open Architectures, OSS, and OSS license analysis

• **Goal**: identify software architecture principles and IP licenses that mediate OA
• OSS elements subject to different IP licenses
• Govt/business policies and initiatives now encouraging OA with OSS elements
• How to determine the requirements needed to realize OA strategies with OSS?

OSS elements subject to different IP licenses

- Intellectual Property licenses stipulate obligations (requirements) and rights (capabilities) regarding use of IP
  - GPL (Gnu Public License) stipulate right to access, study, modify, and reciprocal obligation to redistribute modified source
  - Mozilla now offers a “tri-license” for its software like Firefox:
    - GPL, MPL (lightweight), or Restricted (accommodating proprietary services)
    - Other OSS covered by different IP obligations and rights

OSS elements subject to different IP licenses

• How to determine which obligations and rights apply to a configured system at what time?
  • At *design-time* (maximum flexibility)
  • At *build-time* (may/not be able to redistribute components at hand)
  • At *run-time* (may/not need to install/link-to components from other sources)
  • At *evolution-time* (component suppliers, licenses, connections, etc. may change)

Specifying and analyzing system access control requirements as “licenses”

- *Security policies* imply capabilities that correspond to “rights and obligations” in licenses
- Should be possible to specify and analyze system *security architecture* that conform to a *security meta-model*, much like we do for software licenses
- Should be possible to develop computational tools and development environments that can analyze security at design-time, build-time, and run-time, as well as when the system evolves
Software license meta-model for specifying constraint annotations
Logical modality and objects of software license rights and obligations constraints

<table>
<thead>
<tr>
<th>Actor</th>
<th>Modality</th>
<th>Action</th>
<th>Object</th>
<th>License (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Right</td>
<td>May or Need Not</td>
<td>The set of actions is large, comprising whatever actions the licenses in question utilize</td>
<td>Any Under This License, Any Source Under This License, Any Component Under This License</td>
<td>This License or Object's License</td>
</tr>
<tr>
<td>Concrete Right</td>
<td>Licensee or Licensor</td>
<td></td>
<td>Concrete Object</td>
<td>Concrete License</td>
</tr>
<tr>
<td>Concrete Obligation</td>
<td>Need Not</td>
<td></td>
<td>Right's Object, All Sources Of Right's Object, X Scope Sources, X Scope Components</td>
<td></td>
</tr>
<tr>
<td>Abstract Obligation</td>
<td>Must or Must Not</td>
<td></td>
<td>Concrete License</td>
<td></td>
</tr>
</tbody>
</table>
License inference scheme
Security license analysis

• License types:
  – Strongly reciprocal, weakly reciprocal, academic, Terms of Service, Proprietary
• Propagation of reciprocal obligations
• Calculating obligations and rights
• Detect conflicting obligations and missing rights
Reasoning structure during analysis

Inference Tree

License Conflicts
- Unavailable Rights
  - Licensee: may: distribute copies of WordProcessor
  - Sought as instantiation of desired right
    - Licensee: may: distribute copies of <AnyBinary>
  - Unavailable because CTL does not grant it and subsumed by
    - Licensee: may: distribute copies of <Any>
- Licensee: may: prepare derivative works of WordProcessor
- Licensee: may: reproduce WordProcessor

Conflicting Obligations
- Licensee: must: sublicense WordProcessor under GPL2.0
  - Licensee: must: sublicense <RightsPatientsStaticScope> under {ThisLicense}
    - Licensee: may: distribute copies of <AnyModifiedBinary>
      - Licensee: may: distribute copies of XMail
    - Licensee: must: sublicense <RightsPatientsStaticScope> under {ThisLicense}
      - Licensee: may: distribute copies of <AnyModifiedBinary>
        - Licensee: may: distribute copies of GnomeEvolution
  - Licensee: must not: sublicense WordProcessor under GPL2.0
    - Licensee: must not: sublicense <Any> under {AnyLicense}
License Report for the Main Architecture

***Printing all obligations
obligation: MPL2.1d licensee must not delete from original code
obligation: MPL3.1 licensee must retain copyright notice
obligation: MPL3.2 licensee must redistribute source code
obligation: CTRL2 licensee must not redistribute source code
obligation: GPL2 licensee must redistribute source code

***Printing all conflicting obligations
obligation: CTRL2 licensee must not redistribute source code
obligation: MPL3.2 licensee must redistribute source code
obligation: GPL2 licensee must redistribute source code
obligation: CTRL2 licensee must not redistribute source code

***Printing all rights
right: MPL3.6 licensee may distribute Covered Code in executable form
right: MPL2.1 licensee may reproduce original code
right: MPL1 licensee may redistribute executable
right: CTRL3 licensee may not reproduce original code
right: CTRL3A licensee may not use Licensor's name, logo, or trademarks
right: CTRL4 licensee may not redistribute executable
right: GPL1 licensee may redistribute executable

***Printing all intersecting rights

License Report for SubArchitecture: Mozilla SubArch

***Printing all obligations
obligation: MPL2.1d licensee must not delete from original code
Challenges of securing open architecture C2 systems
Emerging OA/OSS Security challenges

- Security threats to software systems are increasingly multi-modal and distributed across system components.
- Physically isolated systems are vulnerable to external security attacks.
- What makes an OA C2 system secure changes over time, as new threats emerge and systems evolve.
- Need an approach to continuously assure the security of evolving OA C2 systems that is practical, scalable, robust, tractable, and adaptable.
OA and OSS systems/components evolve: what to do about security?

- Individual components evolve via revisions (e.g., security patches)
- Individual components are updated with functionally enhanced versions;
- Individual components are replaced by alternative components;
- Component interfaces evolve;
- System architecture and configuration evolve;
- System functional and security requirements evolve;
- System security policies, mechanisms, security components, and system configuration parameter settings also change over time.
Current security approaches

- Mandatory access control lists, firewalls;
- Multi-level security;
- Authentication (including certificate authority and passwords);
- Cryptographic support (including public key certificates);
- Encapsulation (including virtualization), hardware confinement (memory, storage, and external device isolation), and type enforcement capabilities;
- Secure programming practices;
- Data content or control signal flow logging/auditing;
- Honey-pots, traps, sink-holes;
- Security technical information guides (STIGs) for configuring the security parameters for applications and operating systems;
- Functionally equivalent but diverse multi-variant software executables.
- Software component security assurance processes.
Current approaches to software system security do not address the challenges of continuously evolving OA C2 systems emerging within agile, adaptive software ecosystems!
Case Study:
Securing the development and evolution of an OA C2 system within an agile, adaptive software ecosystem
Carefully specifying security policy obligations and rights

- The obligation for a user to verify his/her authority to see compartment T, by password or other specified authentication process
- The obligation for all components connected to specified component C to grant it the capability to read and update data in compartment T
- The obligation to reconfigure a system in response to detected threats, when given the right to select and include different component versions, or executable component variants.
- The right to read and update data in compartment T using the licensed component
- The right to add, update, replace specified component D in a specified configuration
- The right to add, update, or remove a security mechanism
- The right to update security policy L.
OA/OSS product lines?

- When functionally similar software components, connectors, or configurations exist,
- Such that equivalent alternatives, versions, or variants may be substituted for one another, then
- We have a strong relationship among these OA system elements that is called a software product line.
- Software product lines for OA systems enable support from agile, adaptive software (component) ecosystems

Design-time view of an OA system
Software product line of functionally similar OA system alternatives

Design-time architecture:
Browser, WP, calendar

Instance architecture:
Firefox, AbiWord, Evolution, Fedora

Mozilla Public License (MPL)
Mozilla Public License (GPL)

Instance architecture:
Firefox, Google Cal., Google Docs, Fedora

Google Public License (GPL)

Instance architecture:
Firefox, Google Cal., Google Docs, Windows

Mozilla Public License (MPL), Google ToS, MS EULA

Instance architecture:
Opera, Google Docs, Evolution, OSX

Opera EULA, Google ToS, Apple License
Product line selection of one alternative system configuration
A security capability specification encapsulating the design-time configuration via multiple virtual machine containers.
Build-time view of OA design selecting proprietary product family alternatives
Build-time view of OA design selecting OSS product family alternatives
Run-time deployment view of OA system family member configuration
Evolution-time software changes
Evolutionary redevelopment, reinvention, and redistribution

• Overall evolutionary dynamic of many OSSD projects is *reinvention and redevelopment*
  • Reinvention enables continuous improvement and collective learning
• OSS evolve through minor mutations
  • Expressed, recombined, redistributed via incremental releases
• OSS systems *co-evolve* with their development community
  • Success of one depends on the success of the other
Product line selection of different functionally similar alternative
Evolved run-time deployment view of a functionally similar alternative OA system configuration
Emerging Transformations with OSS and OA systems
New paths for software platform development and evolution using intra-app components: (GOSS) Widgets/apps acquired from online App Store
Ozone Widget Framework (OWF)

User authenticate with

Identity Provider
Provides identity and authentication services

Accepts users authenticated by

OWF Framework Provider
Provides an OWF framework instance to a population of users

OWF Widget Provider
Provides widgets to multiple framework providers

OWF Widget Providers can belong to different domains

User Browser

Renders widget

Widget Directory

Publishes widget to

Widget Servers

Data

Renders framework

Framework Server
Transforming to multi-party acquisition of software elements within OA ecosystems

Customer/end-user organizations now looking for ways to reduce acquisition cost and effort through shared development/use of common OA software system components (apps, widgets).
Transforming to multi-party acquisition of software elements within OA ecosystems

Customer/end-user organizations now looking for ways to reduce acquisition cost and effort through shared development/use of assembled capabilities for GOSS-based C3CB systems
Figure 1: PEO C4I Storefront Operational Concept

Shared development of Apps and Widgets as OA system components

Ozone Widget Framework for Web PCs and Mobile Devices
Commercial mobile apps also being used (middleware services, not shown)
Enterprise-to-Mobile Middleware *IP Licenses*
(for the NitroDesk *Touchdown* product in 2014)

* LGPL 2.1
* Ical4j from Ben Fortuna
* Public Domain Declaration
* Apache 2
* The Legion of the Bouncy Castle
* Creative Commons BY

* Sony Mobile
* Jesse Anderson
* OpenSSL
* Apple Non-Exclusive
* SQLite
* Microsoft Public License
Growing diversity of challenges in OA cybersecurity

Shared development of Apps and Widgets as OA system components: Cybersecurity?

Ozone Widgets supporting “Bring Your Own Devices” (BYOD)?
New business models for OA software components

- Franchising
- Enterprise licensing
- Metered usage
- Advertising supported
- Subscription
- Free component, paid service fees
- Federated reciprocity for shared development
- Collaborative buying
- Donation
- Sponsorship
- (Government) open source software
- and others

Managing software costs will be demanding. Software acquisition workforce will need automated assistance, else acquisition management costs may dominate development costs for OA software components!
Discussion, Challenges, New Practices, and Conclusions
Emerging challenges in achieving OA software systems

- Program managers/staff *may not understand* how software IP licenses affect OA system design, and vice-versa.

- Software IP and cybersecurity obligations and rights propagate across system development, deployment, and evolution activities *in ways not well understood* by system developers, integrators, end-users, or software acquisition managers.

- *Failure to understand* software IP and cybersecurity obligations and rights propagation can reduce software buying power, increase software life cycle costs, and reduce competition.

- Government agencies *would financially and administratively benefit* from engaging the development and deployment of an (open source) automated software obligations and rights management system.
New practices to realize cost-effective acquisition of OA software systems

- Need to R&D *worked examples* of reference OA system models, and component evolution alternatives.

- Need *open source models of* app/widget security assurance *processes and* reusable cybersecurity *requirements*.

- Need precise *domain-specific languages* (DSLs) and *automated analysis tools* for continuously assessing and continuously improving cybersecurity and IP requirements for OA C2 systems composed from apps/widgets.
Conclusions

- OSSD is a compelling concept for research and practice in Global Software Engineering.
- Our research identifies how new OSS and OA software component technologies, IP and security requirements, and new business models interact in ways that either drive-down or drive-up software life cycle costs.
- New technical risks for component-based OA software systems can dilute the cost-effectiveness of OSSD efforts.
- Need R&D leading to automated systems that can model and analyze OA system IP licenses and cybersecurity requirements.
Acknowledgements

Research collaborators (partial list)

• Mark Ackerman, UMichigan, Ann Arbor; Kevin Crowston, Syracuse U; Les Gasser, UIllinois, Urbana-Champaign; Chris Jensen, Google; Greg Madey, Notre Dame U; John Noll, LERO; Megan Squire, Elon U; and others.

• Thomas Alspaugh, Hazel Asuncion (UWashington-Bothwell), Margaret Elliott, and others at the UCI ISR.

Funding support (No endorsement, review, or approval implied).

• National Science Foundation: #0083075, #0205679, #0205724, #0350754, #0534771, #0749353, #0808783, and #1256593.

• Naval Postgraduate School
  – Acquisition Research Program (2007-2016+)
    • N00244-1-16-0004 (2016-2017)
  – Center for the Edge Research Program (2010-2012).

• Computing Community Consortium (2009-2010).
Thank you!