Beyond Open Architecture: Issues, Challenges, and Opportunities in Open Source Software Development

Walt Scacchi

Tutorial presented at the

2016 International Conference on Global Software Engineering

Irvine, CA

3 August 2016

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

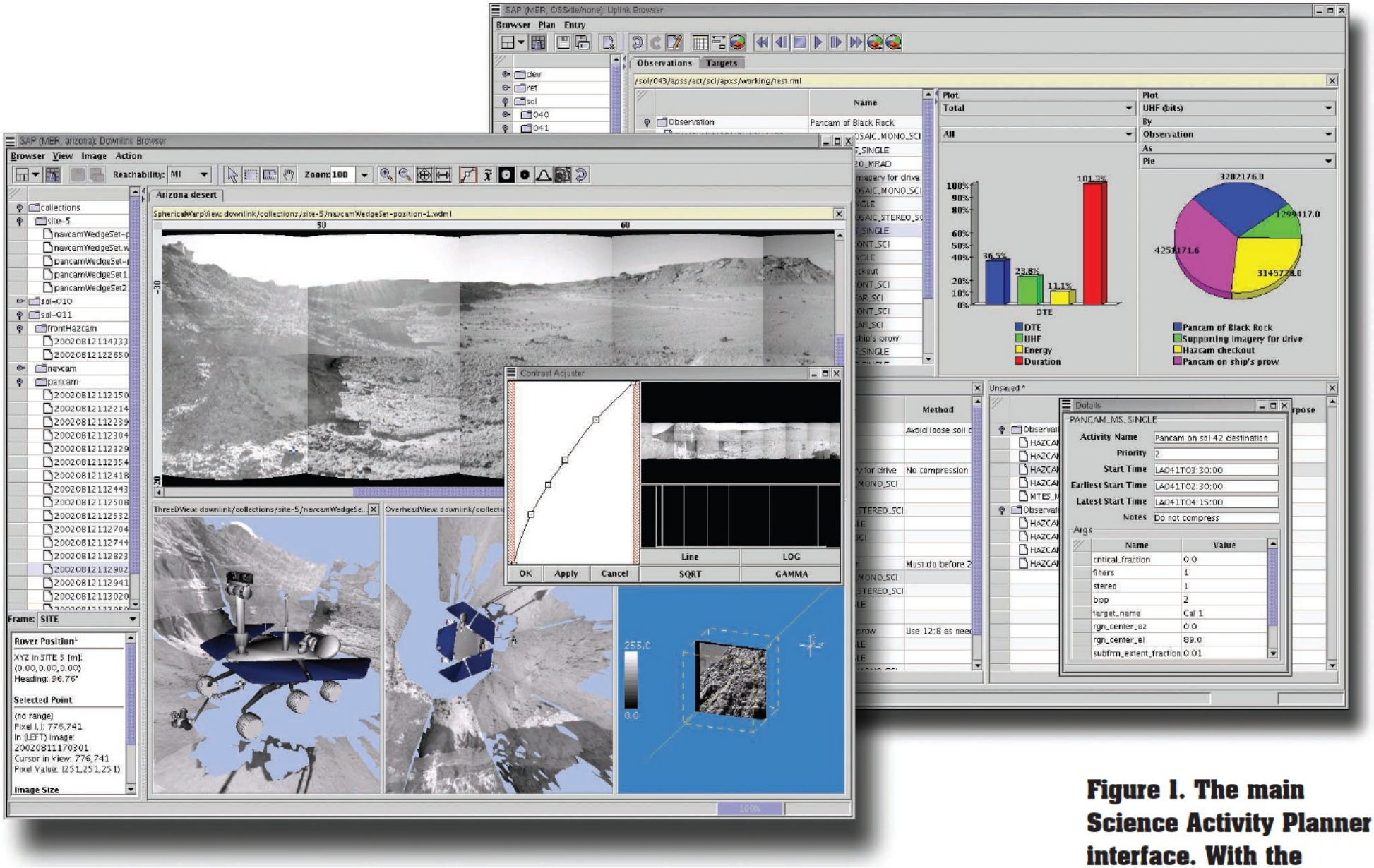


Figure 1. The main Science Activity Planner interface. With the downlink browser

Source: J.S. Norris, Mission Critical Development of Open Source Software: Lessons Learned, *IEEE Software*, 21(1), 42-49, Jan 2004.

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.

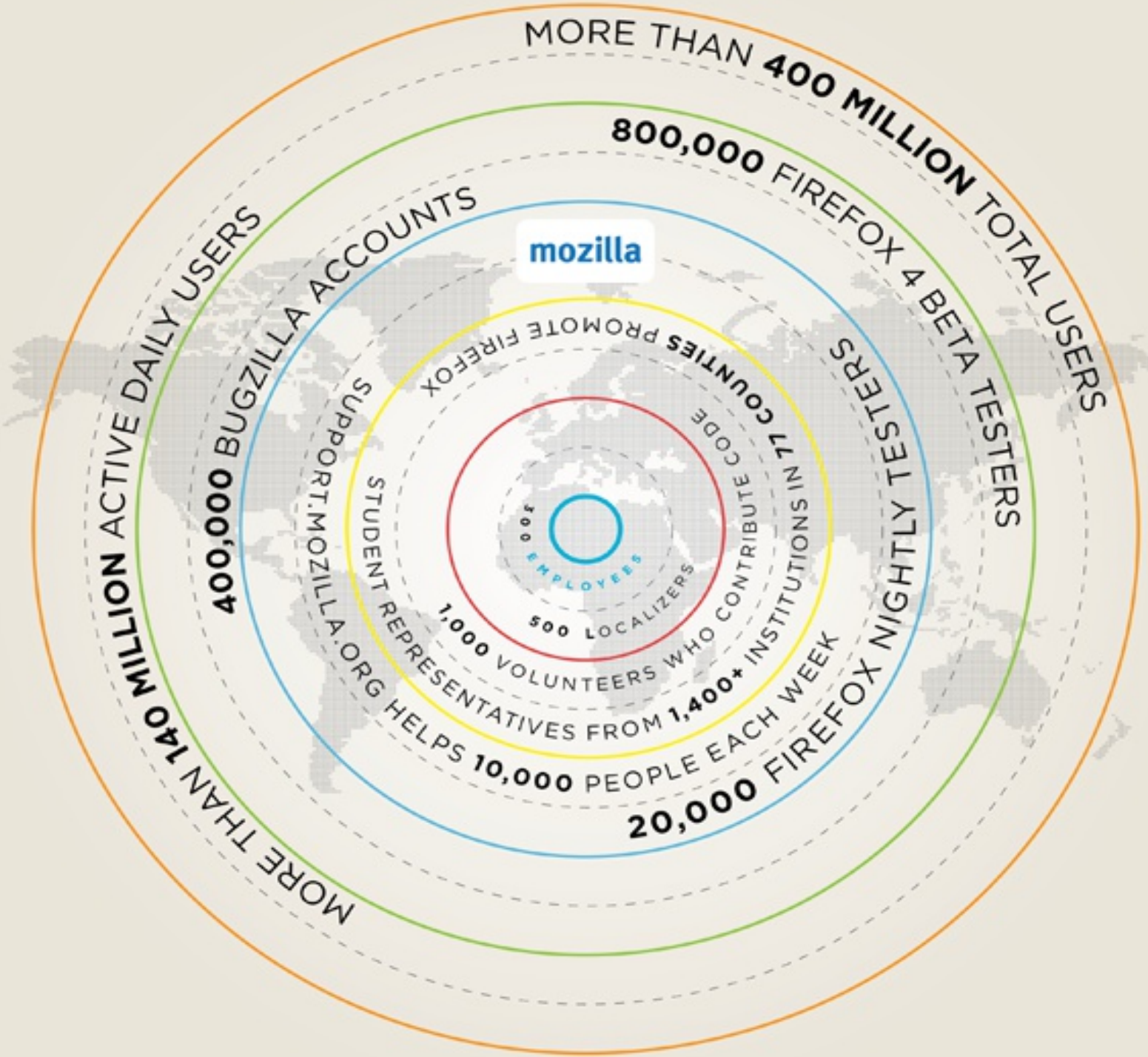
http://ar.libreoffice.org/	العربية	Arabic
http://bo.libreoffice.org/	བོད་ཡིག	Tibetan
http://cs.libreoffice.org/	čeština	Czech
http://da.libreoffice.org/	Dansk	Danish
http://de.libreoffice.org/	Deutsch	German
http://el.libreoffice.org/	Ελληνικά	Greek
http://eo.libreoffice.org/	Esperanto	Esperanto
http://es.libreoffice.org/	Español	Spanish
http://et.libreoffice.org/	Eesti keel	Estonian
http://fa.libreoffice.org/	فارسی	Persian
http://fi.libreoffice.org/	Suomi	Finnish
http://fr.libreoffice.org/	Français	French
http://gl.libreoffice.org/	Galego	Galician
http://he.libreoffice.org/	עברית	Hebrew
http://hi.libreoffice.org/	हिन्दी	Hindi
http://hu.libreoffice.org/	Magyar	Hungarian
http://it.libreoffice.org/	Italiano	Italian
http://ja.libreoffice.org/	日本語	Japanese
http://lo.libreoffice.org/	□□□□□□□	Lao
http://lt.libreoffice.org/	Lietuvių kalba	Lithuanian

Most downloads over all time

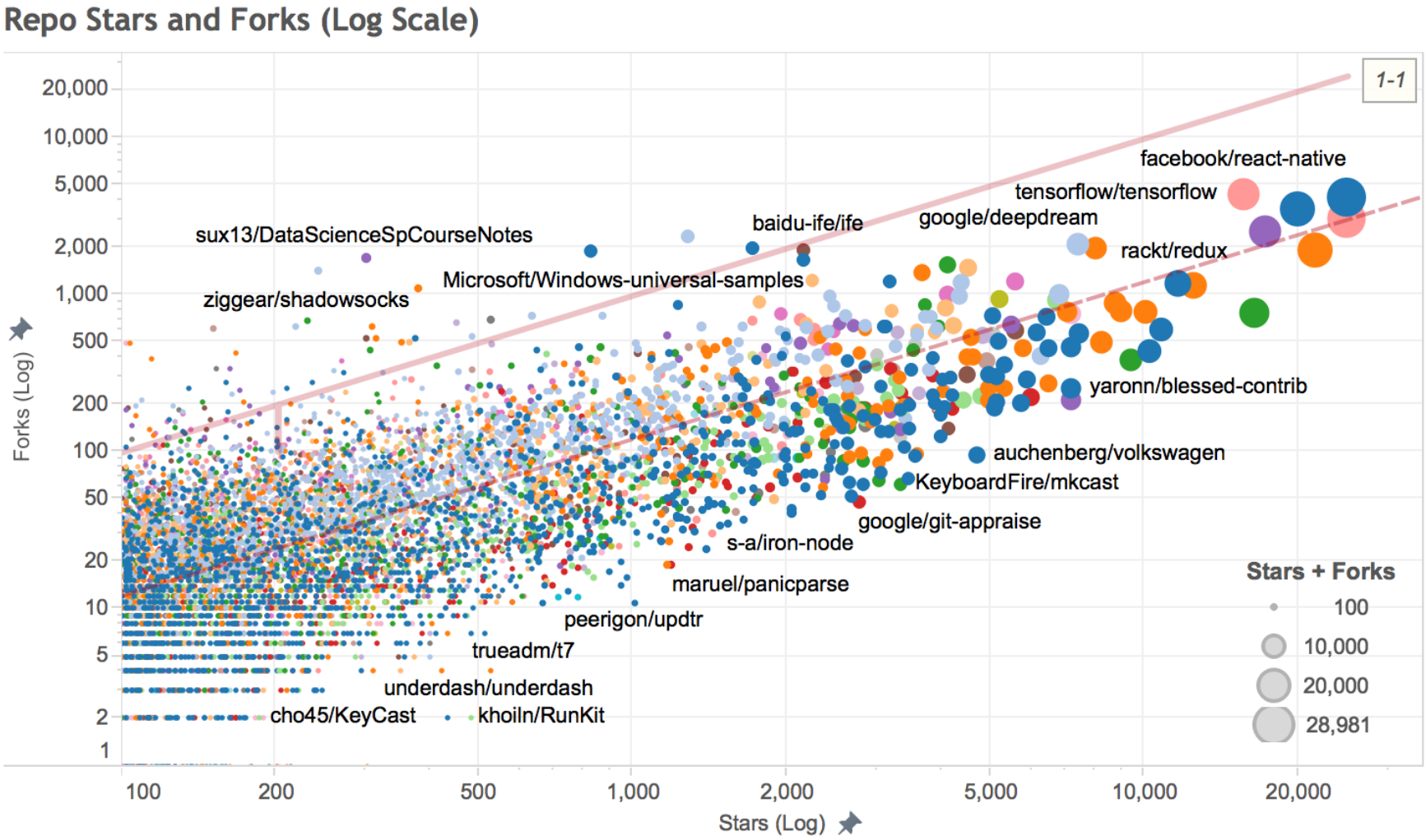
Rank	Project Name	Downloads
1	Microsoft's TrueType core fonts	1.4B
2	Notepad++ Plugin Manager	993.6M
3	VLC media player	897.1M
4	eMule	682.1M
5	Vuze - Azureus	542.1M
6	MinGW - Minimalist GNU for Windows	435.2M
7	FileZilla	413.6M
8	7-Zip	409.1M
9	Ares Galaxy	406.5M
10	PortableApps.com: Portable Software/USB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

Most downloads last week

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


MORE THAN 400 MILLION TOTAL USERS
800,000 FIREFOX 4 BETA TESTERS
20,000 FIREFOX NIGHTLY TESTERS
MORE THAN 140 MILLION ACTIVE DAILY USERS
400,000 BUGZILLA ACCOUNTS
mozilla
SUPPORT.MOZILLA.ORG HELPS 10,000 PEOPLE EACH WEEK
STUDENT REPRESENTATIVES FROM 1,400+ INSTITUTIONS IN 17 COUNTRIES PROMOTE FIREFOX
1,000 VOLUNTEERS WHO CONTRIBUTE CODE
500 LOCALIZERS
500 EMPLOYEES

GitHub OSS Repositories (2015 sample)



Users Filter

Languages Filter

(All)

Click to Highlight

- JavaScript
- Java
- Unknown
- Objective-C
- Python
- Swift
- Go
- C++
- HTML
- C
- CSS
- PHP
- Ruby
- C#
- Shell
- Lua

***Data**: Repos created in 2015, >= 100 stars | **Date Range**: 1/1/2015 to 1/1/2016*
***FAQ**: See the final tab "A" for more info*

***Hover**: View more info | **Click**: View repo url*
Interact** with the **filters

NAVIGATION
- Recent posts

ADMIN LOGIN
Username: *
Password: *
LOG IN
Log in using OpenID
Request new password

Like 44 people like this. Sign Up to see what your friends like.

SEARCH
Search this site:
SEARCH

GETTING DATA
- Code Forges Study
- Database schema
- Data collection details
- Direct database access
- Download data (at FLOSSdata)

USING DATA
- Visualizations & Examples
- Mailing List
- How to cite this data
- Research publications using this data

Home » Blogs » megan's blog

How do various corporations populate the Apache projects?

Submitted by megan on July 21, 2016 - 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:

project_name	organization	org dev count	all devs	pct of team
Synapse	WSO2	15	24	62.50
Ignite	GridGain	15	25	60.00
Jackrabbit	Adobe	27	49	55.10
Knox	Hortonworks	15	28	53.57
Geronimo	IBM	33	63	52.38
Ambari	Hortonworks	68	131	51.91
Falcon	InMobi	11	22	50.00
Calcite	Hortonworks	7	16	43.75
REEF	Microsoft	14	32	43.75
Sentry	Cloudera	14	33	42.42
Flume	Cloudera	11	26	42.31
Nutch	NASA JPL	6	15	40.00
OpenJPA	IBM	12	30	40.00
AsterixDB	UC Irvine	11	28	39.29
Airavata	Indiana University	12	32	37.50
Hive	Hortonworks	21	56	37.50
Phoenix	Salesforce	10	27	37.04
Spark	Databricks	16	44	36.36
Avro	Cloudera	5	14	35.71
BVal	IBM	5	14	35.71
Aries	IBM	19	54	35.19
Mesos	Mesosphere	9	26	34.62
Karaf	JBoss	7	23	30.43
OpenMeetings	Unipro	6	20	30.00
Syncope	Tirasa	6	20	30.00

Research findings on OSSD

OSSD Research Surveys

- Scacchi, W. (2007). Free/Open Source Software Development: Recent Research Results and Emerging Opportunities. *Proc. 6th. ESEC/FSE*, 459–468. Also see, Scacchi, W. Free/Open Source Software Development: Recent Research Results and Methods, in M.V. Zelkowitz (ed.), *Advances in Computers*, 69, 243-295, 2007.
- Gasser, L. and Scacchi, W. (2008). Towards a Global Research Infrastructure for Multidisciplinary Study of Free/Open Source Software Development, in *Open Source Development, Community and Quality*; B. Russo, E. Damiani, S. Hissan, B. Lundell, and G. Succi (Eds.), IFIP Vol. 275, Springer, Boston, MA. 143-158.
- Hauge, O., Ayala, C. and Conradi, R. (2010). Adoption of Open Source Software in Software-Intensive Organizations - A Systematic Literature Review. *Information and Software Technology,* 52(11), 1133-1154.
- Aksulu, A. and Wade, M.R. (2010). A Comprehensive Review and Synthesis of Open Source Research, *J. Assoc. Info. Systems*, 11(11), 576-656.
- Scacchi, W., Crowston, K., Jensen, C., Madey, G., Squire, M., and others (2010). *Towards a Science of Open Source Systems*, Final Report, Computing Community Consortium, November 2010. http://foss2010.isr.uci.edu/content/foss-2010-reports/
- Höst, M., Oručević-Alagić, A. (2011). A Systematic Review of Open Source Software in Commercial Software Product Development, *Information and Software Technology*, 53(6), June, 616-624.
- Crowston, K., Wei, K., Howison, J., and Wiggins, A. (2012). Free/libre open source software development: what we know and what we do not know. *ACM Computing Surveys*, 44(2).

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

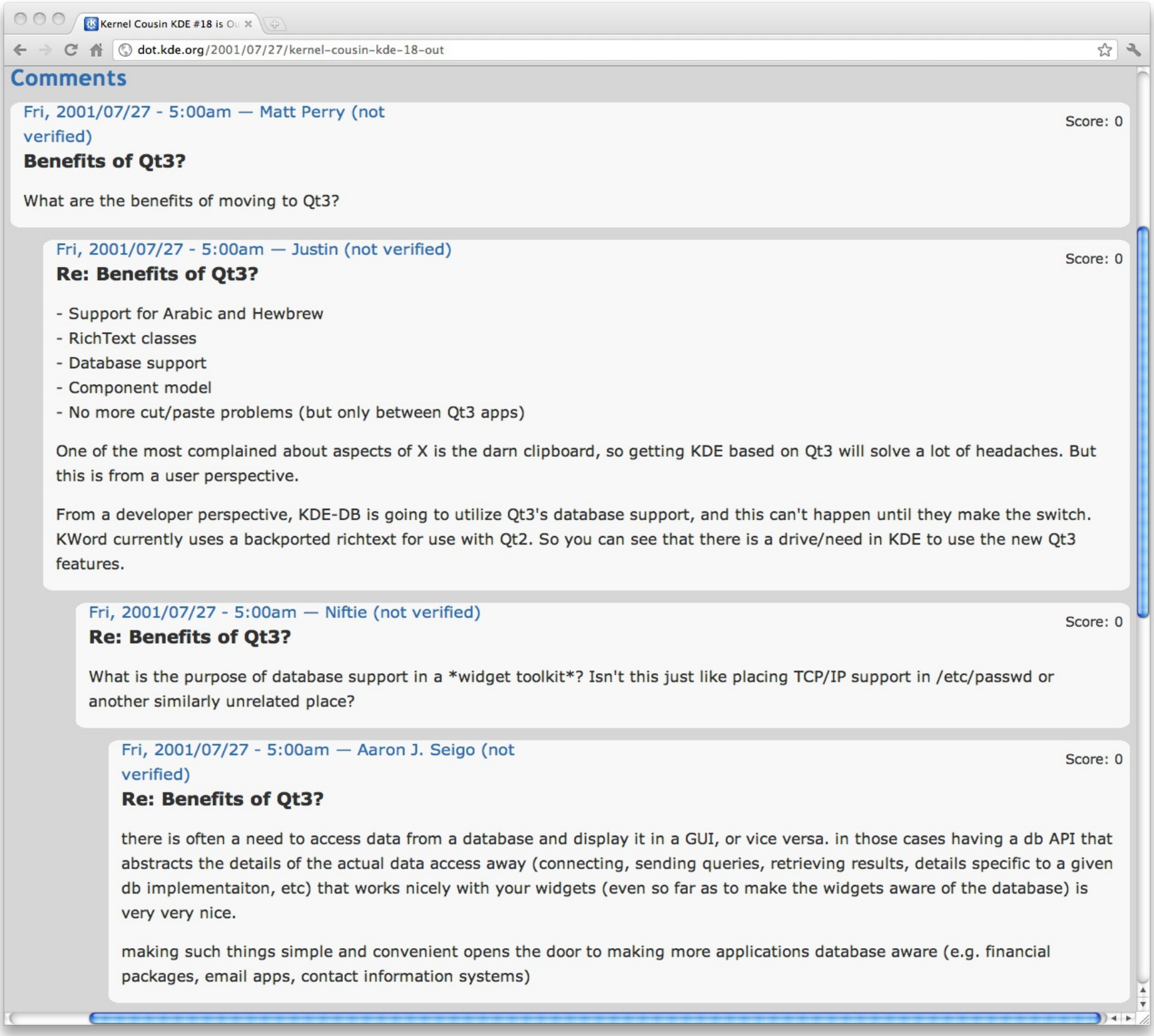
- Sustained commitment of personal or enterprise resources helps *subsidize* OSSD projects via:
  - Personal computer(s)
  - Internet access
  - Hosting personal Web site
  - Hosting project repositories (now on GitHub)
  - Personal choice of software development tools or tool set

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"


Kernel Cousin KDE #18 is Ou
dot.kde.org/2001/07/27/kernel-cousin-kde-18-out
Comments
Fri, 2001/07/27 - 5:00am — Matt Perry (not verified)
Score: 0
Benefits of Qt3?
What are the benefits of moving to Qt3?
Fri, 2001/07/27 - 5:00am — Justin (not verified)
Score: 0
Re: Benefits of Qt3?
- Support for Arabic and Hewbrew
- 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.
Fri, 2001/07/27 - 5:00am — Niftie (not verified)
Score: 0
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?
Fri, 2001/07/27 - 5:00am — Aaron J. Seigo (not verified)
Score: 0
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

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

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

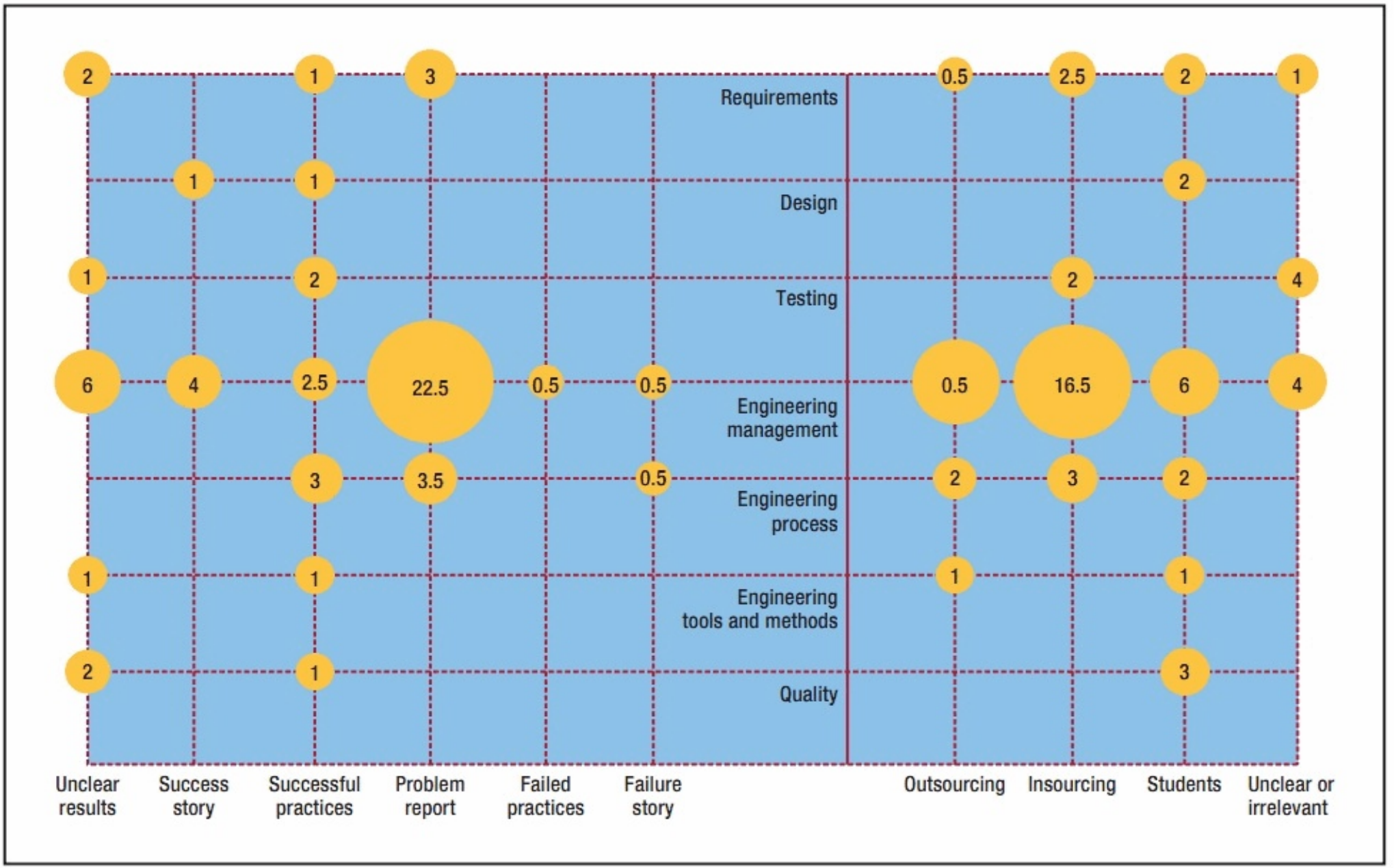


FIGURE 1. Bubble-plot overview of what we know about global software engineering (GSE). Results are based on a systematic review of the GSE literature available from 2000 to 2007.[4] The left side classifies the 59 relevant studies thematically in terms of success or failure, and the right side classifies 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 *meritocractic 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

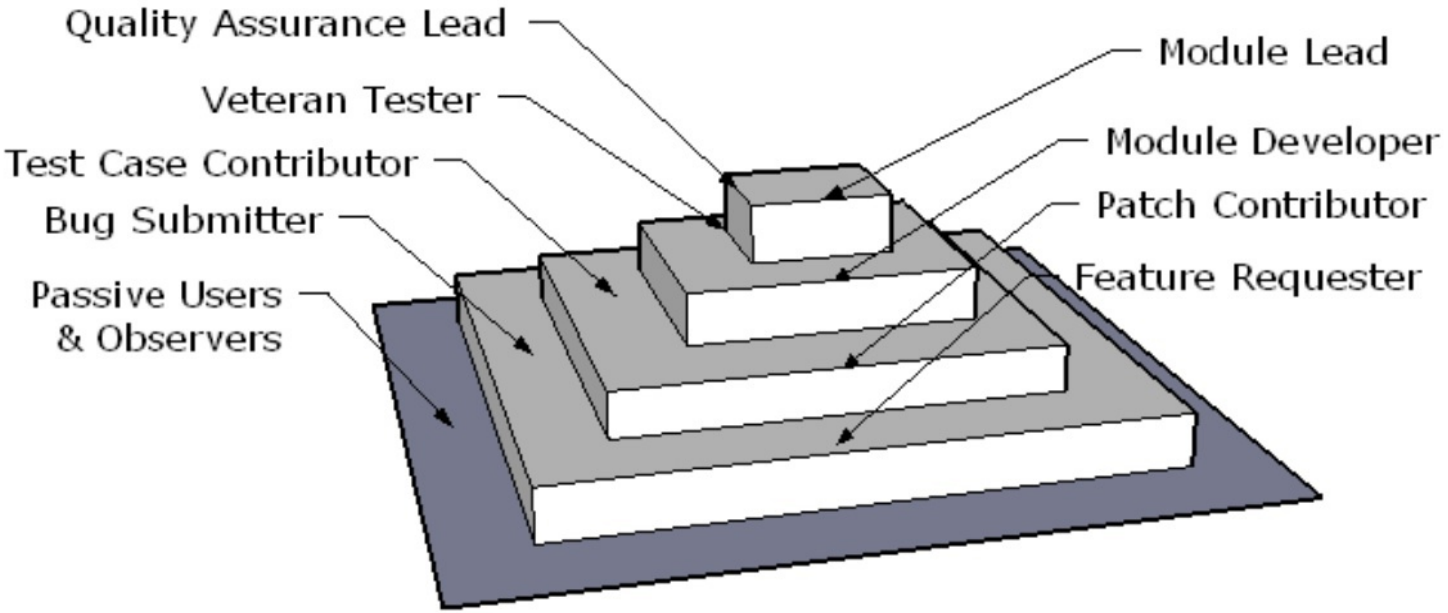


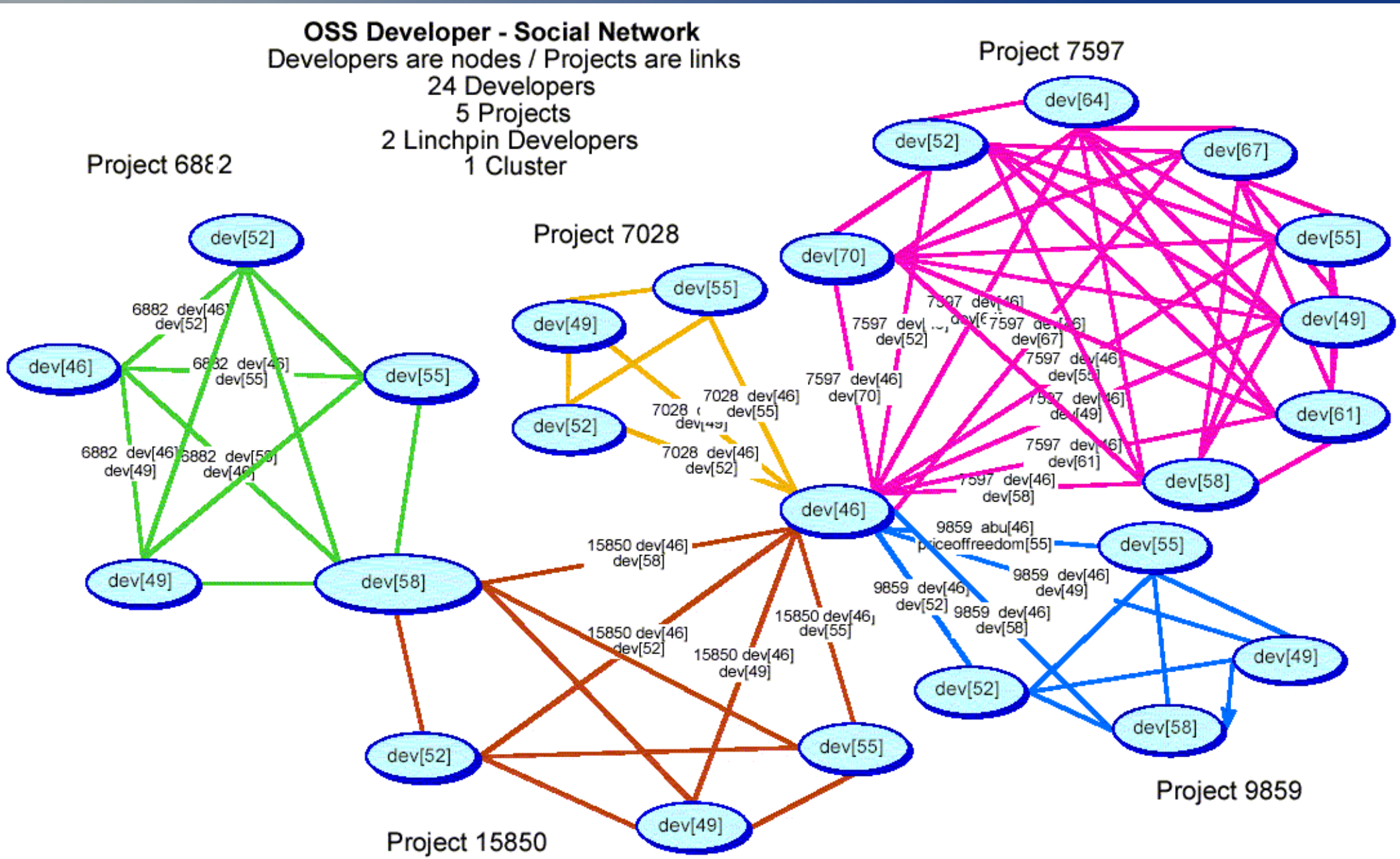
Figure 2. An "onion" pyramid representation of a generic OSSD project organizational hierarchy with multiple role-sets and advancement tracks.

Source: C. Jensen and W. Scacchi, Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, in *Proc. 29th. Intern. Conf. Software Engineering,* Minneapolis, MN, May 2007, 364-374.

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



Source: G. Madey, *et al.,* 2005

OSSD as multi-project software ecosystems

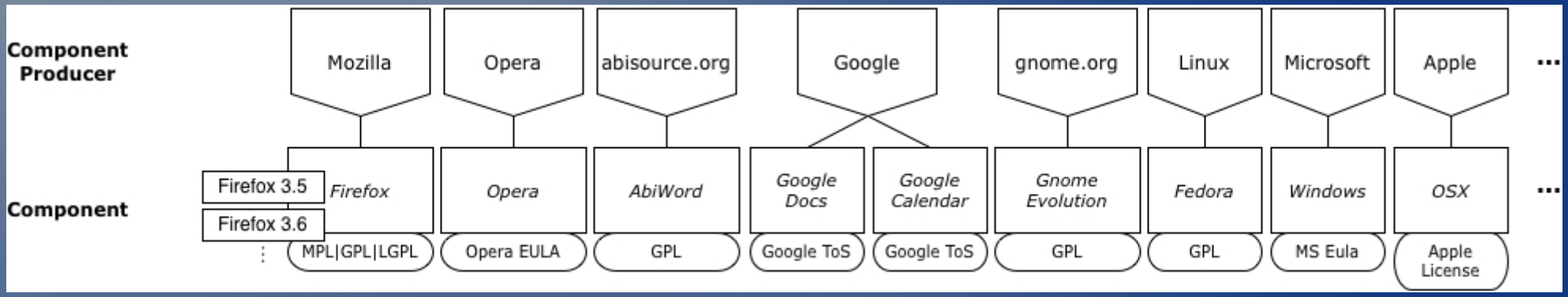
What is a (software) ecosystem?

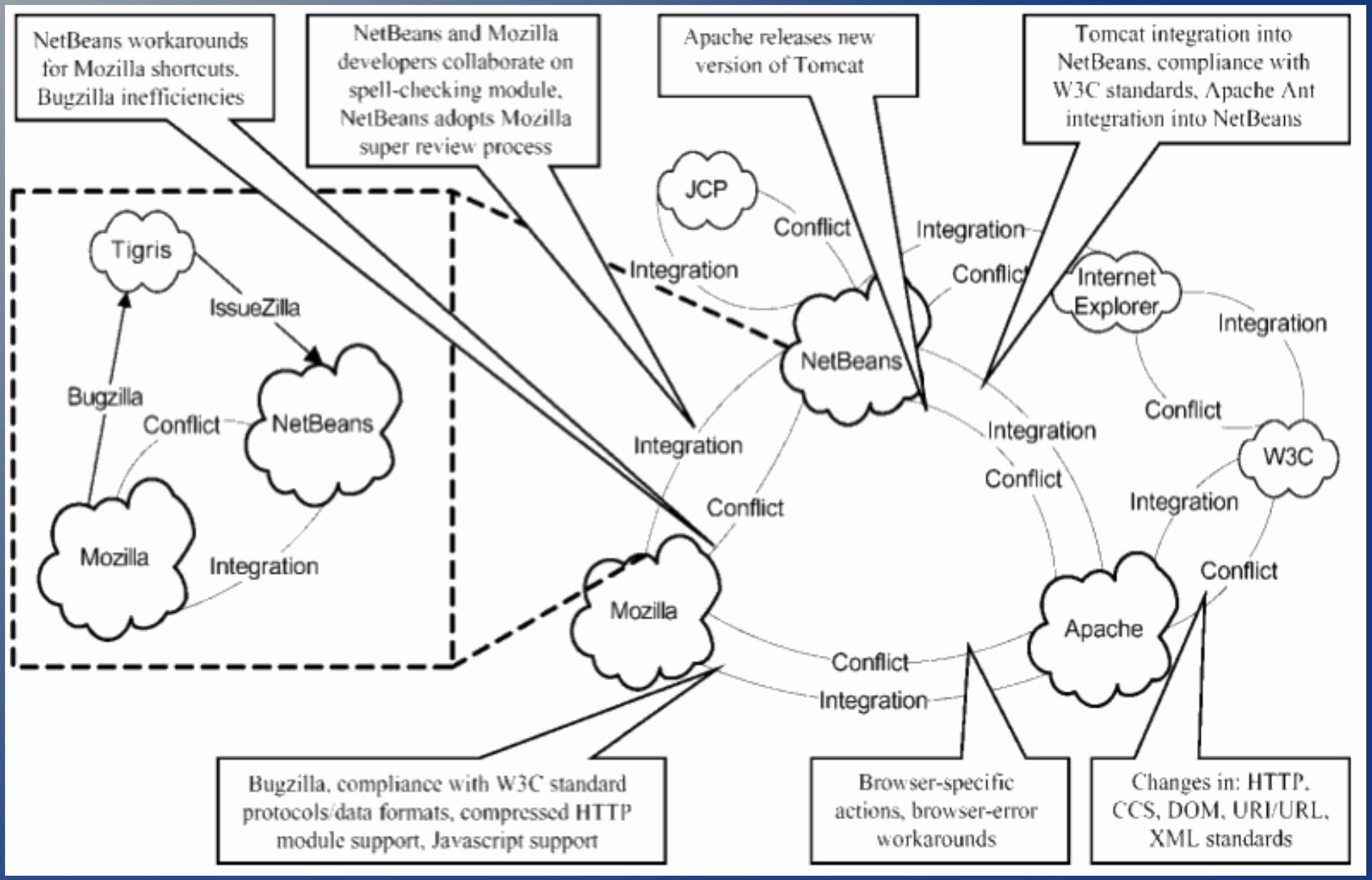
- *Ecology of systems* with diverse species juxtaposed in adaptive prey-predator food chain relationships.
- *Software supply network* of component producers, system integrators, and consumers.
- *Socio-technical network of processes* that transform the flow of resources, enacted by actors in different roles, using tools, to produce products, services, or capabilities.

Multi-project software ecosystem

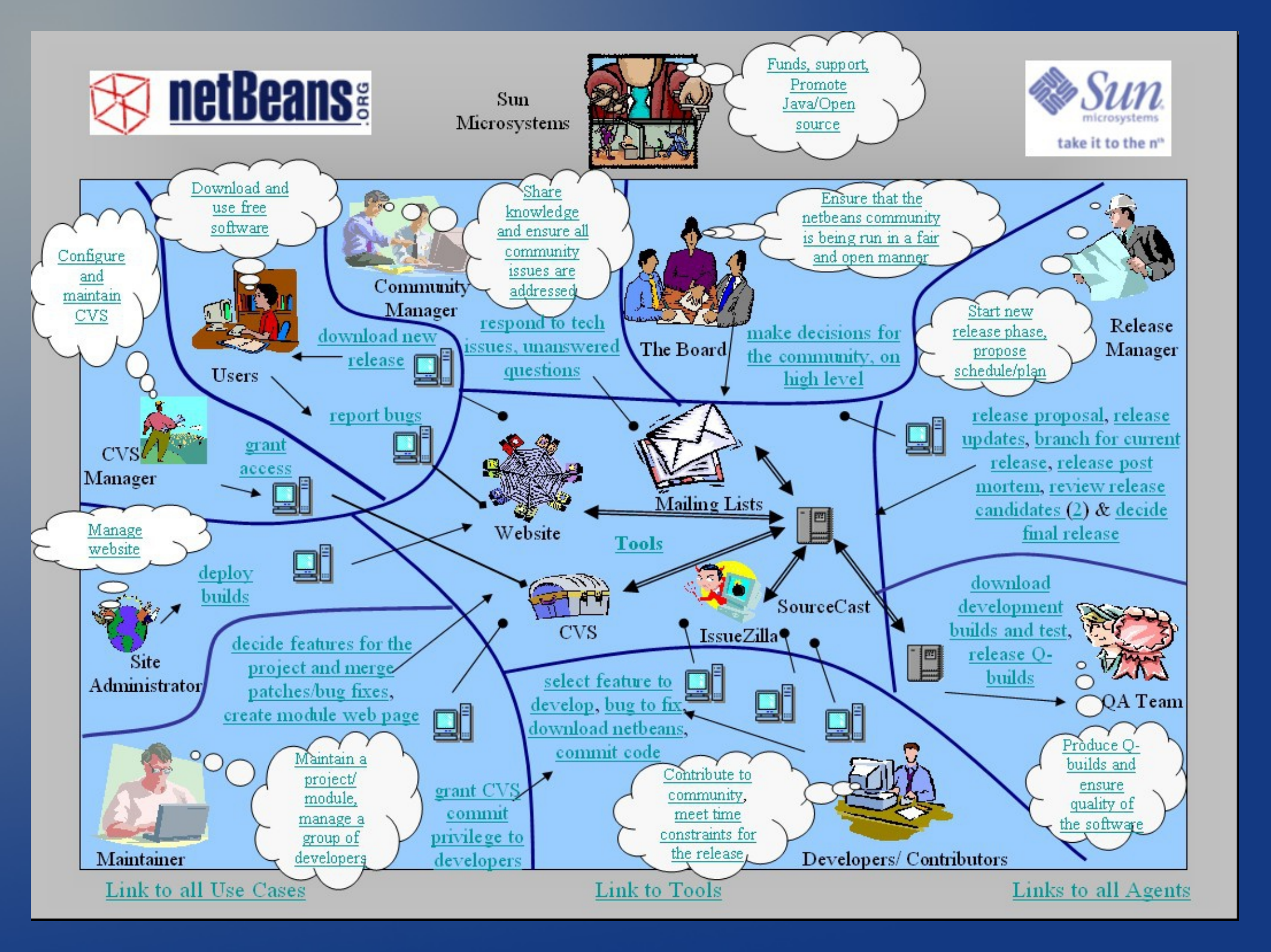
- Mutually dependent OSS development and evolution propagate architectural styles, dependencies, and vulnerabilities
- *Architectural bricolage* arises when autonomous OSSD projects, artifacts, tools, and systems co-mingle or merge
  - Enables discontinuous or *exponential growth* of OSS source code, functionality, complexity, contributions

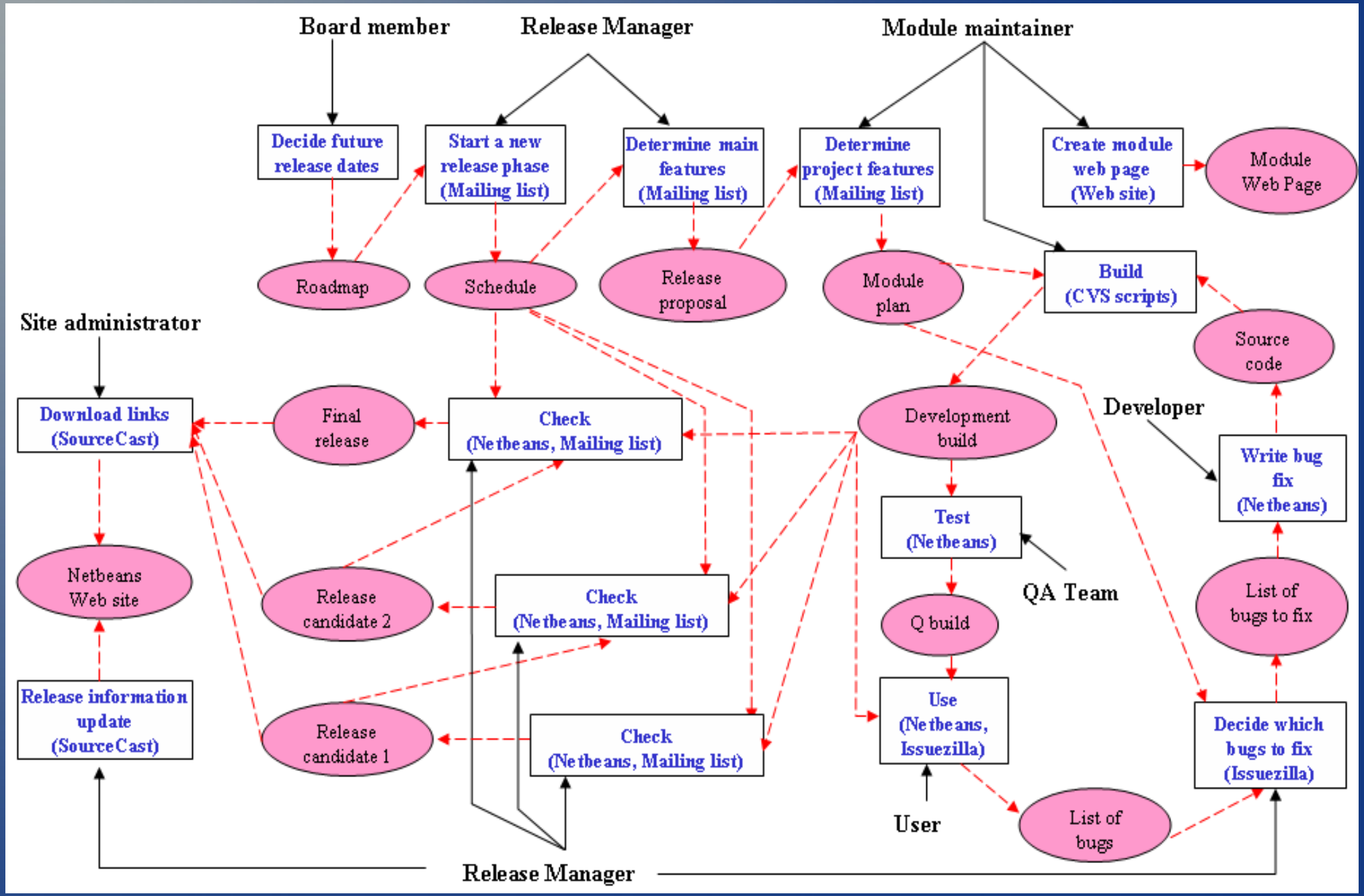
Software ecosystem of producers and the (licensed) software components for Web-compatible enterprise systems



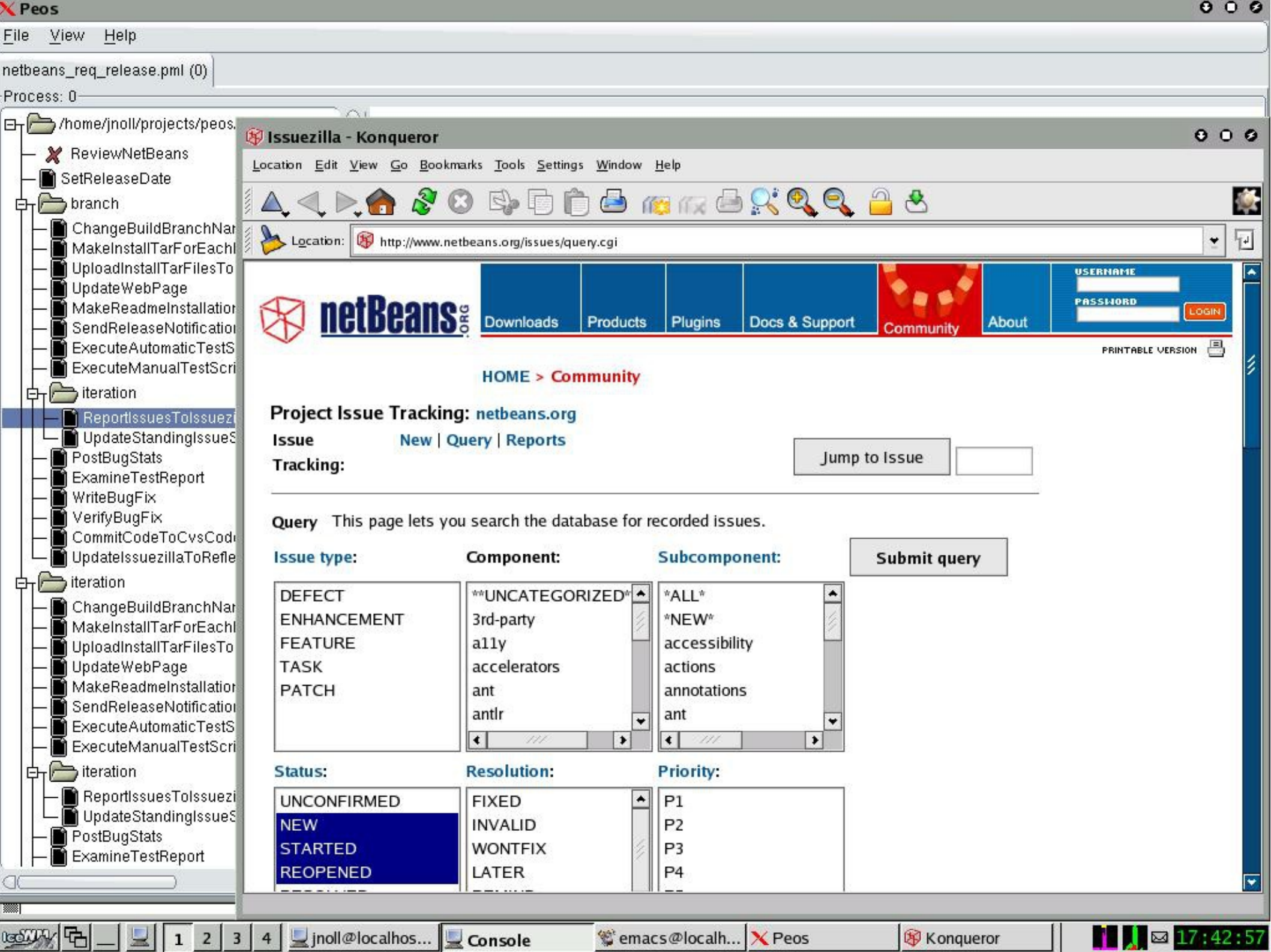


Source: C. Jensen and W. Scacchi, Process Modeling Across the Web Information Infrastructure, *Software Process--Improvement and Practice*, 10(3), 255-272, July-September 2005.


netBeans.ORG
Sun Microsystems
Funds, support, Promote Java/Open source
Sun microsystems
take it to the n
Download and use free software
Configure and maintain CVS
Community Manager
Share knowledge and ensure all community issues are addressed
Ensure that the netbeans community is being run in a fair and open manner
Release Manager
Start new release phase, propose schedule/plan
Users
download new release
respond to tech issues, unanswered questions
The Board
make decisions for the community, on high level
CVS Manager
report bugs
grant access
release proposal, release updates, branch for current release, release post mortem, review release candidates (2) & decide final release
Website
Mailing Lists
Tools
Manage website
deploy builds
CVS
SourceCast
IssueZilla
download development builds and test, release Q-builds
QA Team
Site Administrator
decide features for the project and merge patches/bug fixes, create module web page
select feature to develop, bug to fix, download netbeans, commit code
Maintain a project/ module, manage a group of developers
grant CVS commit privilege to developers
Contribute to community, meet time constraints for the release
Produce Q-builds and ensure quality of the software
Maintainer
Developers/ Contributors
Link to all Use Cases
Link to Tools
Links to all Agents



Legend: Boxes are *activities* (using *informalisms*); Ellipses are *resources* required or provided; Actor *roles* in boldface; *flow dependencies* as arrows.


Peos
File View Help
netbeans_req_release.pml (0)
Process: 0
/home/jnoll/projects/peos.
ReviewNetBeans
SetReleaseDate
branch
ChangeBuildBranchNar
MakeInstallTarForEachI
UploadInstallTarFilesTo
UpdateWebPage
MakeReadmeInstallatior
SendReleaseNotificatio
ExecuteAutomaticTestS
ExecuteManualTestScri
iteration
ReportIssuesToIssuezi
UpdateStandingIssueS
PostBugStats
ExamineTestReport
WriteBugFix
VerifyBugFix
CommitCodeToCvsCod
UpdateIssuezillaToRefle
iteration
ChangeBuildBranchNar
MakeInstallTarForEachI
UploadInstallTarFilesTo
UpdateWebPage
MakeReadmeInstallatior
SendReleaseNotificatio
ExecuteAutomaticTestS
ExecuteManualTestScri
iteration
ReportIssuesToIssuezi
UpdateStandingIssueS
PostBugStats
ExamineTestReport
Issuezilla - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://www.netbeans.org/issues/query.cgi
netBeans.org
Downloads Products Plugins Docs & Support Community About
USERNAME
PASSWORD
LOGIN
PRINTABLE VERSION
HOME > Community
Project Issue Tracking: netbeans.org
Issue Tracking:
New | Query | Reports
Jump to Issue
Query This page lets you search the database for recorded issues.
Issue type:
DEFECT
ENHANCEMENT
FEATURE
TASK
PATCH
Component:
**UNCATEGORIZED*
3rd-party
a11y
accelerators
ant
antlr
Subcomponent:
ALL
NEW
accessibility
actions
annotations
ant
Submit query
Status:
UNCONFIRMED
NEW
STARTED
REOPENED
Resolution:
FIXED
INVALID
WONTFIX
LATER
Priority:
P1
P2
P3
P4
1 2 3 4
jnoll@localhos...
Console
emacs@localh...
Peos
Konqueror
17:42:57

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.

Physical C2 facility

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.

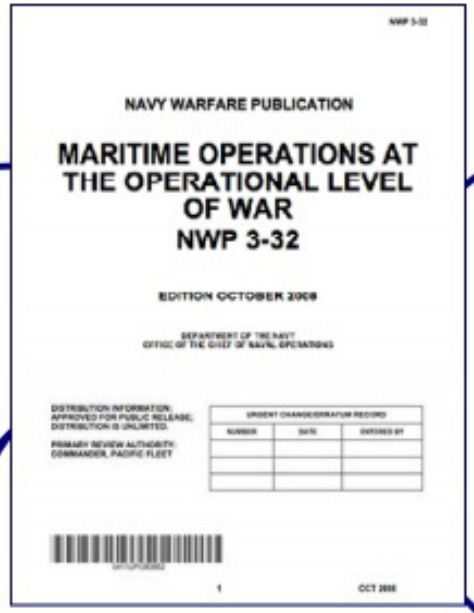
*Scacchi, W. Brown, C. and Nies, K. (2012).
Exploring the Potential of Virtual Worlds for Decentralized Command and Control, *Proc. 17th Intern. Command and Control Research & Technology Symposium*, Paper-096, Alexandria, VA.

Recurring vision for C2 systems [c. 2010]

C2BMC
MIPS

(i.e. BMD)

Planning



Monitoring (SA)

GCCS (COP)

JFMCC Supporting Plan

To JTF Counterstrike
PLAN ALFA

PowerPoint slides & Word Documents

Directing

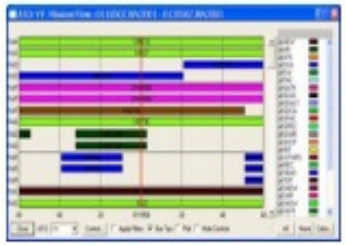
Execution

J/ADOCS (Fires)

Outlook .PST files

Chat Windows

Excel Spreadsheets



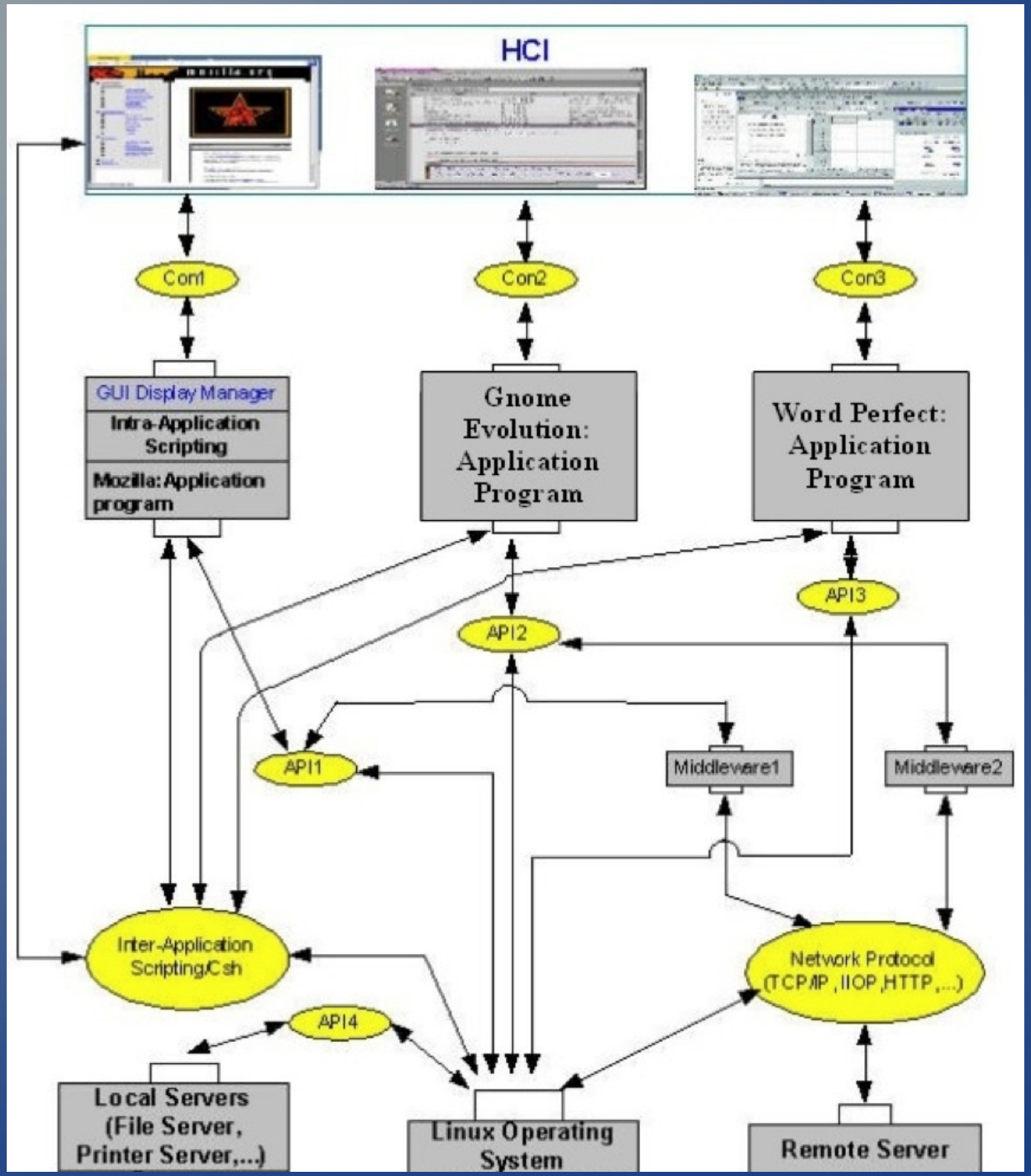
TBMCS (ATO)

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?

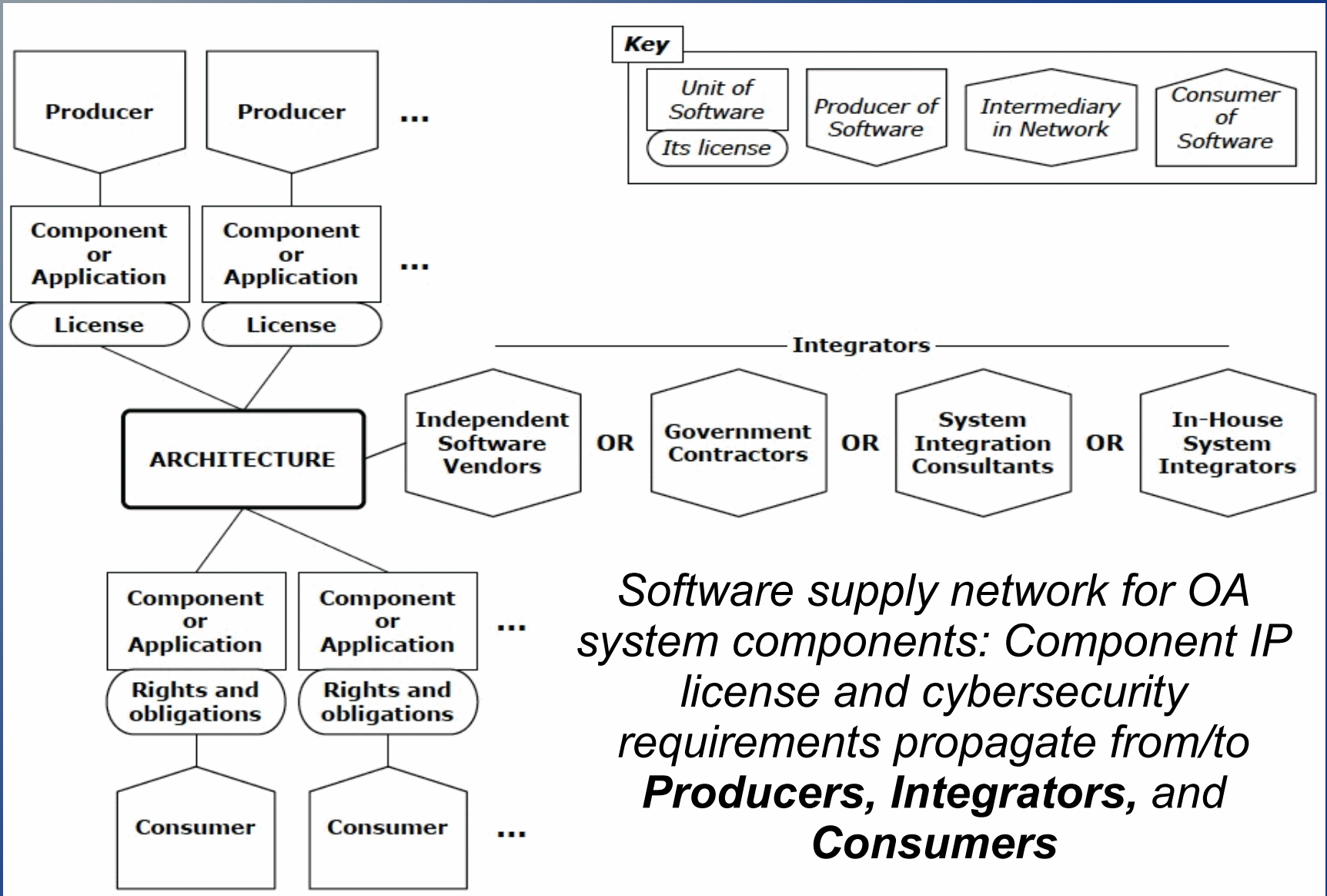
Open Architecture (OA) Software Concepts

- 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?

Source: W. Scacchi and T. Alspaugh, Emerging Issues in the Acquisition of Open Source Software within the U.S. Department of Defense, *Proc. 5th Annual Acquisition Research Symposium*, Vol. 1, 230-244, NPS-AM-08-036, Naval Postgraduate School, Monterey, CA, 2008.

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

Source: Scacchi W. and Alspaugh, T. (2012). Understanding the Role of Licenses and Evolution in Open Architecture Software Ecosystems, *Journal of Systems and Software*, 85(7), 1479-1494, July.

OSS elements subject to different IP licenses

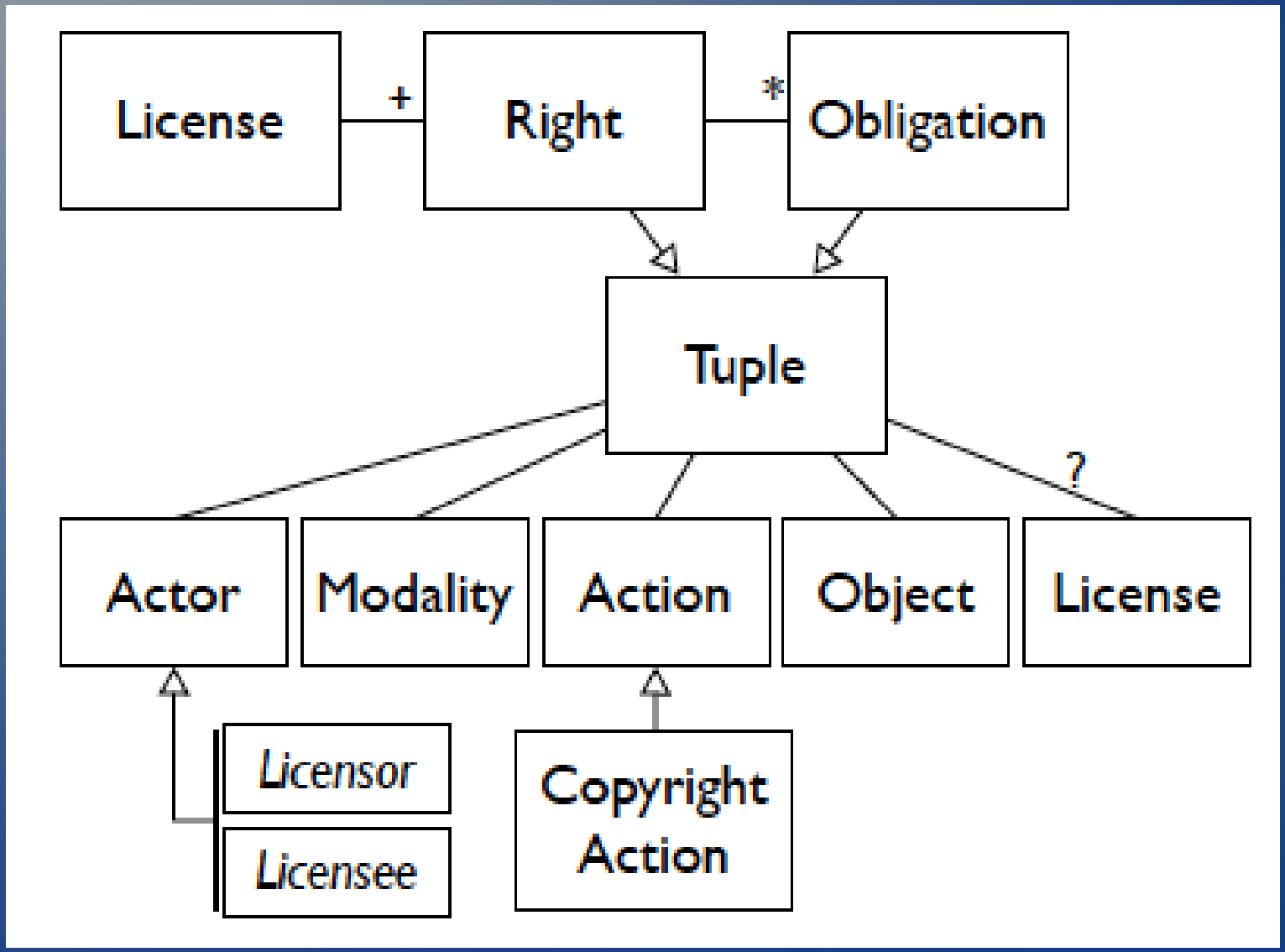
- How to determine which obligations and rights apply to a configured system at what time?
  - At *design-time* (maximum flexibility)
  - At *build-time (*may/not be able to redistribute components at hand)
  - At *run-time* (may/not need to install/link-to components from other sources)
  - At *evolution-time*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

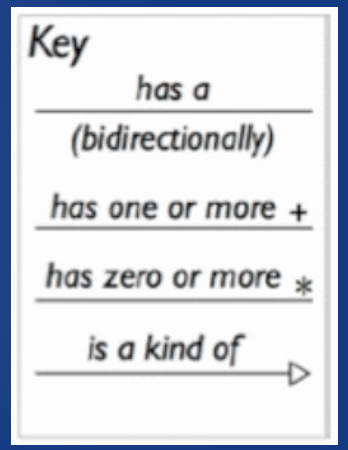
Source: Scacchi W. and Alspaugh, T. (2012). Understanding the Role of Licenses and Evolution in Open Architecture Software Ecoystems, *Journal of Systems and Software*, 85(7), 1479-1494, July.

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

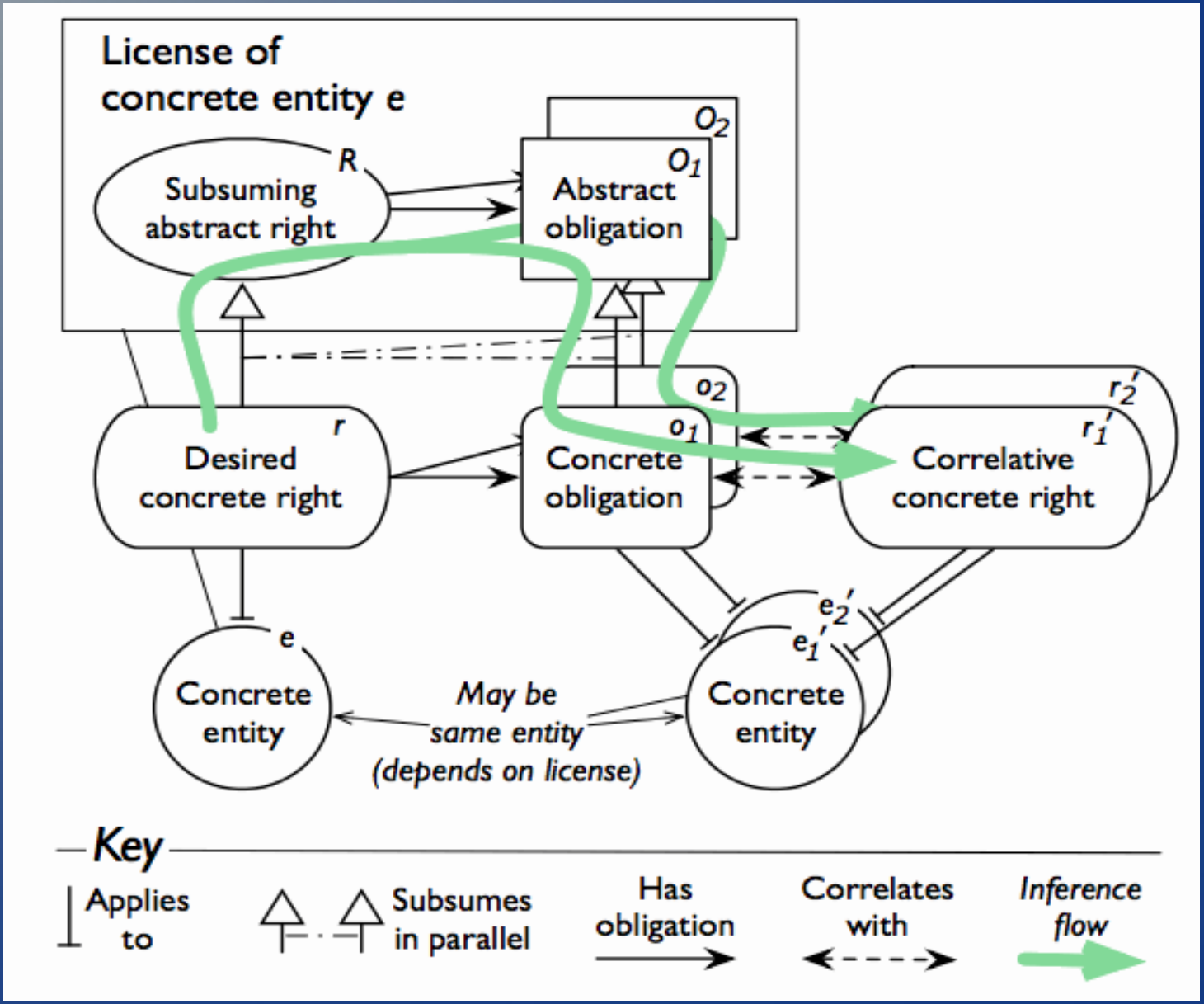




Logical modality and objects of software license rights and obligations constraints

	Actor	Modality	Action	Object	License (optional)
Abstract Right	*Licensee or Licensor*	*May or Need Not*	The set of actions is large, comprising whatever actions the licenses in question utilize	*Any Under This License*	*This License or Object's License*
				Any Source Under This License	
				Any Component Under This License	
Concrete Right				Concrete Object	Concrete License
Concrete Obligation					
Abstract Obligation		*Must or Must Not*		*Right's Object*	*Concrete License or Right's License*
				All Sources Of Right's Object	
				X Scope Sources	
				X Scope Components	

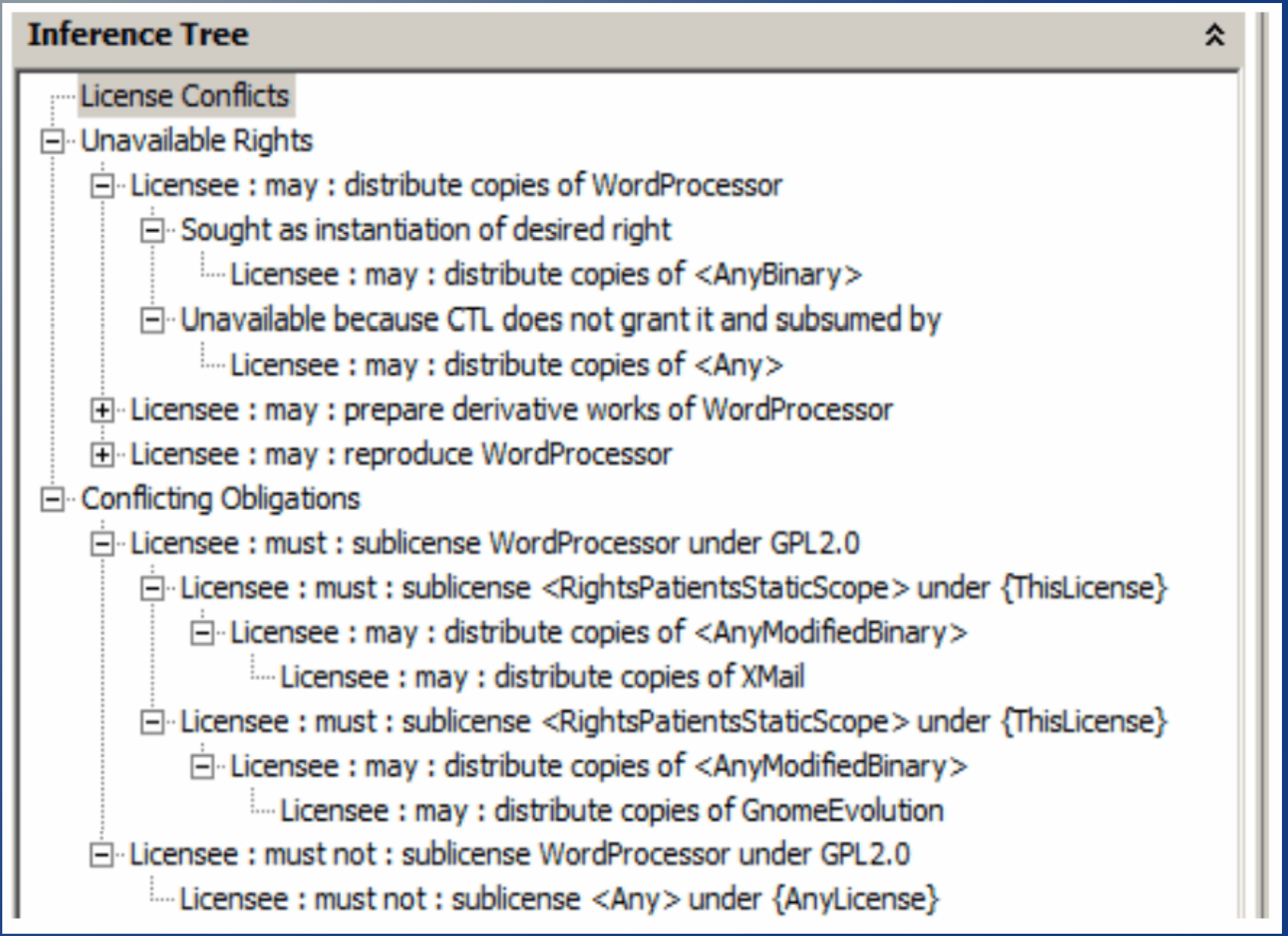
License inference scheme

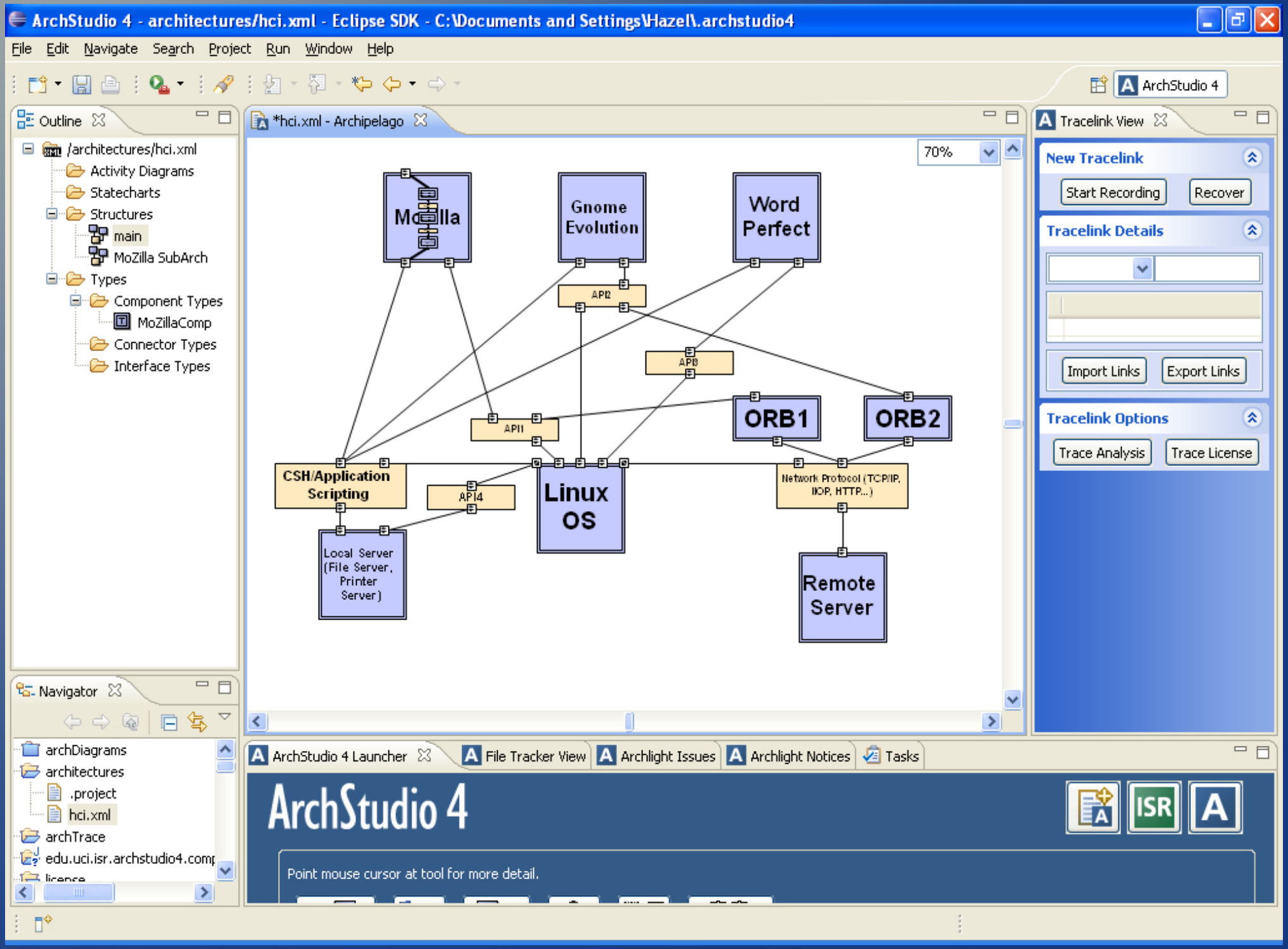


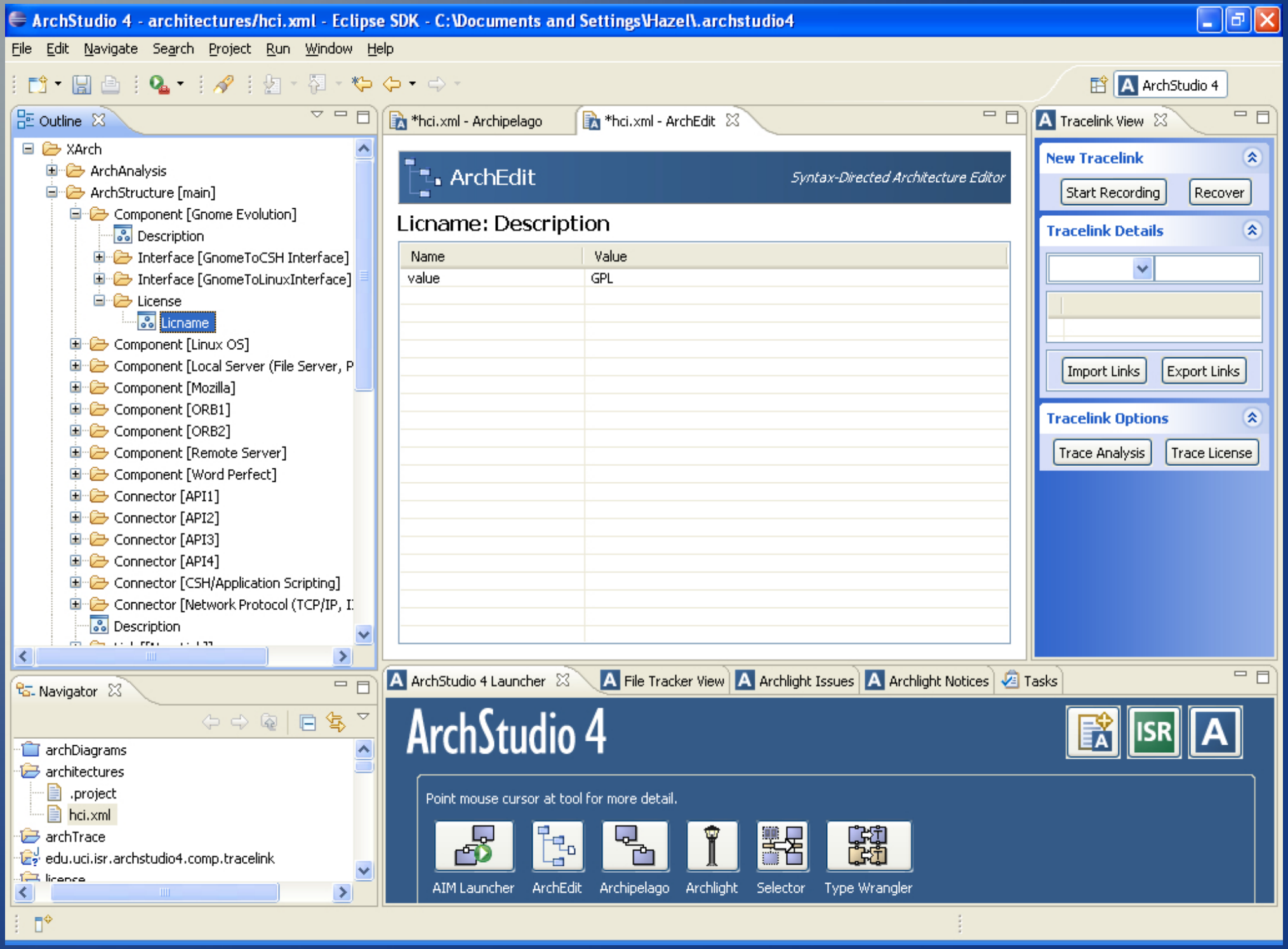
Security license analysis

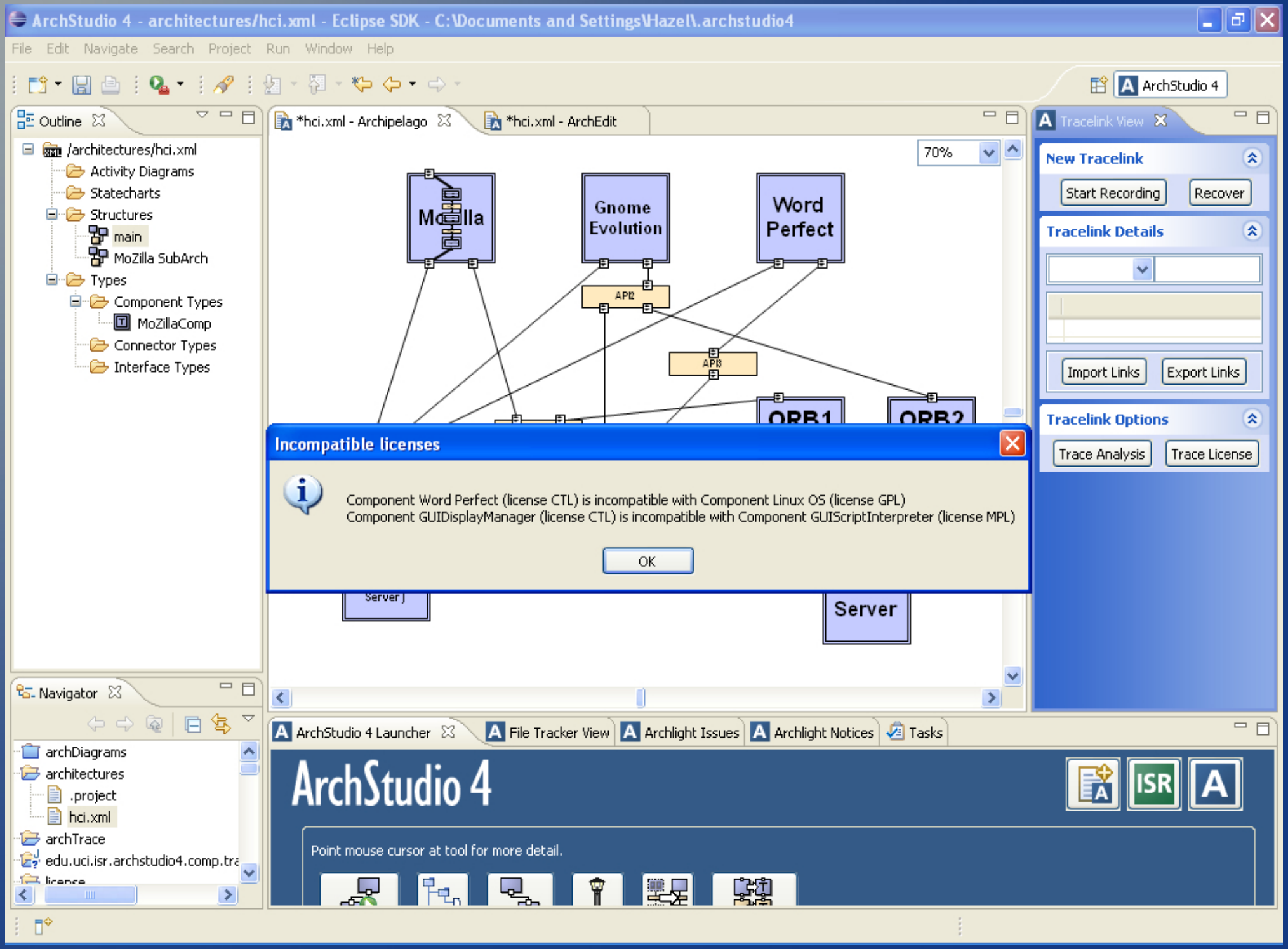
- License types:
  - Strongly reciprocal, weakly reciprocal, academic, Terms of Service, Proprietary
- Propagation of reciprocal obligations
- Calculating obligations and rights
- Detect conflicting obligations and missing rights

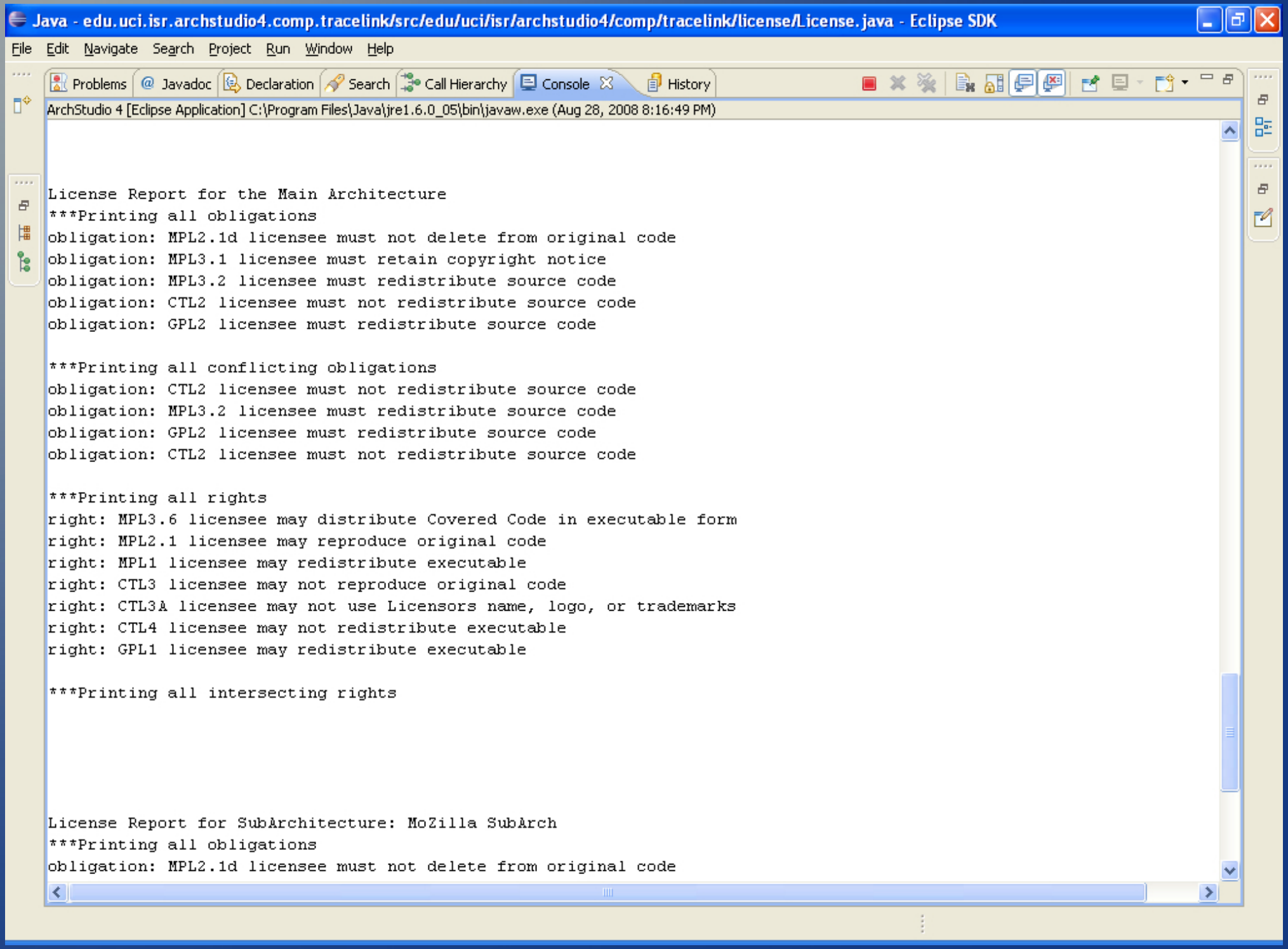
Reasoning structure during analysis




ArchStudio 4 - architectures/hci.xml - Eclipse SDK - C:\Documents and Settings\Hazel\.archstudio4
File Edit Navigate Search Project Run Window Help
ArchStudio 4
Outline
/architectures/hci.xml
Activity Diagrams
Statecharts
Structures
main
MoZilla SubArch
Types
Component Types
MoZillaComp
Connector Types
Interface Types
*hci.xml - Archipelago
70%
Mozilla
Gnome Evolution
Word Perfect
API2
API3
API1
ORB1
ORB2
CSH/Application Scripting
API4
Linux OS
Network Protocol (TCP/IP, IIOP, HTTP...)
Local Server (File Server, Printer Server)
Remote Server
Tracelink View
New Tracelink
Start Recording
Recover
Tracelink Details
Import Links
Export Links
Tracelink Options
Trace Analysis
Trace License
Navigator
archDiagrams
architectures
.project
hci.xml
archTrace
edu.uci.isr.archstudio4.comp
ArchStudio 4 Launcher
File Tracker View
Archlight Issues
Archlight Notices
Tasks
ArchStudio 4
ISR
Point mouse cursor at tool for more detail.


ArchStudio 4 - architectures/hci.xml - Eclipse SDK - C:\Documents and Settings\Hazel\.archstudio4
File Edit Navigate Search Project Run Window Help
ArchStudio 4
Outline
XArch
ArchAnalysis
ArchStructure [main]
Component [Gnome Evolution]
Description
Interface [GnomeToCSH Interface]
Interface [GnomeToLinuxInterface]
License
Licname
Component [Linux OS]
Component [Local Server (File Server, P
Component [Mozilla]
Component [ORB1]
Component [ORB2]
Component [Remote Server]
Component [Word Perfect]
Connector [API1]
Connector [API2]
Connector [API3]
Connector [API4]
Connector [CSH/Application Scripting]
Connector [Network Protocol (TCP/IP, I
Description
*hci.xml - Archipelago
*hci.xml - ArchEdit
ArchEdit
Syntax-Directed Architecture Editor
Licname: Description
Name
Value
value
GPL
Tracelink View
New Tracelink
Start Recording
Recover
Tracelink Details
Import Links
Export Links
Tracelink Options
Trace Analysis
Trace License
Navigator
archDiagrams
architectures
.project
hci.xml
archTrace
edu.uci.isr.archstudio4.comp.tracelink
ArchStudio 4 Launcher
File Tracker View
Archlight Issues
Archlight Notices
Tasks
ArchStudio 4
ISR
Point mouse cursor at tool for more detail.
AIM Launcher
ArchEdit
Archipelago
Archlight
Selector
Type Wrangler


ArchStudio 4 - architectures/hci.xml - Eclipse SDK - C:\Documents and Settings\Hazel\.archstudio4
File Edit Navigate Search Project Run Window Help
ArchStudio 4
Outline
/architectures/hci.xml
Activity Diagrams
Statecharts
Structures
main
MoZilla SubArch
Types
Component Types
MoZillaComp
Connector Types
Interface Types
*hci.xml - Archipelago
*hci.xml - ArchEdit
70%
Mozilla
Gnome Evolution
Word Perfect
API2
API3
ORB1
ORB2
Server
Incompatible licenses
Component Word Perfect (license CTL) is incompatible with Component Linux OS (license GPL)
Component GUIDisplayManager (license CTL) is incompatible with Component GUIScriptInterpreter (license MPL)
OK
Tracelink View
New Tracelink
Start Recording
Recover
Tracelink Details
Import Links
Export Links
Tracelink Options
Trace Analysis
Trace License
Navigator
archDiagrams
architectures
.project
hci.xml
archTrace
edu.uci.isr.archstudio4.comp.tra
ArchStudio 4 Launcher
File Tracker View
Archlight Issues
Archlight Notices
Tasks
ArchStudio 4
ISR
Point mouse cursor at tool for more detail.


Java - edu.uci.isr.archstudio4.comp.tracelink/src/edu/uci/isr/archstudio4/comp/tracelink/license/License.java - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Problems Javadoc Declaration Search Call Hierarchy Console History
ArchStudio 4 [Eclipse Application] C:\Program Files\Java\jre1.6.0_05\bin\javaw.exe (Aug 28, 2008 8:16:49 PM)
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: CTL2 licensee must not redistribute source code
obligation: GPL2 licensee must redistribute source code
***Printing all conflicting obligations
obligation: CTL2 licensee must not redistribute source code
obligation: MPL3.2 licensee must redistribute source code
obligation: GPL2 licensee must redistribute source code
obligation: CTL2 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: CTL3 licensee may not reproduce original code
right: CTL3A licensee may not use Licensors name, logo, or trademarks
right: CTL4 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

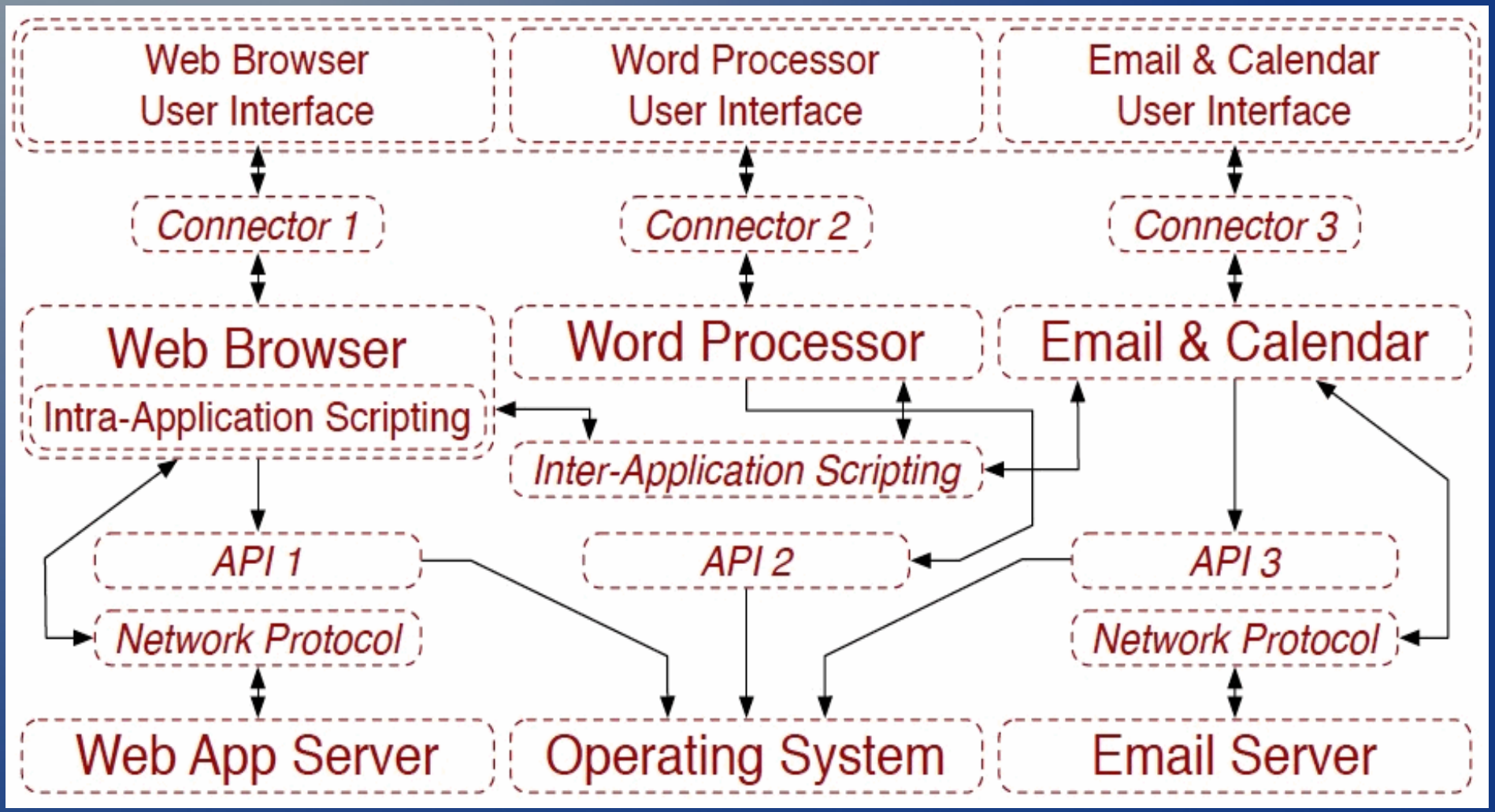
- The obligation for a user to verify his/her authority to see compartment T, by password or other specified authentication process
- The obligation for all components connected to specified component C to grant it the capability to read and update data in compartment T
- The obligation to reconfigure 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?

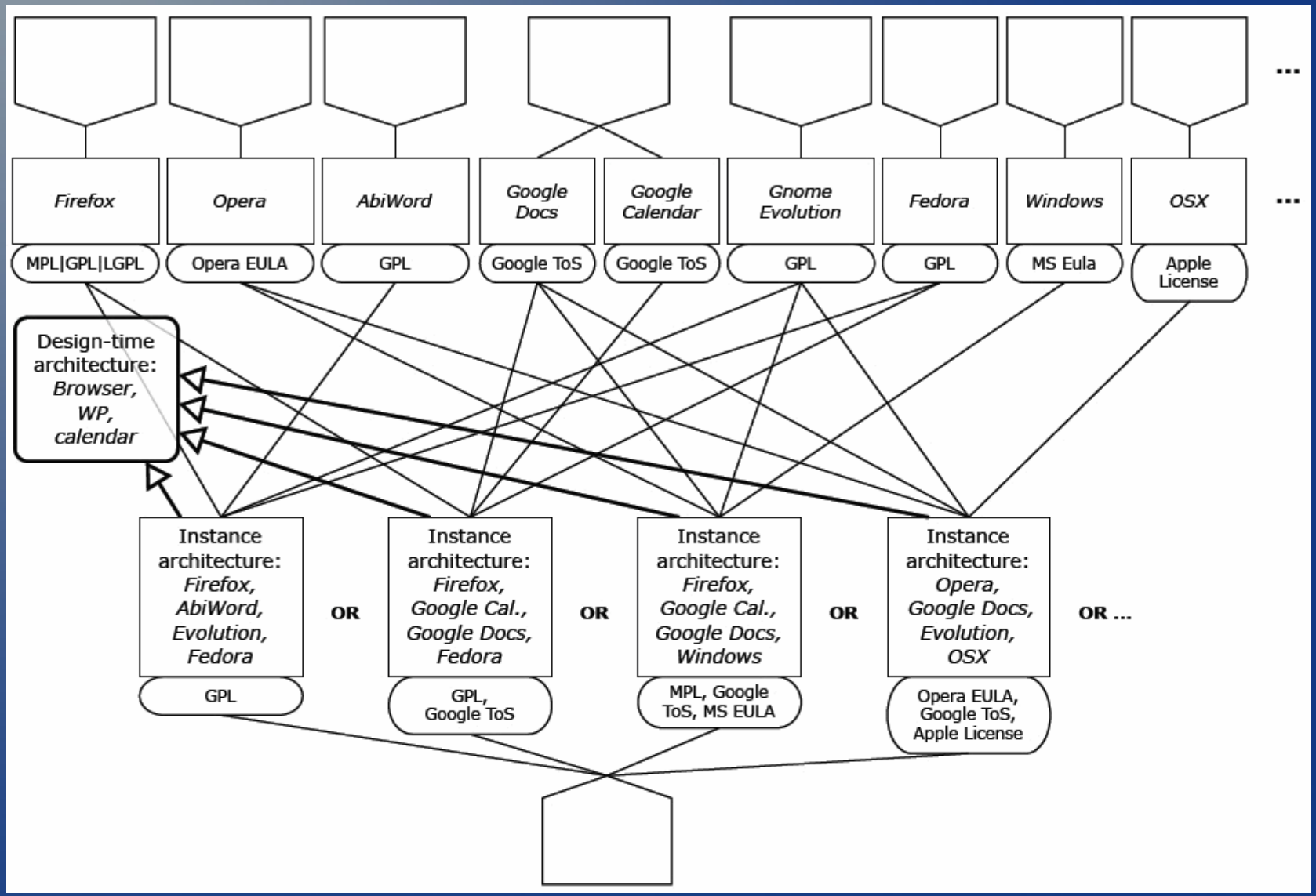
- When functionally similar software components, connectors, or configurations exist,
- Such that equivalent alternatives, versions, or variants may be substituted for one another, then
- We have a strong relationship among these OA system elements that is called a *software product line*.
- Software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

- Reed, H., Benito, P., Collens, J. and Stein, F. (2012). Supporting Agile C2 with an Agile and Adaptive IT Ecosystem, *Proc. 17Th Intern. Command and Control Research and Technology Symposium* (ICCRTS), Paper-044, Fairfax, VA, June 2012.

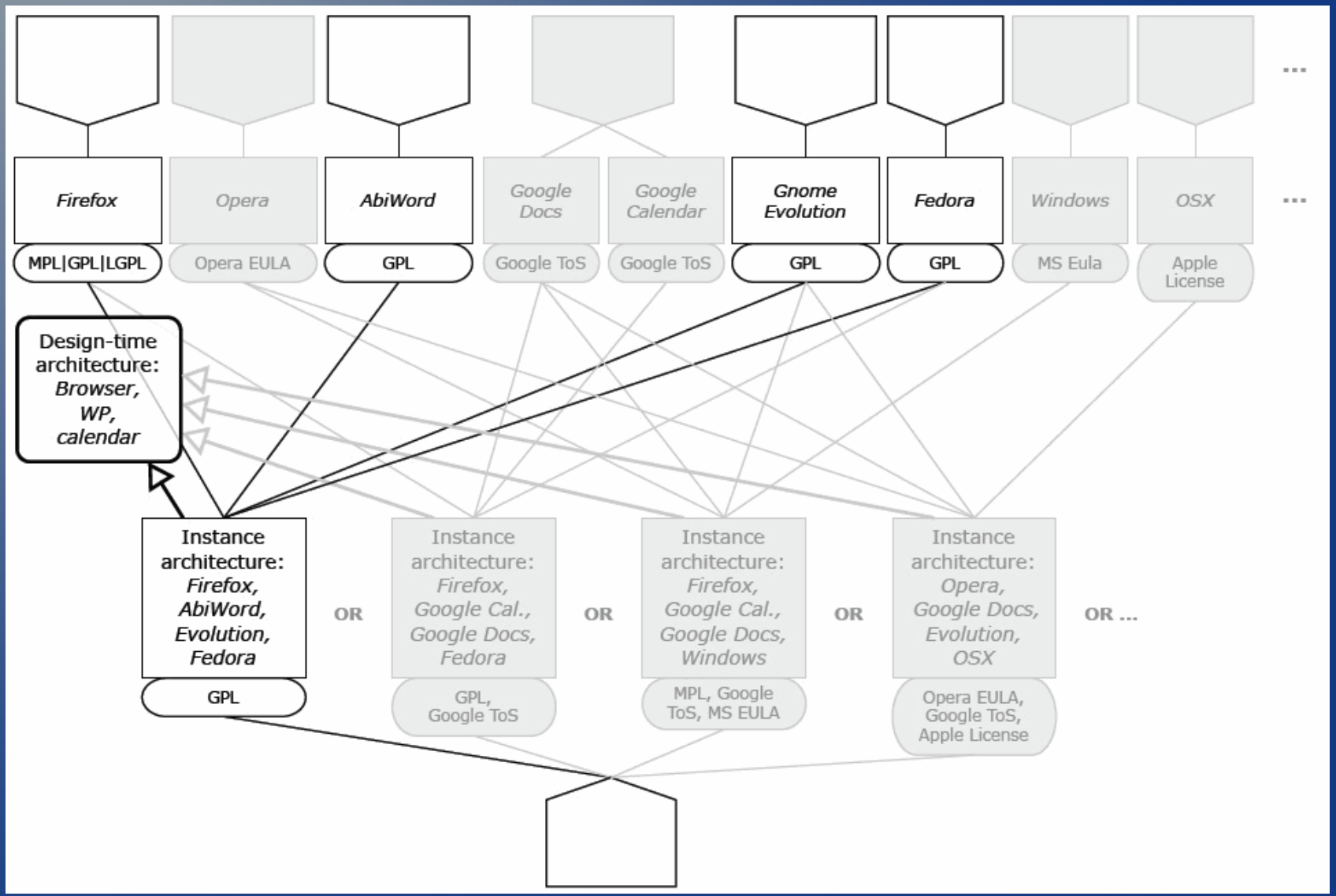
Design-time view of an OA system



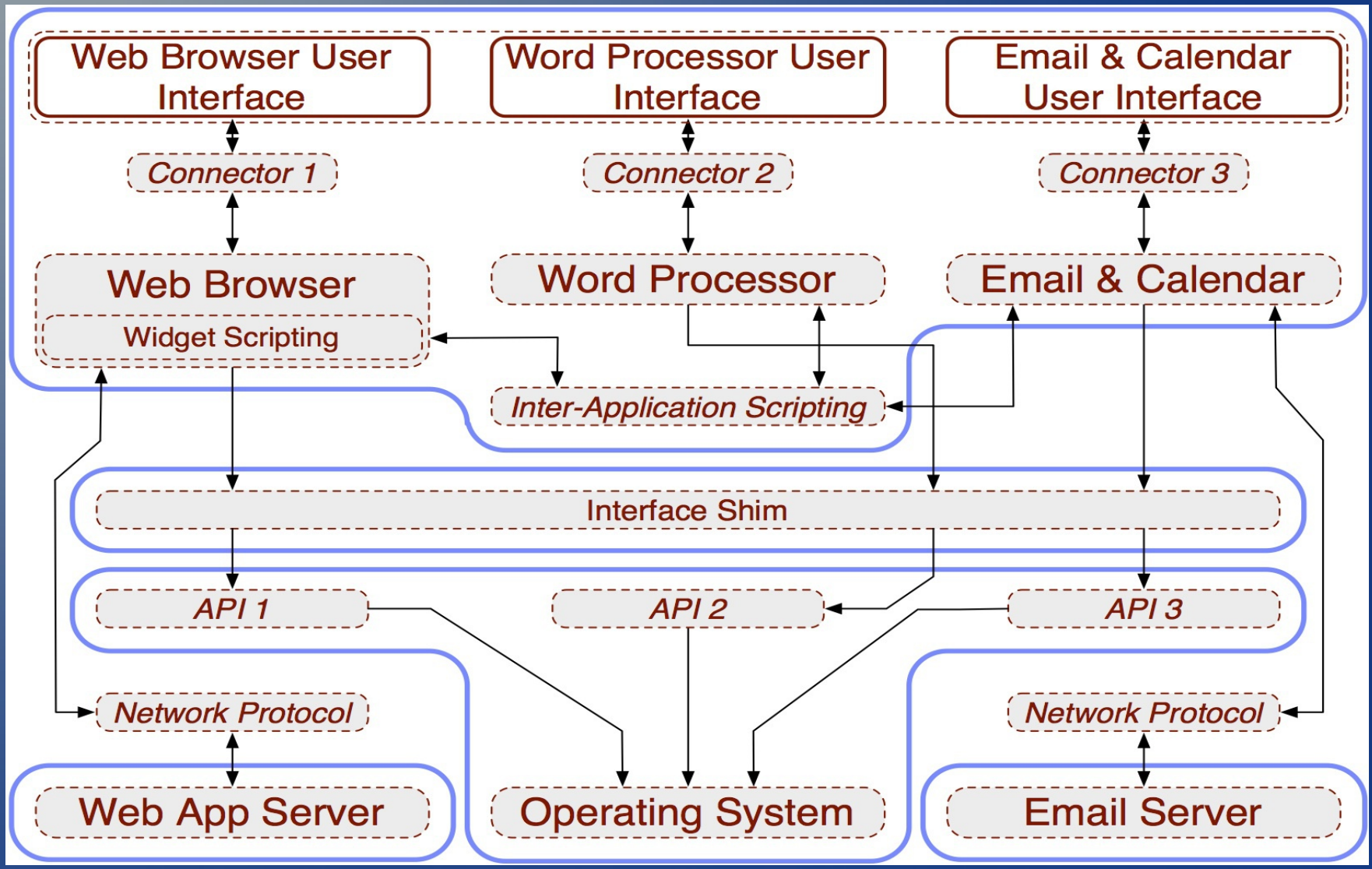
Software product line of *functionally similar* OA system alternatives



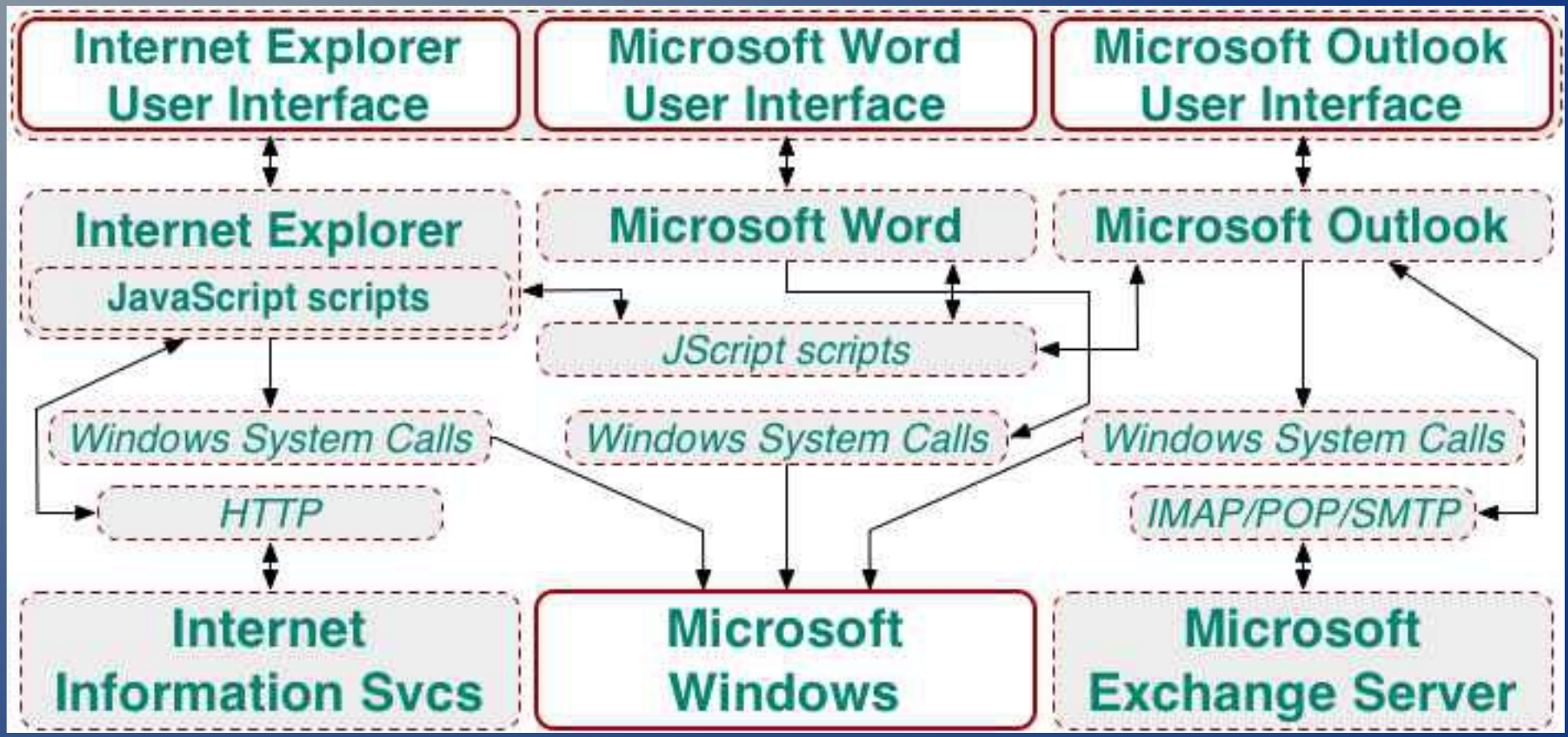
Product line selection of one alternative system configuration



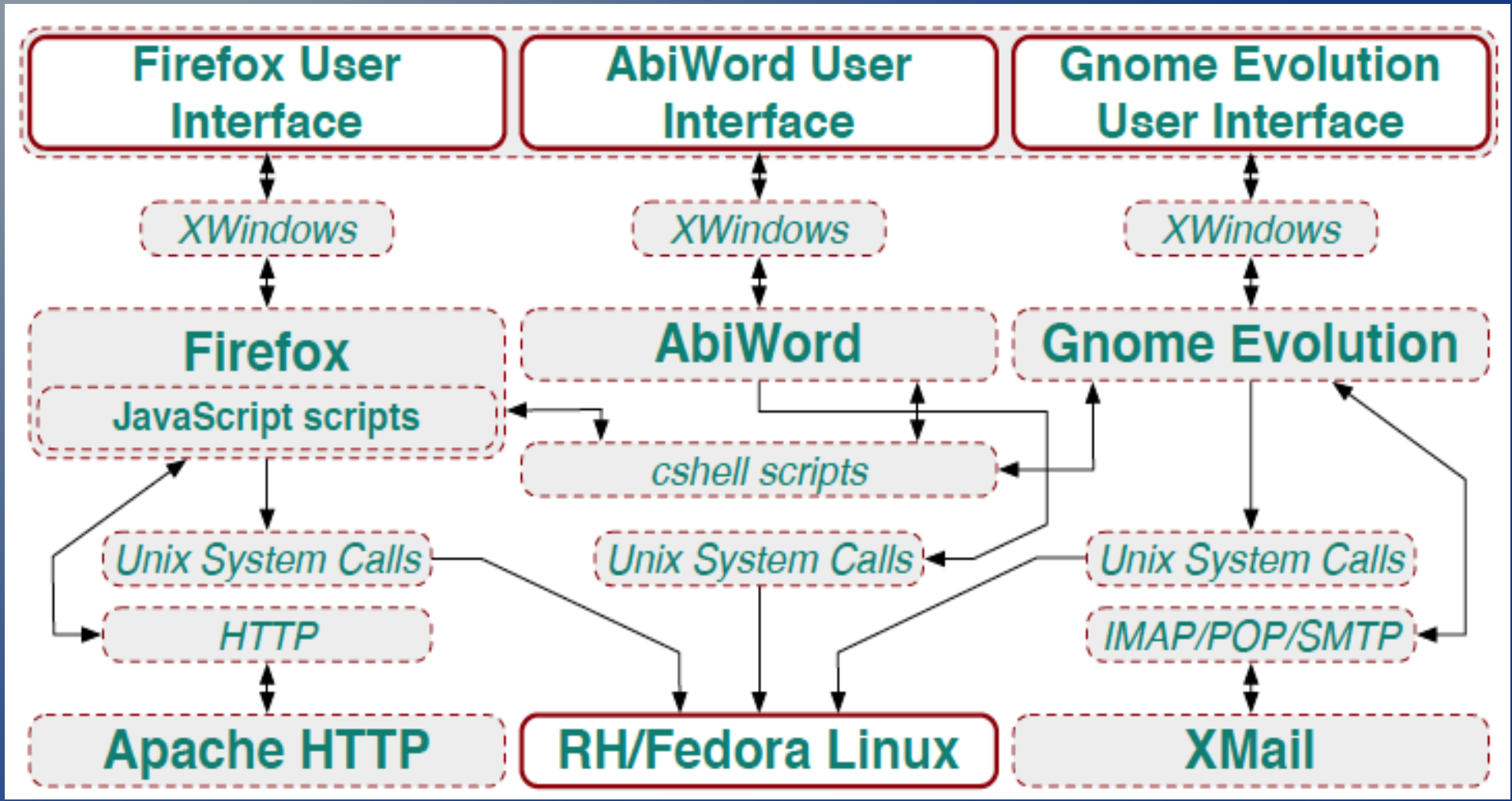
A security capability specification encapsulating the *design-time* configuration via multiple virtual machine containers



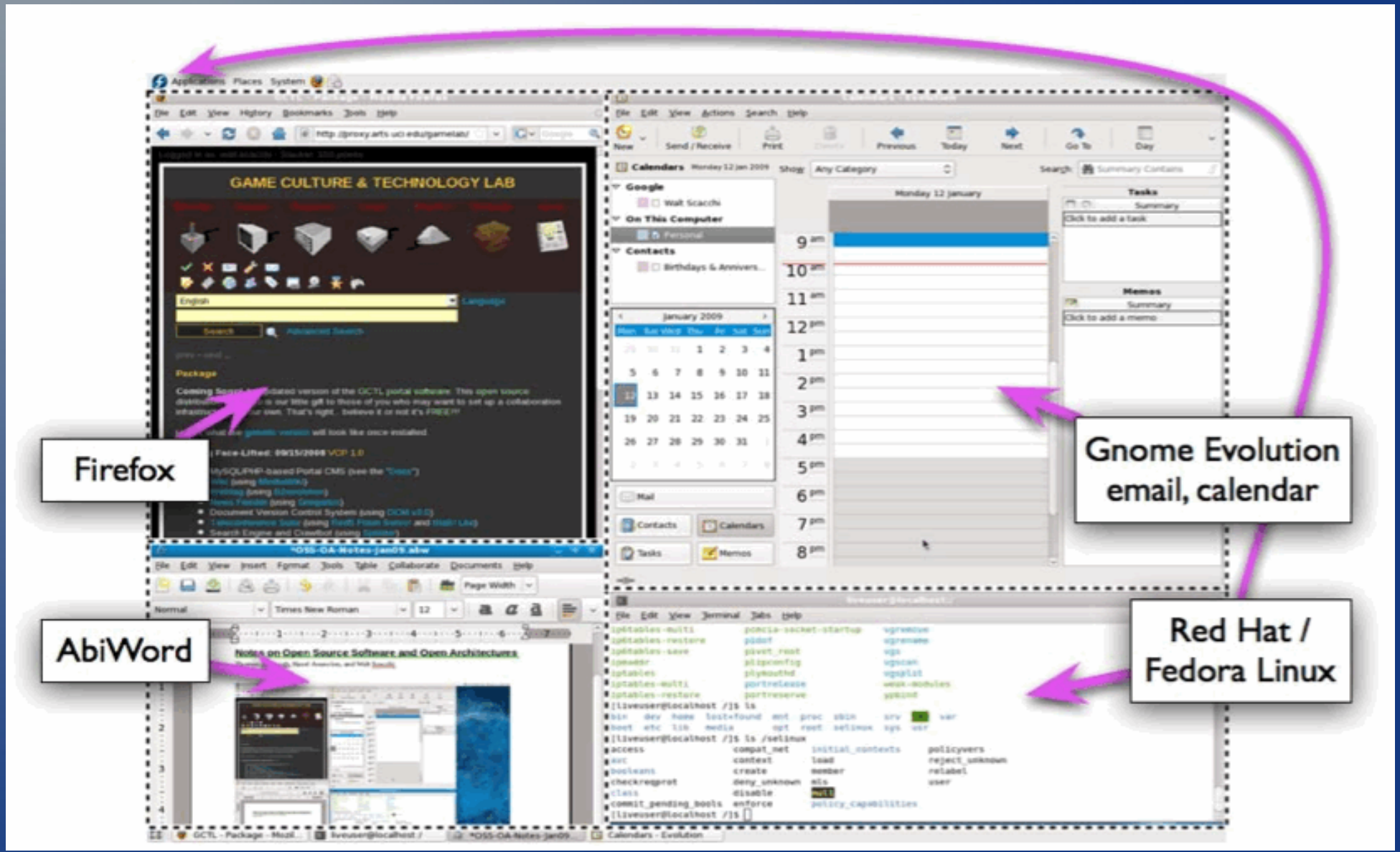
Build-time view of OA design selecting *proprietary* product family alternatives



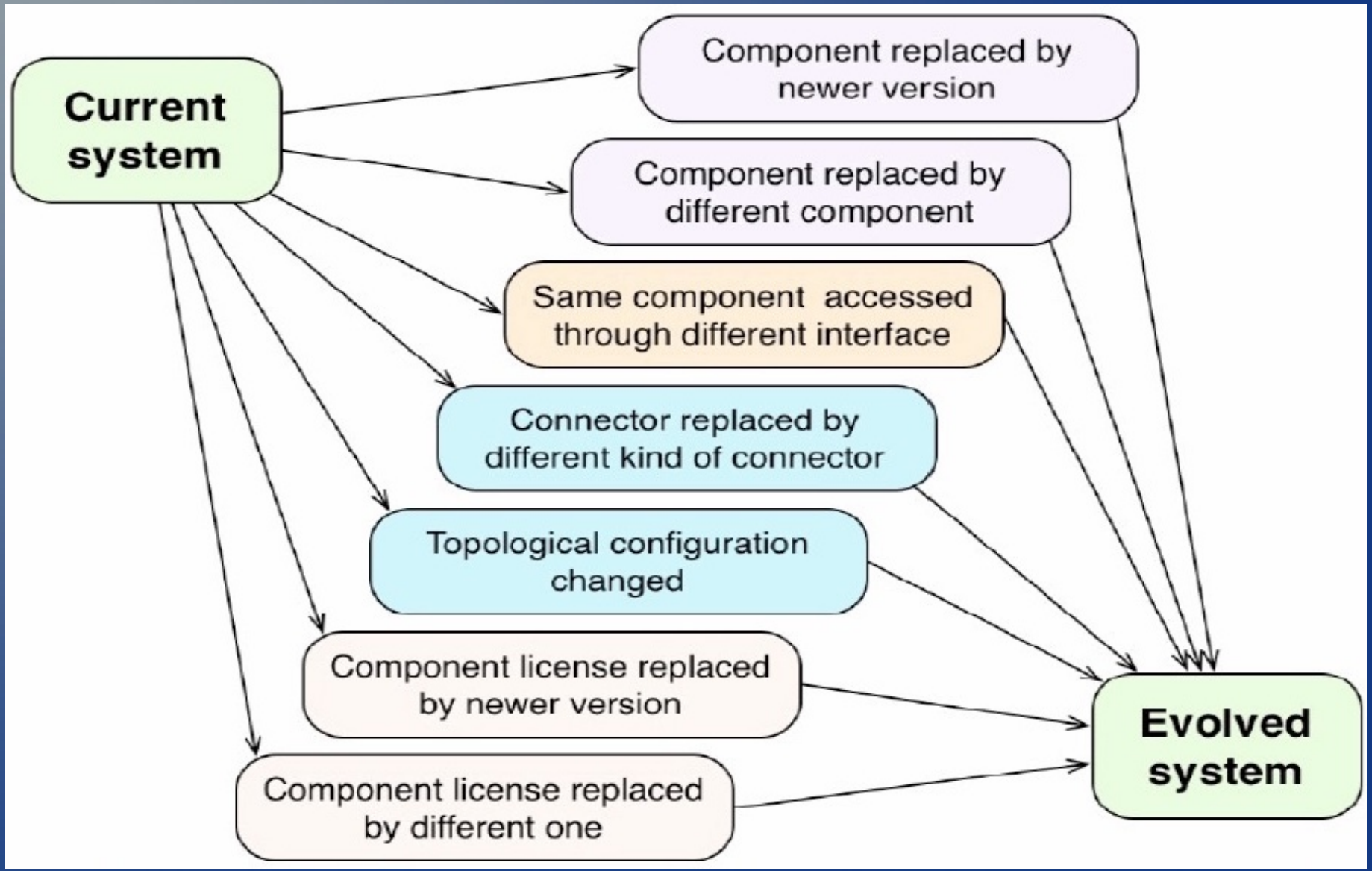
Build-time view of OA design selecting *OSS* product family alternatives



Run-time deployment view of OA system family member configuration



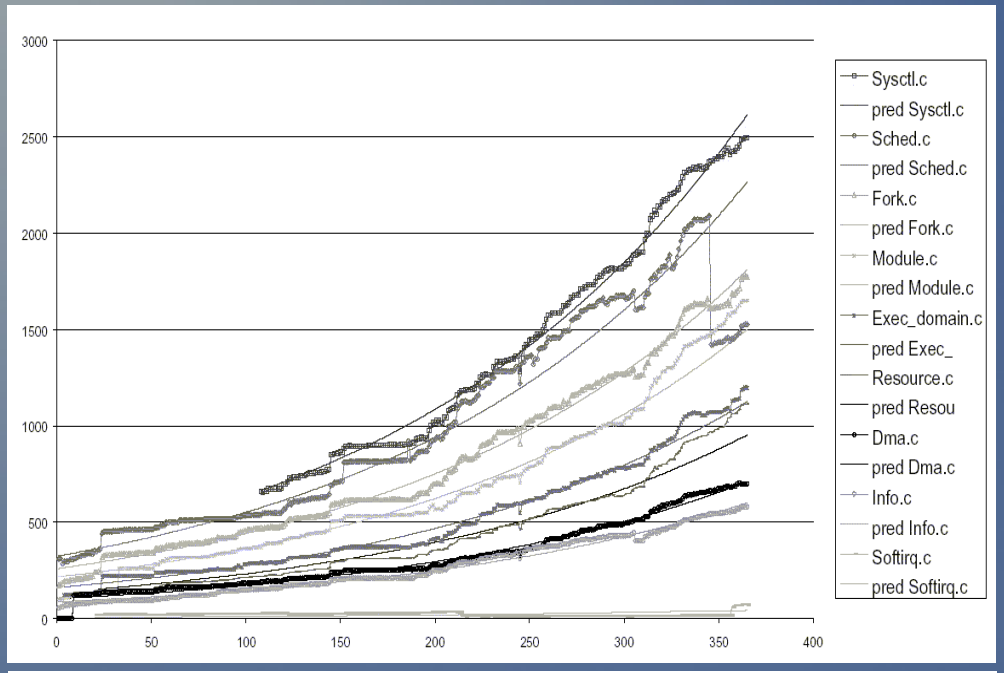
Evolution-time software changes

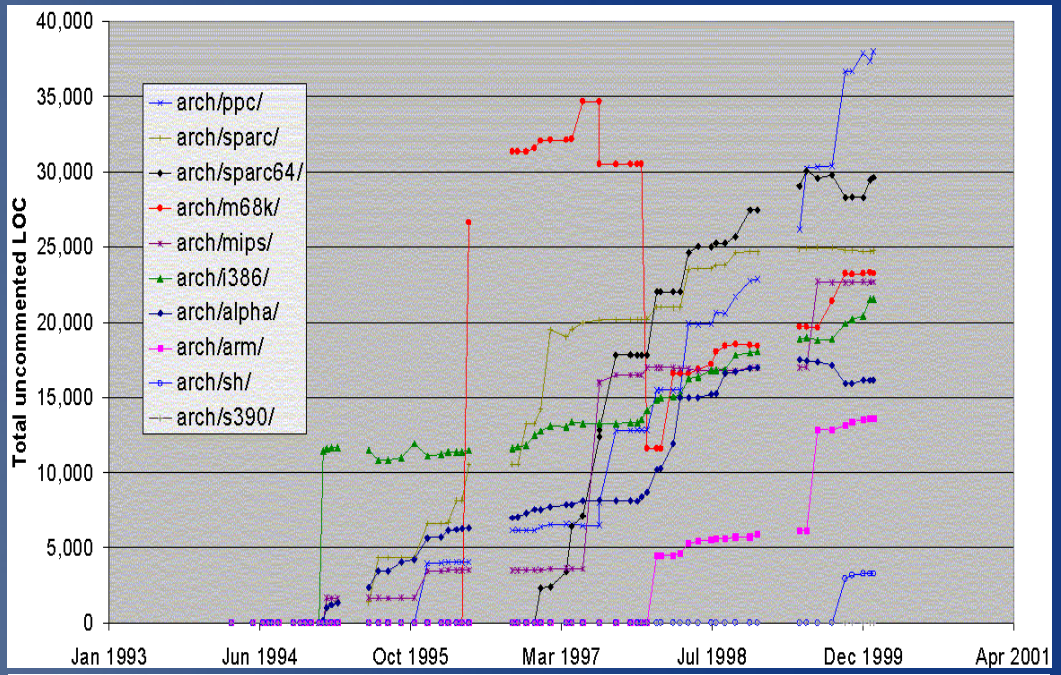


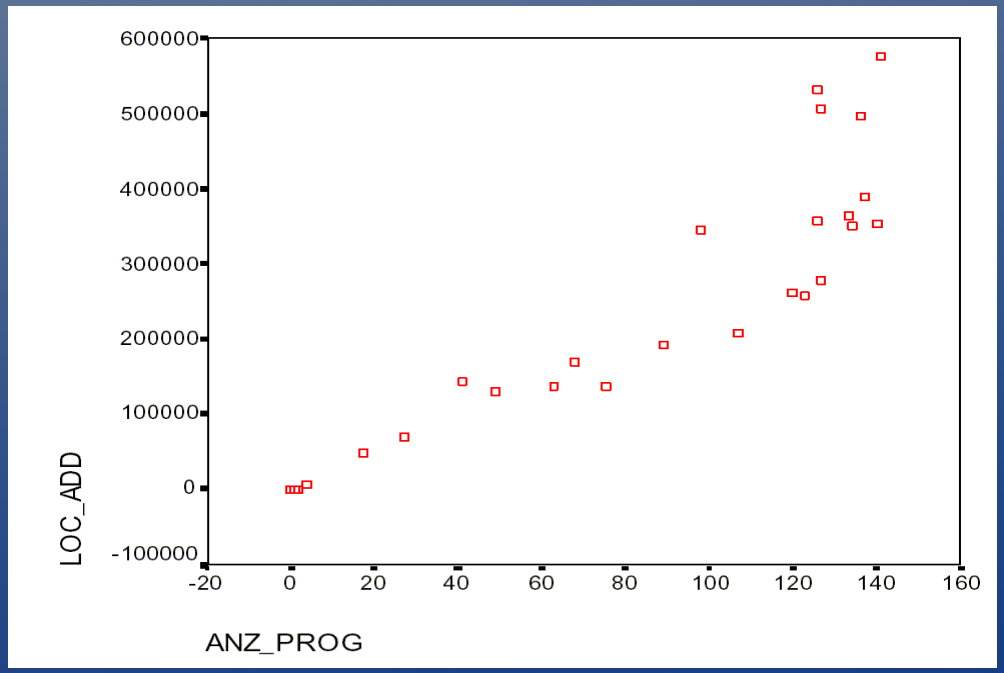
Evolutionary redevelopment, reinvention, and redistribution

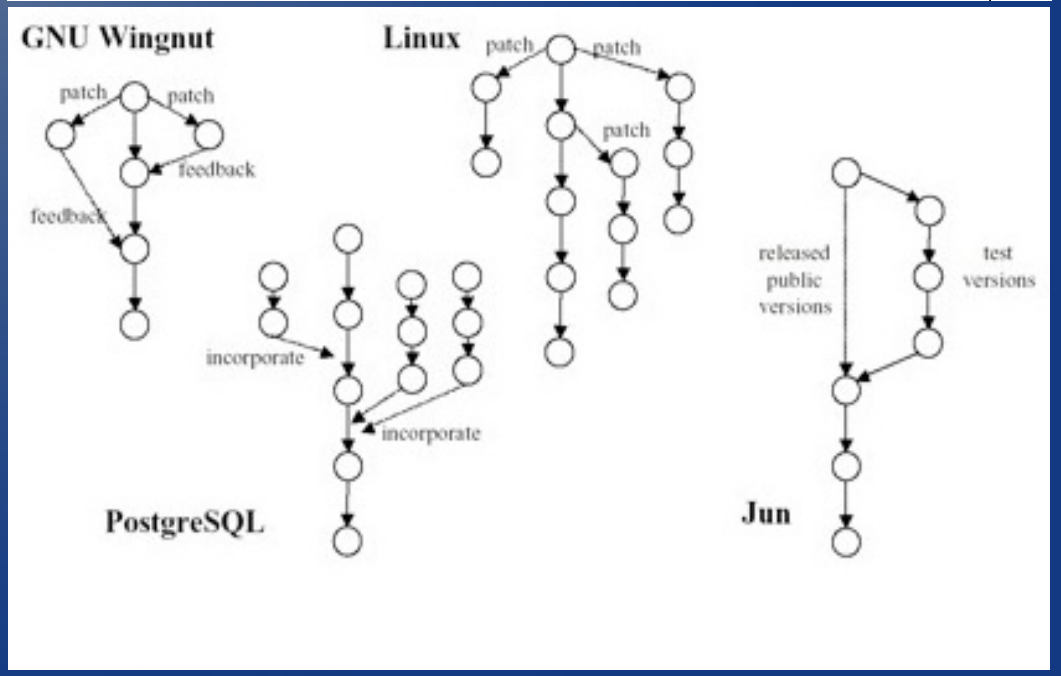
- Overall evolutionary dynamic of many OSSD projects is *reinvention and redevelopment*
  - Reinvention enables continuous improvement and collective learning
- OSS evolve through minor mutations
  - Expressed, recombined, redistributed via incremental releases
- OSS systems *co-evolve* with their development community
  - Success of one depends on the success of the other

OSS evolution trends

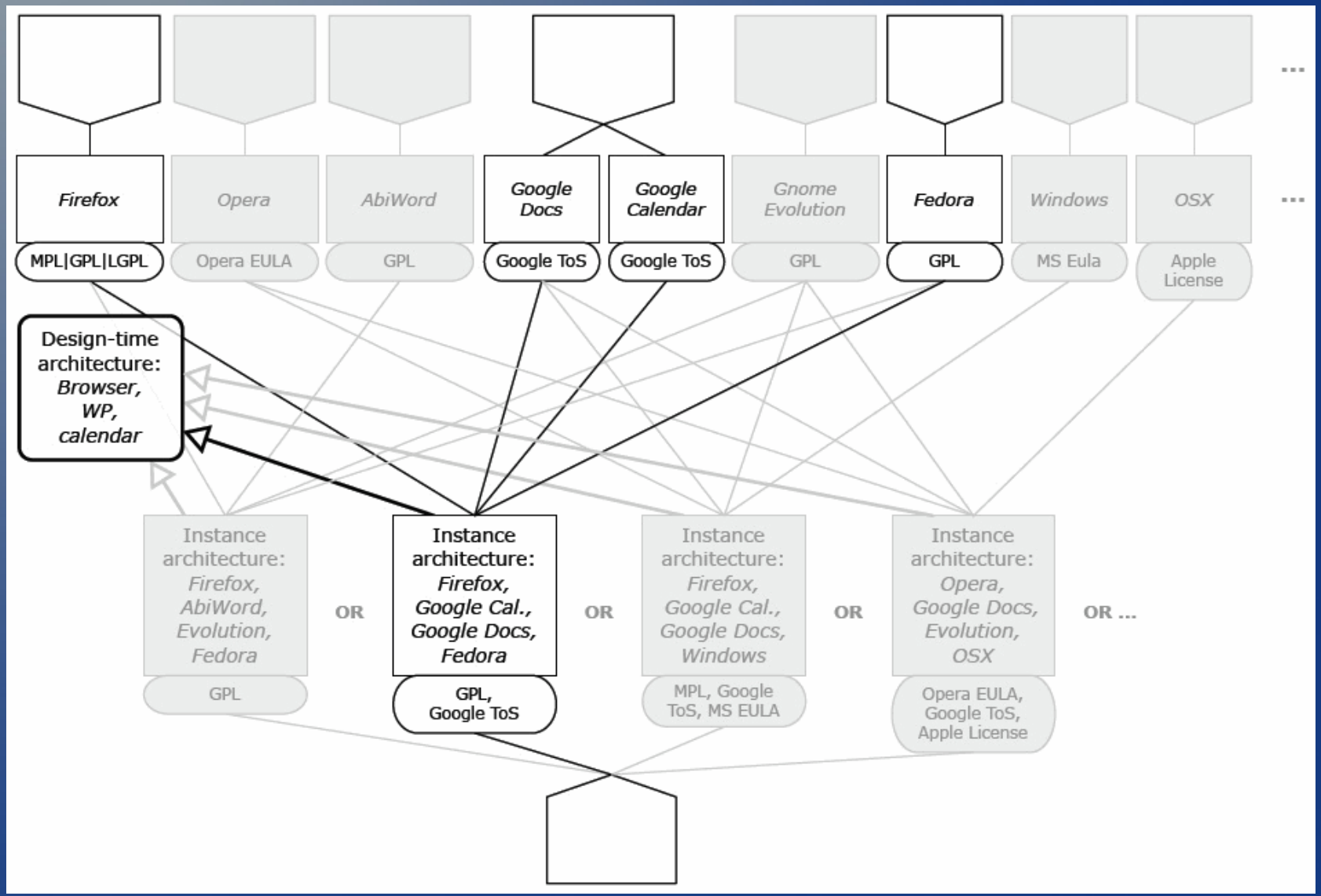




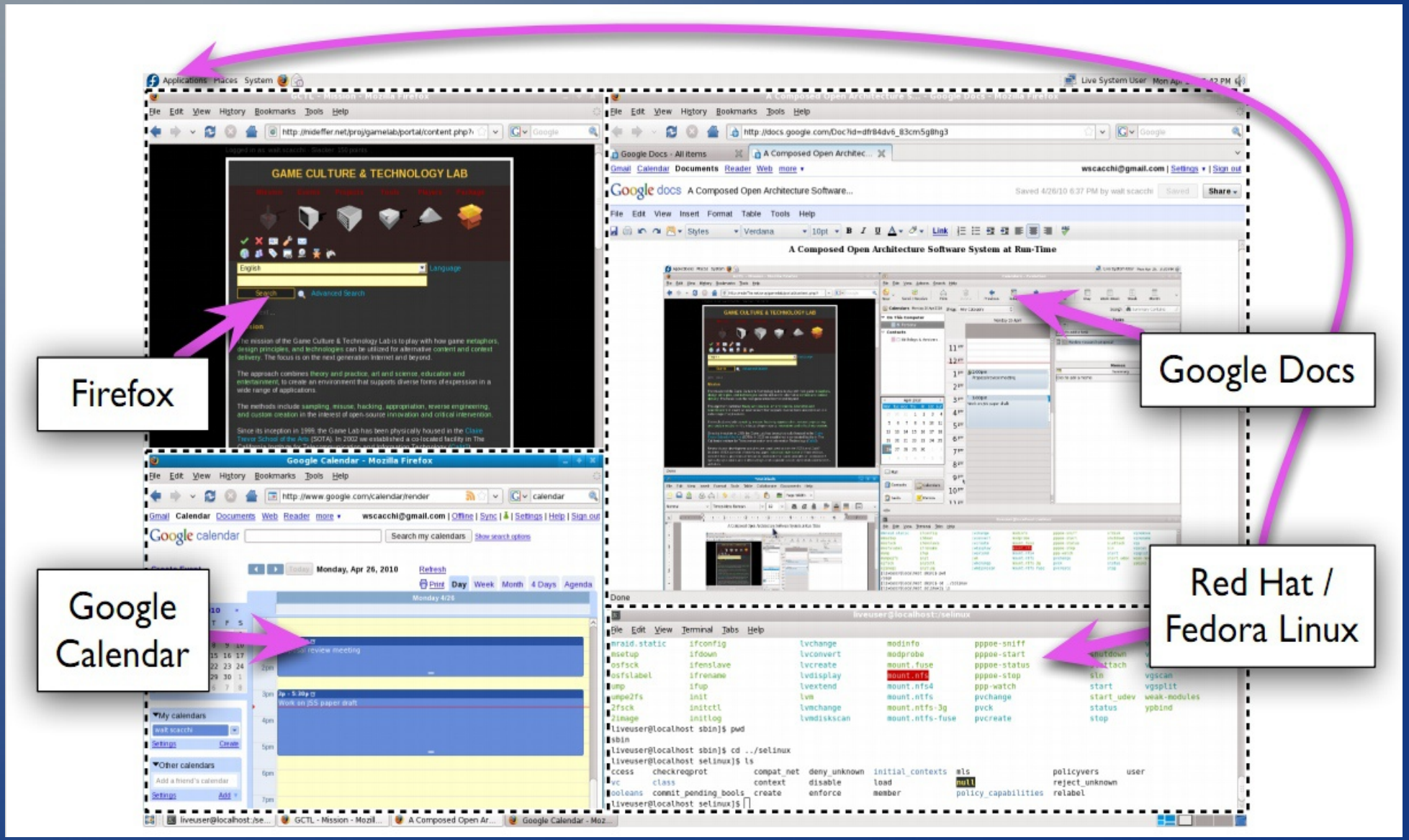




Product line selection of different functionally similar alternative

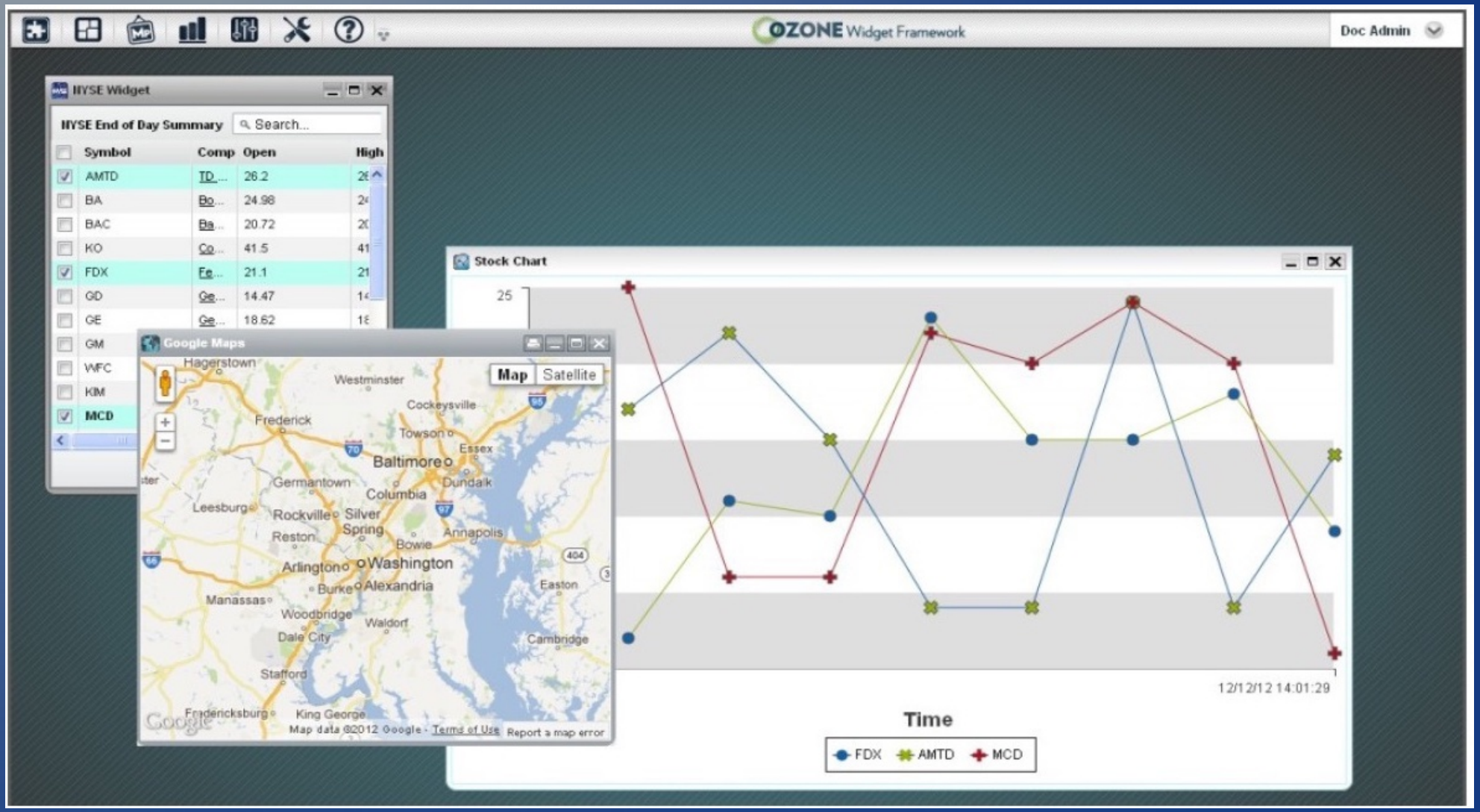


Evolved run-time deployment view of a *functionally similar* alternative OA system configuration

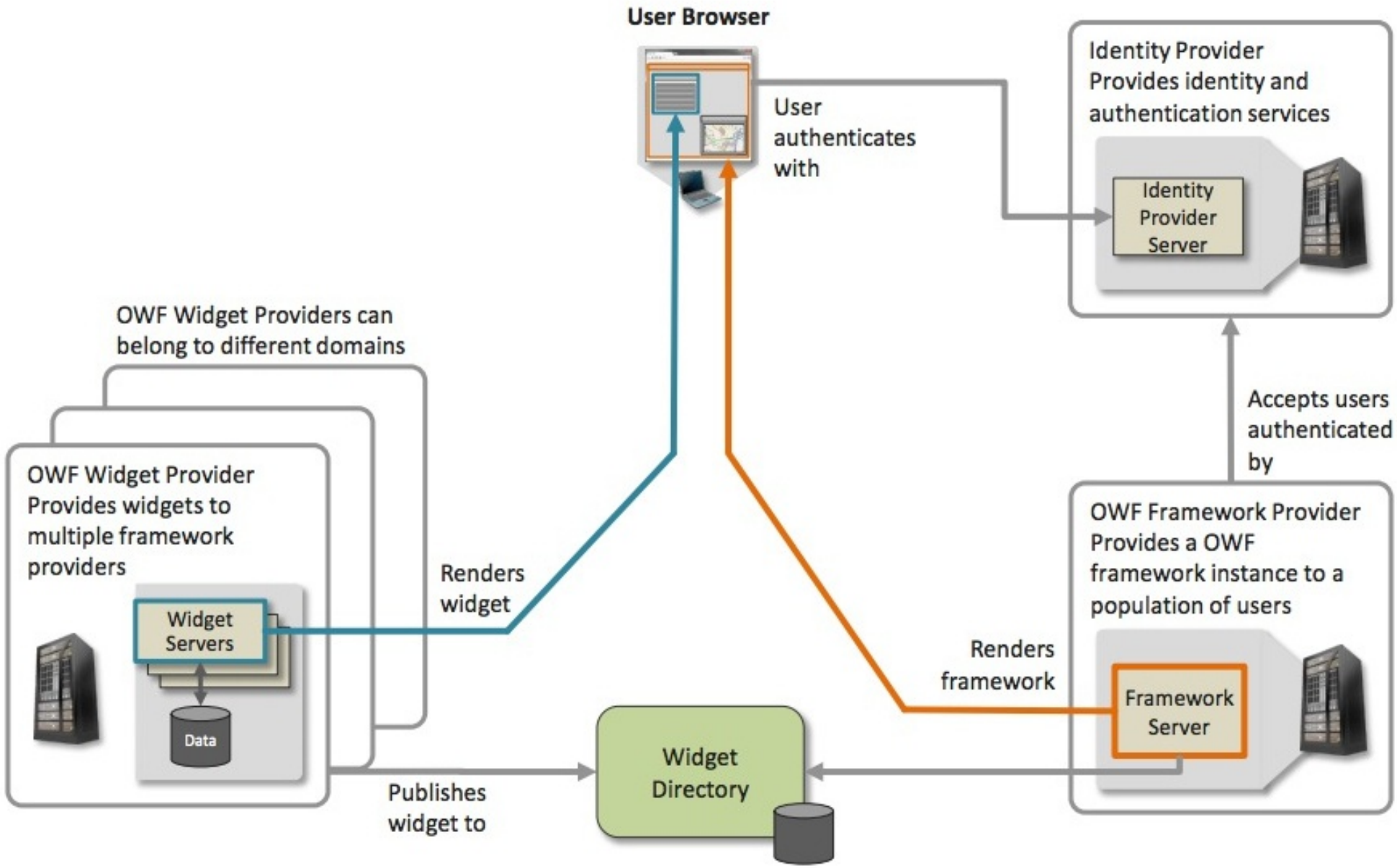


Emerging Transformations with OSS and OA systems

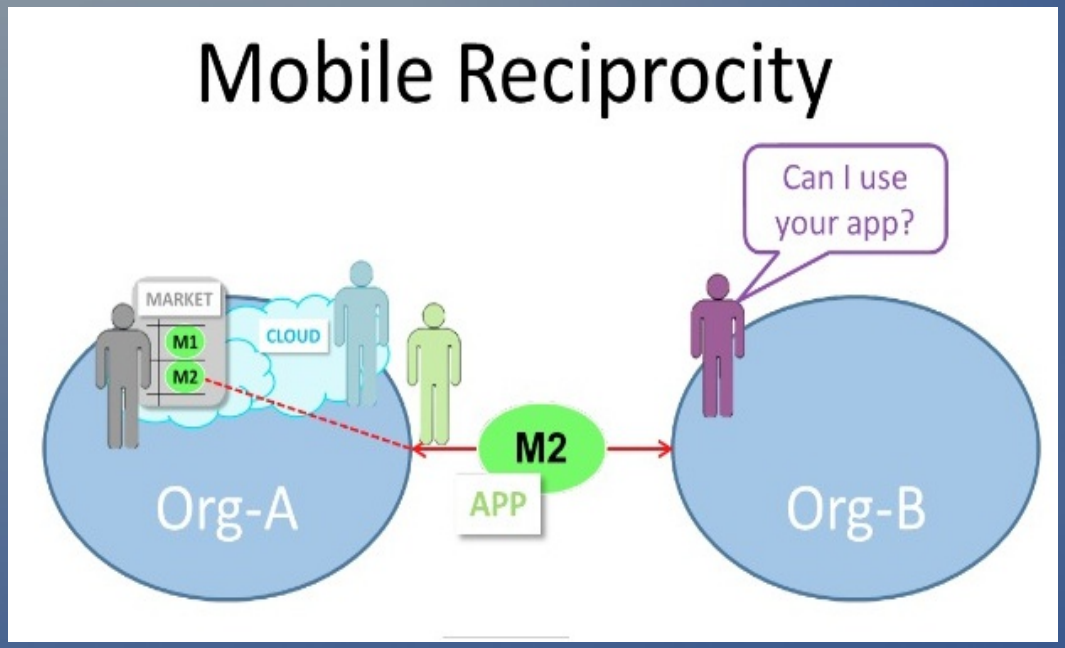
New paths for software platform development and evolution using intra-app components: (GOSS) *Widgets/apps* acquired from online App Store

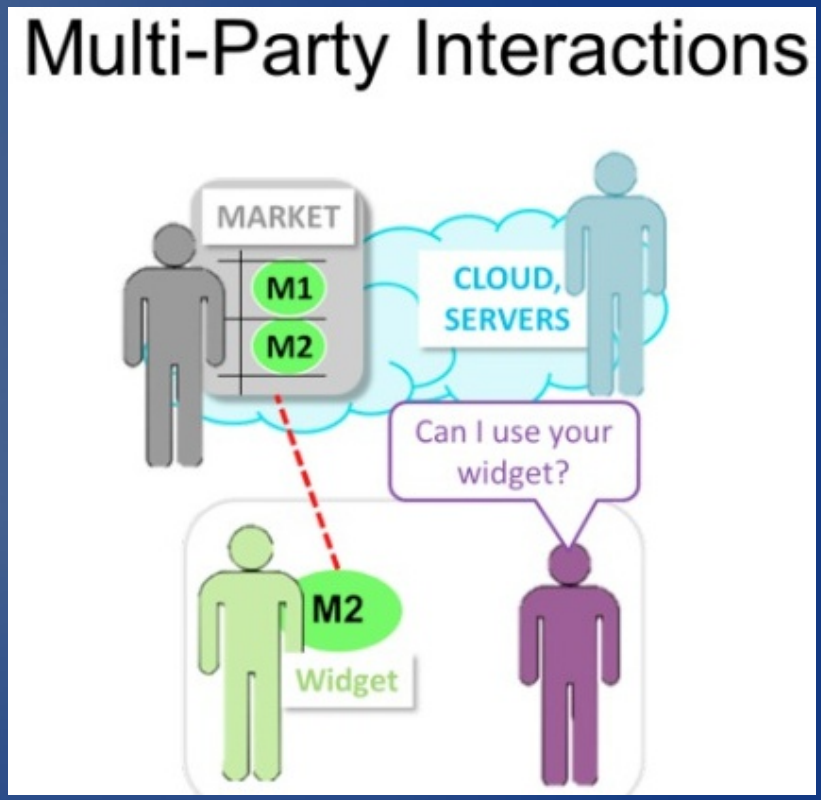


Ozone Widget Framework (OWF)



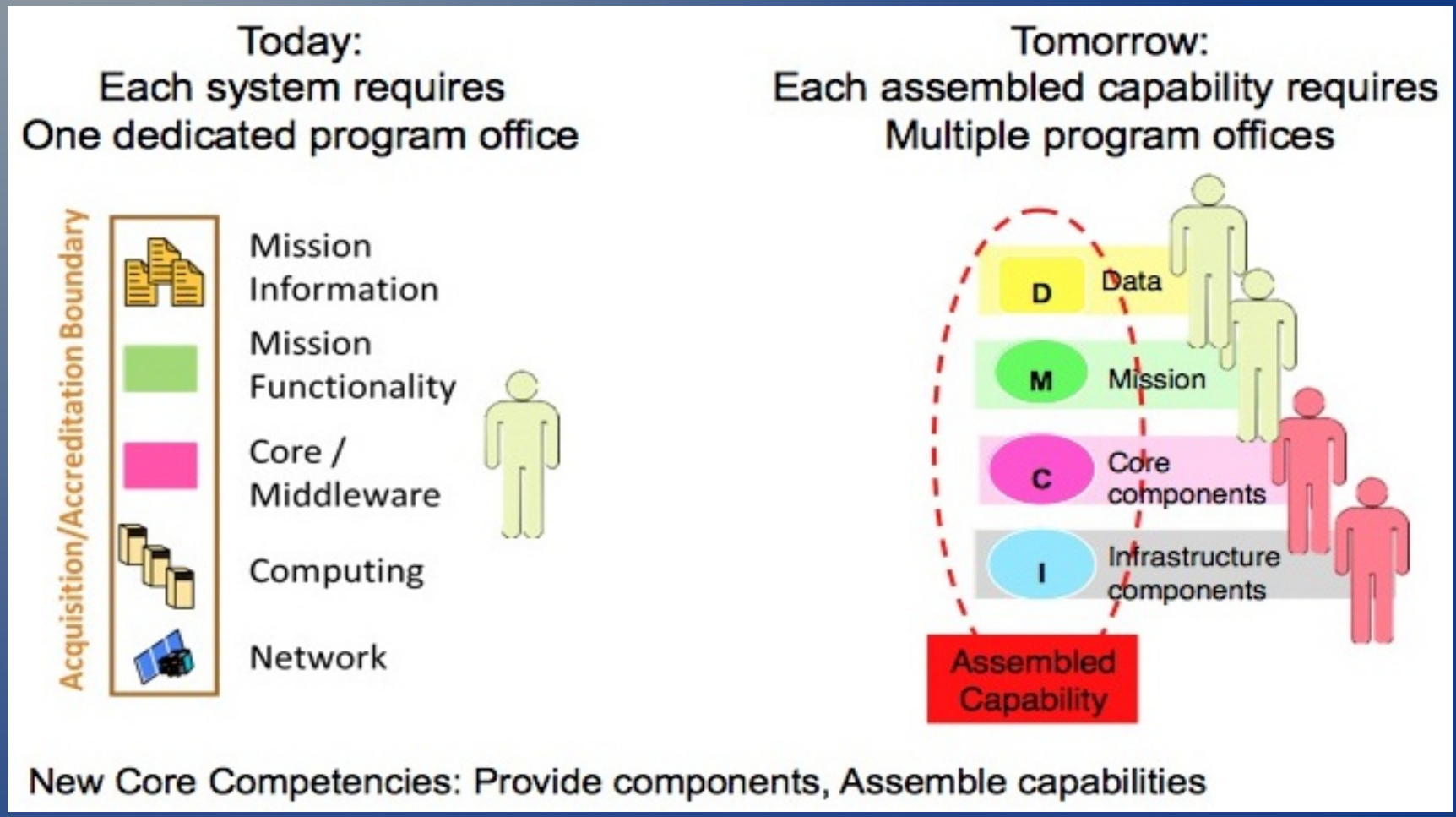
Transforming to multi-party acquisition of software elements within OA ecosystems





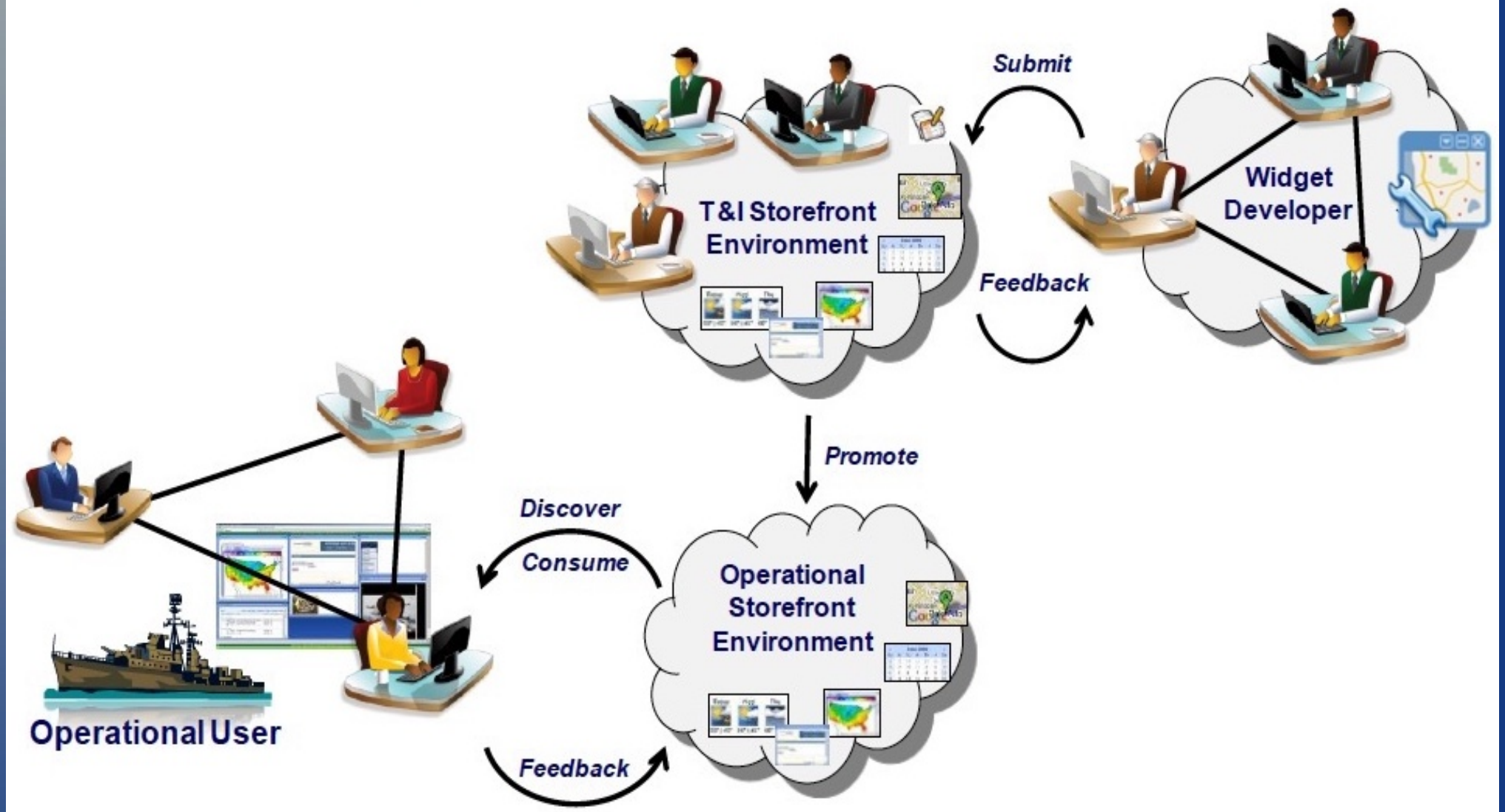
Customer/end-user organizations now looking for ways to reduce acquisition cost and effort through *shared development/use of common OA software system components* (apps, widgets).

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



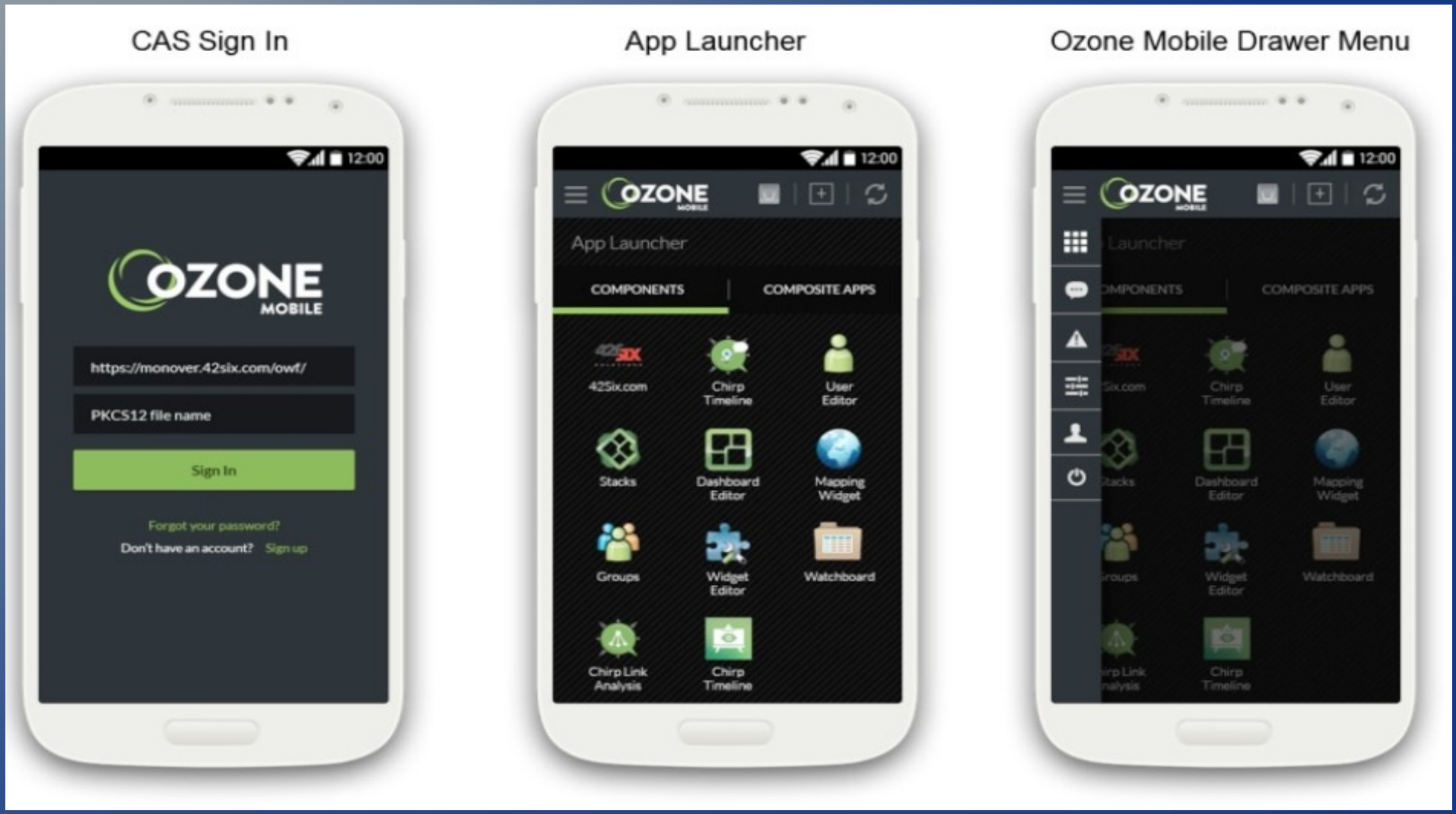
Customer/end-user organizations now looking for ways to reduce acquisition cost and effort through *shared development/use of assembled capabilities* for GOSS-based C3CB systems

Figure 1: PEO C4I Storefront Operational Concept



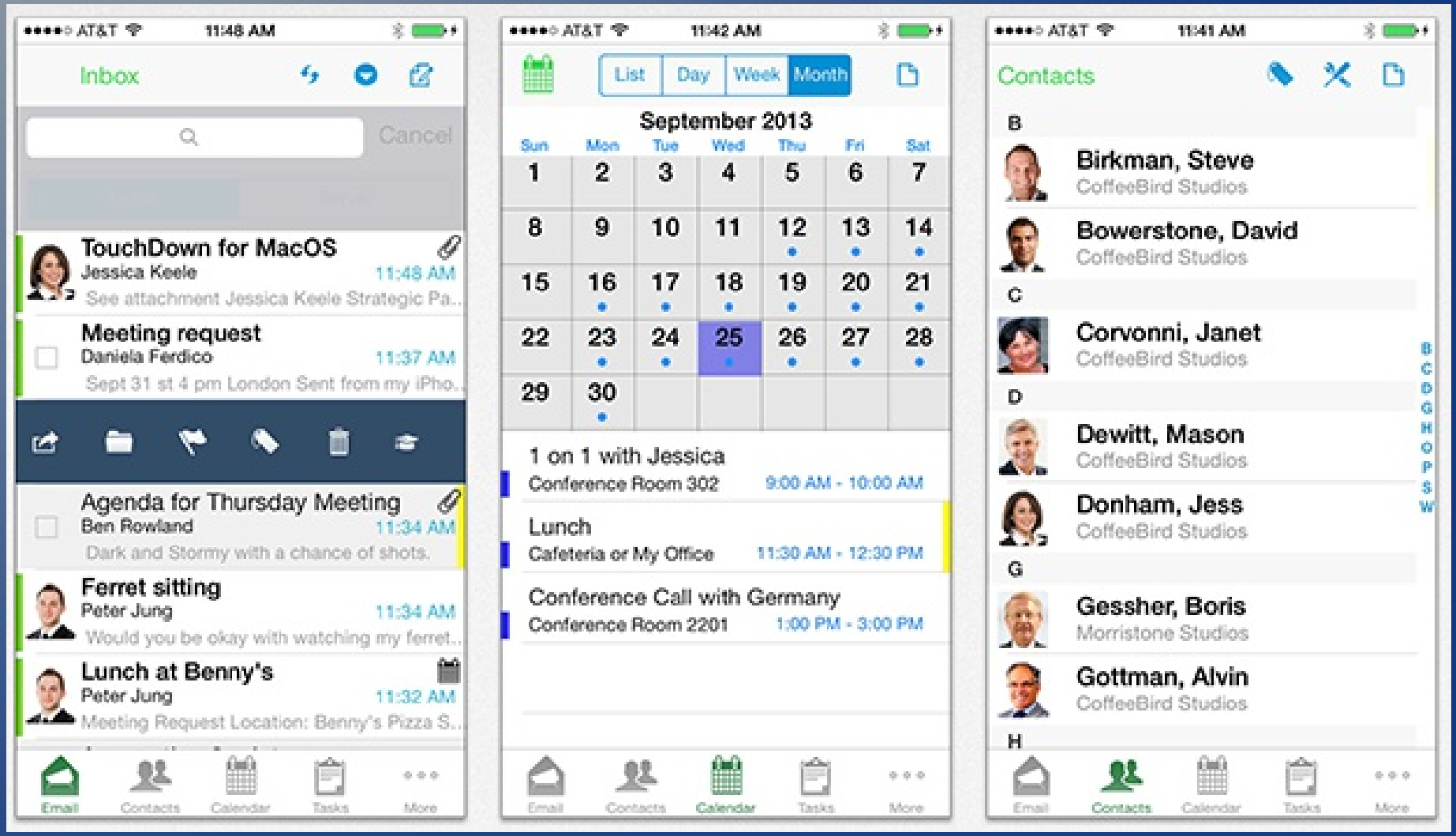
Source: George, A. Galdorisi, G, Morris, M. O"Neil, M. (2014). DoD Application Store: Enabling C2 Agility?, *Proc. 19th Intern. Command and Control Research & Technology Symposium*, Alexandria, VA.

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)



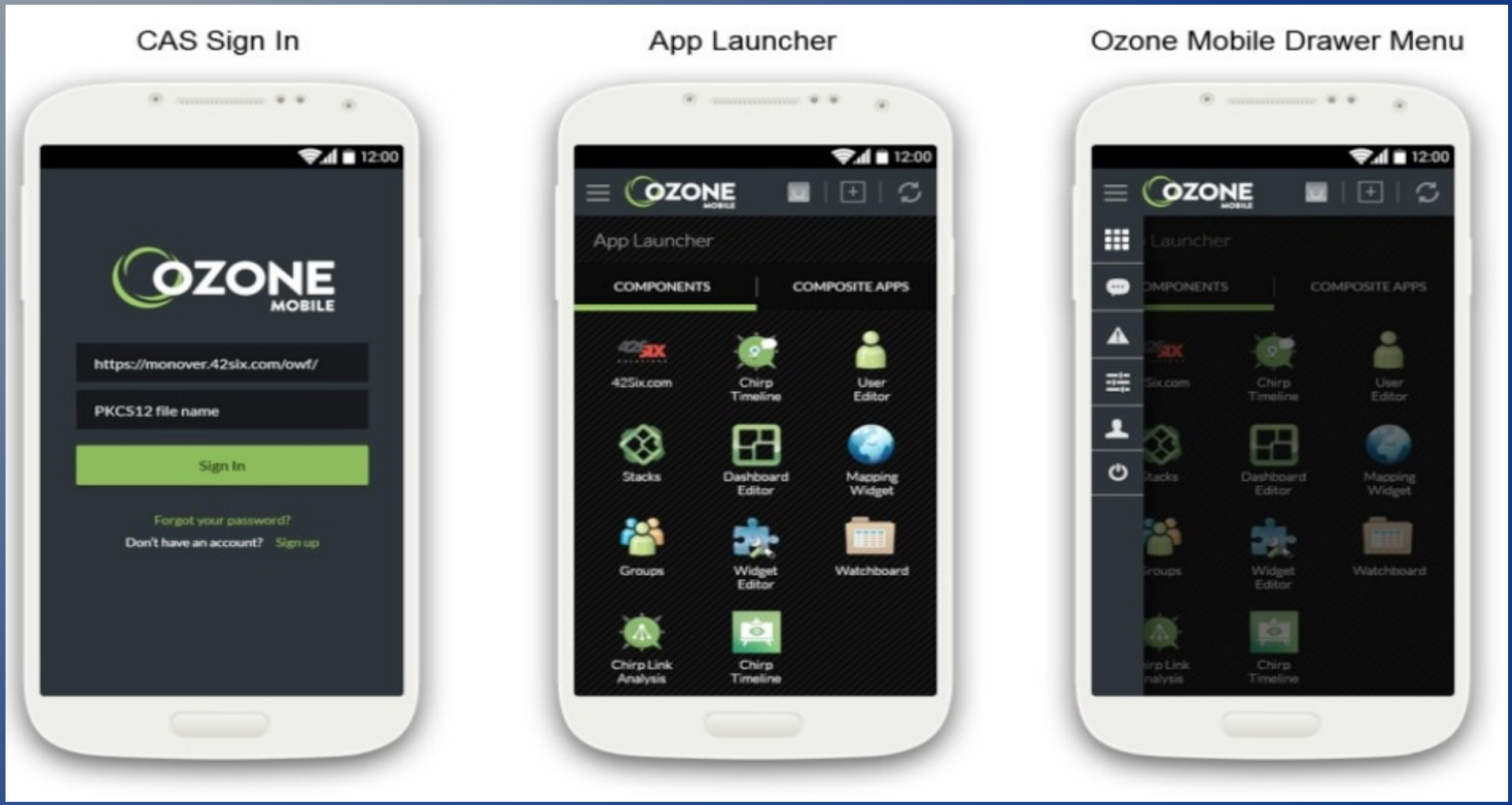
Enterprise-to-Mobile Middleware *IP Licenses*
(for the NitroDesk *Touchdown* product in 2014)

* 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

- Scacchi, W. and Alspaugh, T. (2012) Addressing Challenges in the Acquisition of Secure Software Systems with Open Architectures, *Proc. 9th Acquisition Research Symposium*, Vol. 1, 165-184, Naval Postgraduate School, Monterey, CA.
- Scacchi, W. and Alspaugh, T. (2013a). Processes in Securing Open Architecture Software Systems, *Proc. 2013 Intern. Conf. Software and System Processes*, San Francisco, CA, May 2013.
- Scacchi, W. and Alspaugh, T.A. (2013b). Streamlining the Process of Acquiring Secure Open Architecture Software Systems, *Proc. 10th Annual Acquisition Research Symposium*, Monterey, CA, 608-623, May 2013.
- Scacchi, W. and Alspaugh, T.A. (2013c). Challenges in the Development and Evolution of Secure Open Architecture Command and Control Systems, *Proc. 18th Intern. Command and Control Research and Technology Symposium*, Paper-098, Alexandria, VA, June 2013.

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
  - They 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!


INSTITUTE for SOFTWARE RESEARCH
UNIVERSITY of CALIFORNIA • IRVINE