

Addressing Challenges in the Acquisition of Secure Software Systems with Open Architectures

Walt Scacchi and Thomas Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3455 USA

Overview

- Challenges of securing open architecture (OA) systems
- Specifying security requirements for software systems
- Case study: Specifying secure software product lines within an OA software ecosystem for enterprise systems
- Discussion and conclusions

Challenges of securing open architecture (OA) systems

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, via portable storage devices like USB drives, modified end-user devices, and social engineering techniques.
- What makes an OA system secure changes over time, as new threats emerge and systems evolve.
- Need an approach *to continuously assure the security of evolving OA systems* that is practical, scalable, robust, tractable, and adaptable.

Software 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 for configuring the security parameters for applications and operating systems;
- Functionally equivalent but diverse multi-variant software executables.

What acquisition research is needed for security?

- How to *verify* the security of OA system designs throughout acquisition life cycle: system development, deployment, and post-deployment support.
- How to *validate* the effectiveness of OA system security measures and knowledge of vulnerabilities into the ongoing development for systems in an operational, testable form that system integrators and administrators can employ, and acquisition program managers can assess.

Specifying the security requirements for software systems

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.

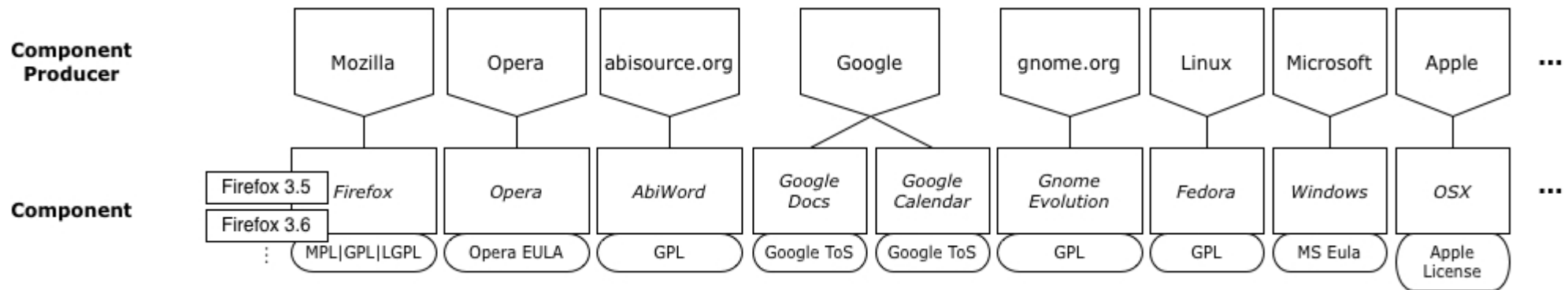
Case Study:

Securing software product lines within an OA software ecosystem for enterprise systems

Software 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*.

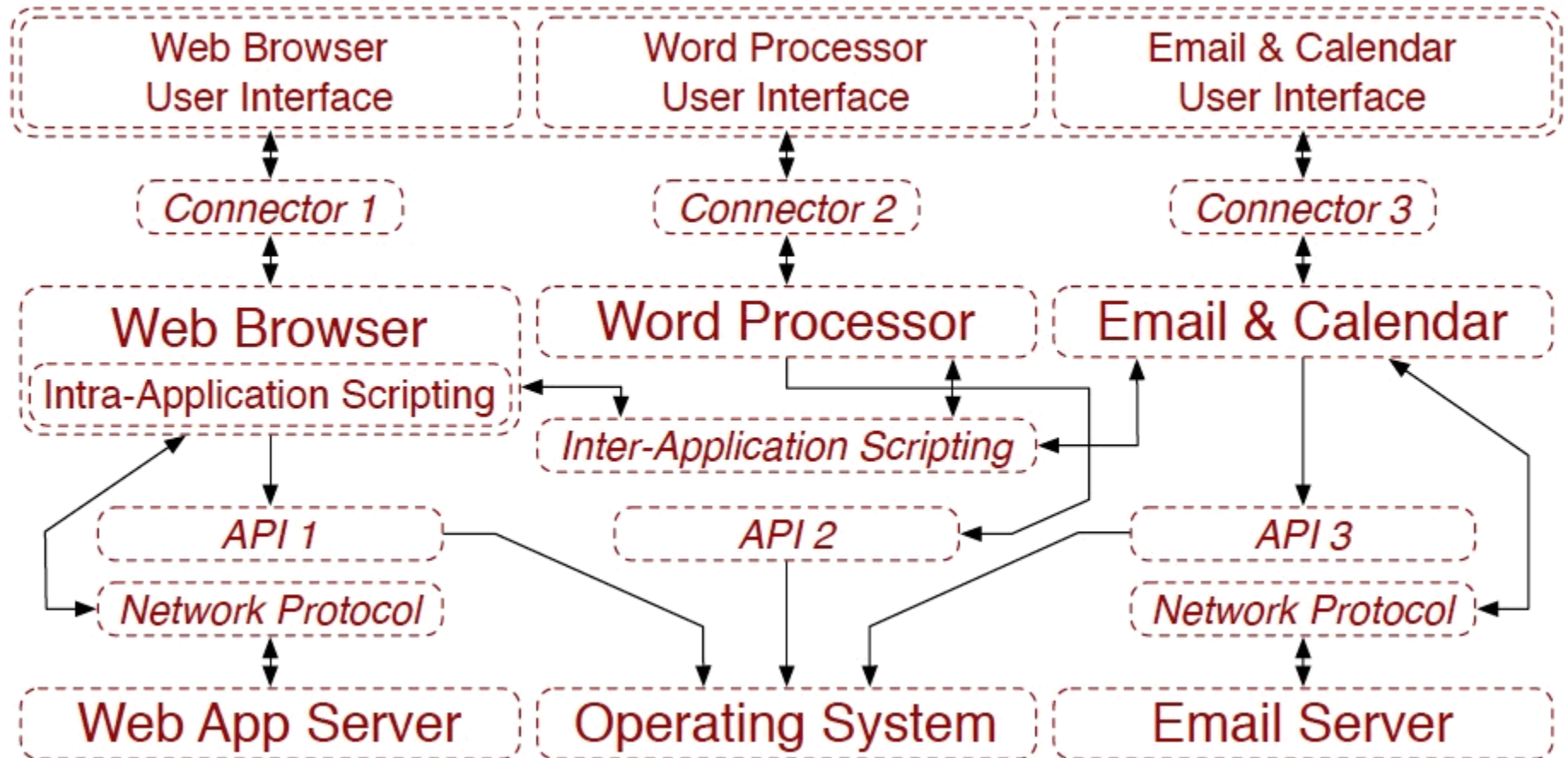
Software ecosystem of producers and the software components for an enterprise system



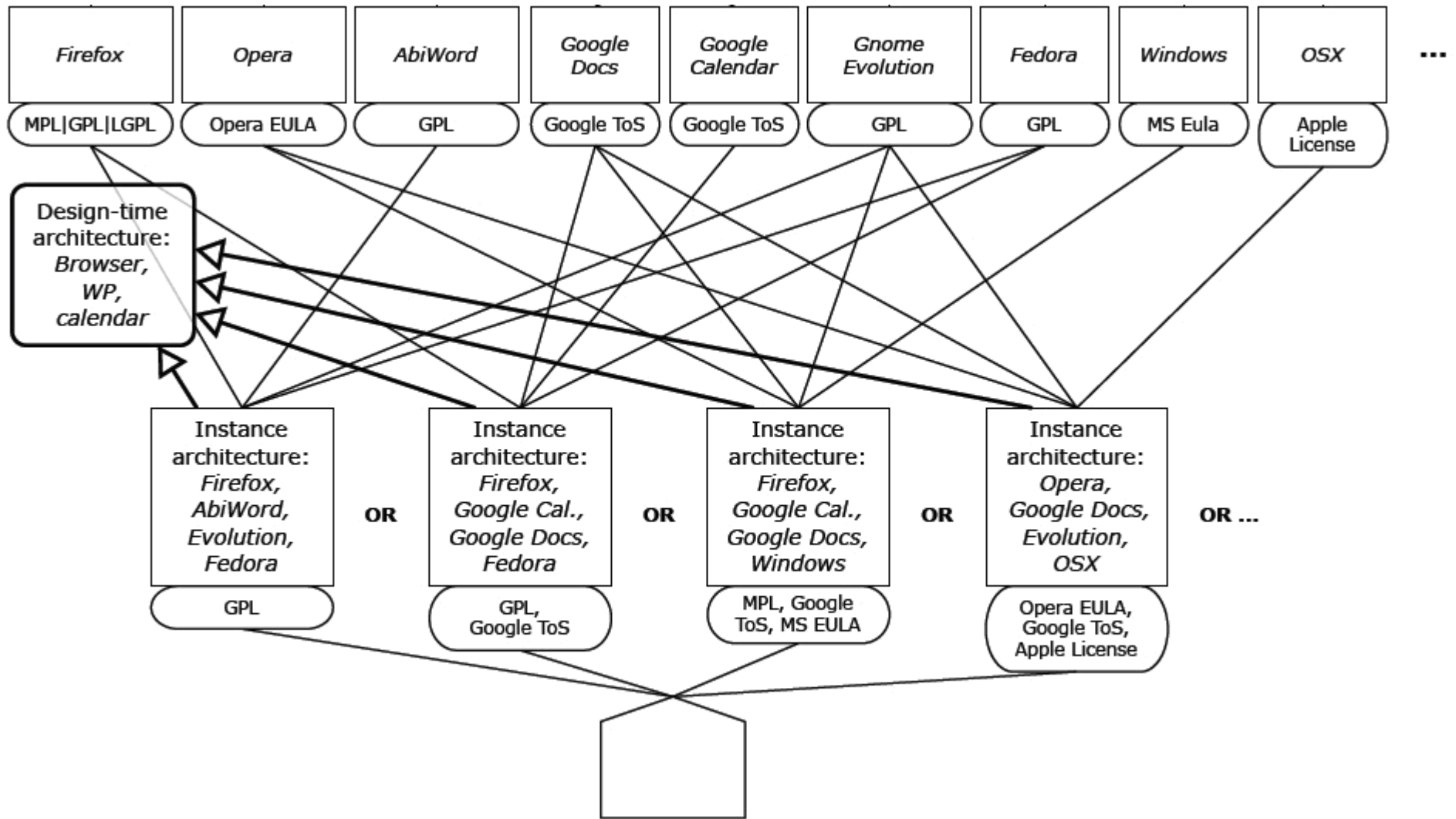
Framework for specifying multi-level OA system security policy

Security policies	
Developers, system integrators and users	Persistent data
System configurations	
Components	Ephemeral data
Connectors	
Platforms	User I/O data

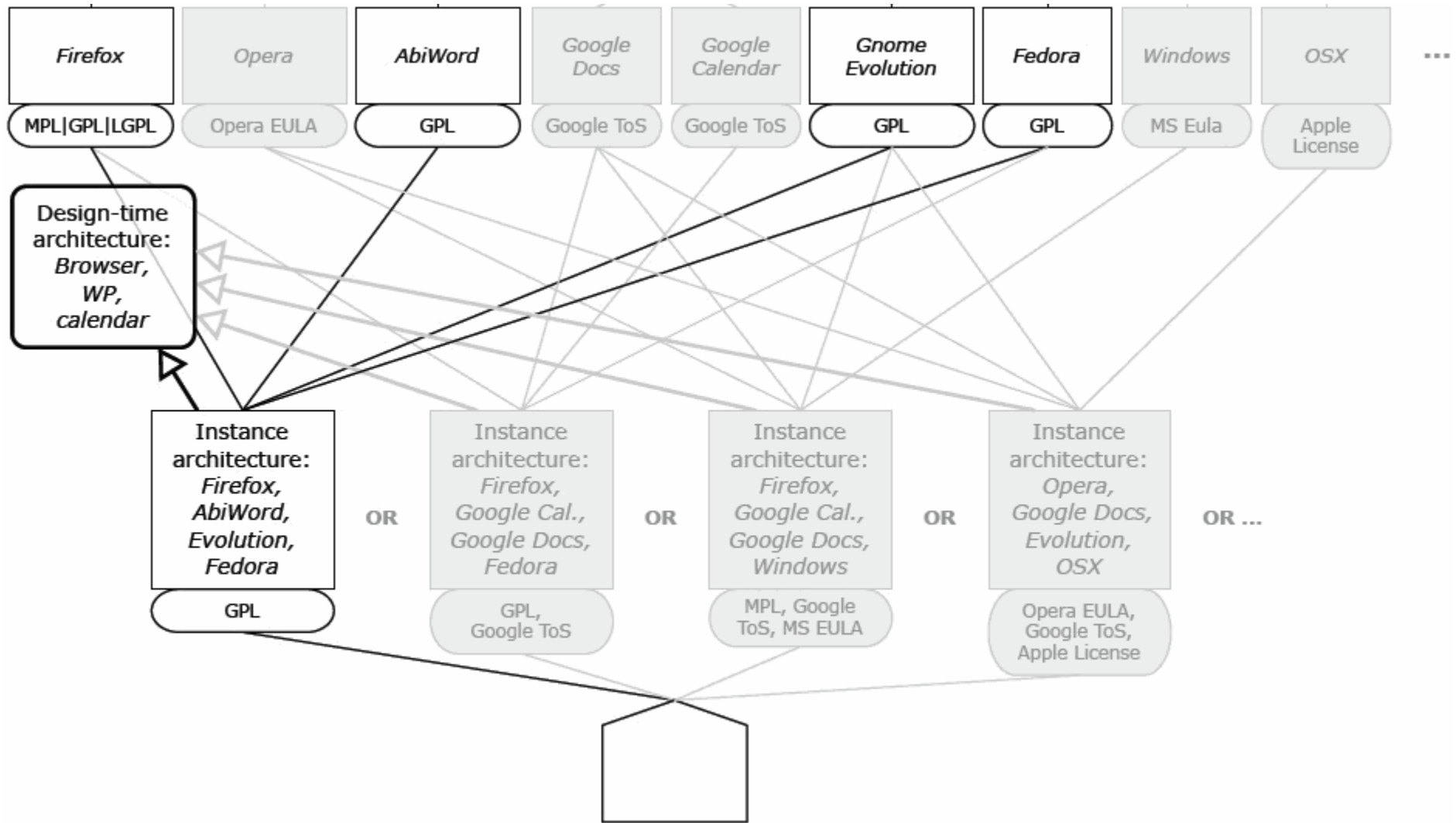
A *design-time* specification of an OA enterprise system that accommodates multiple alternative system configurations



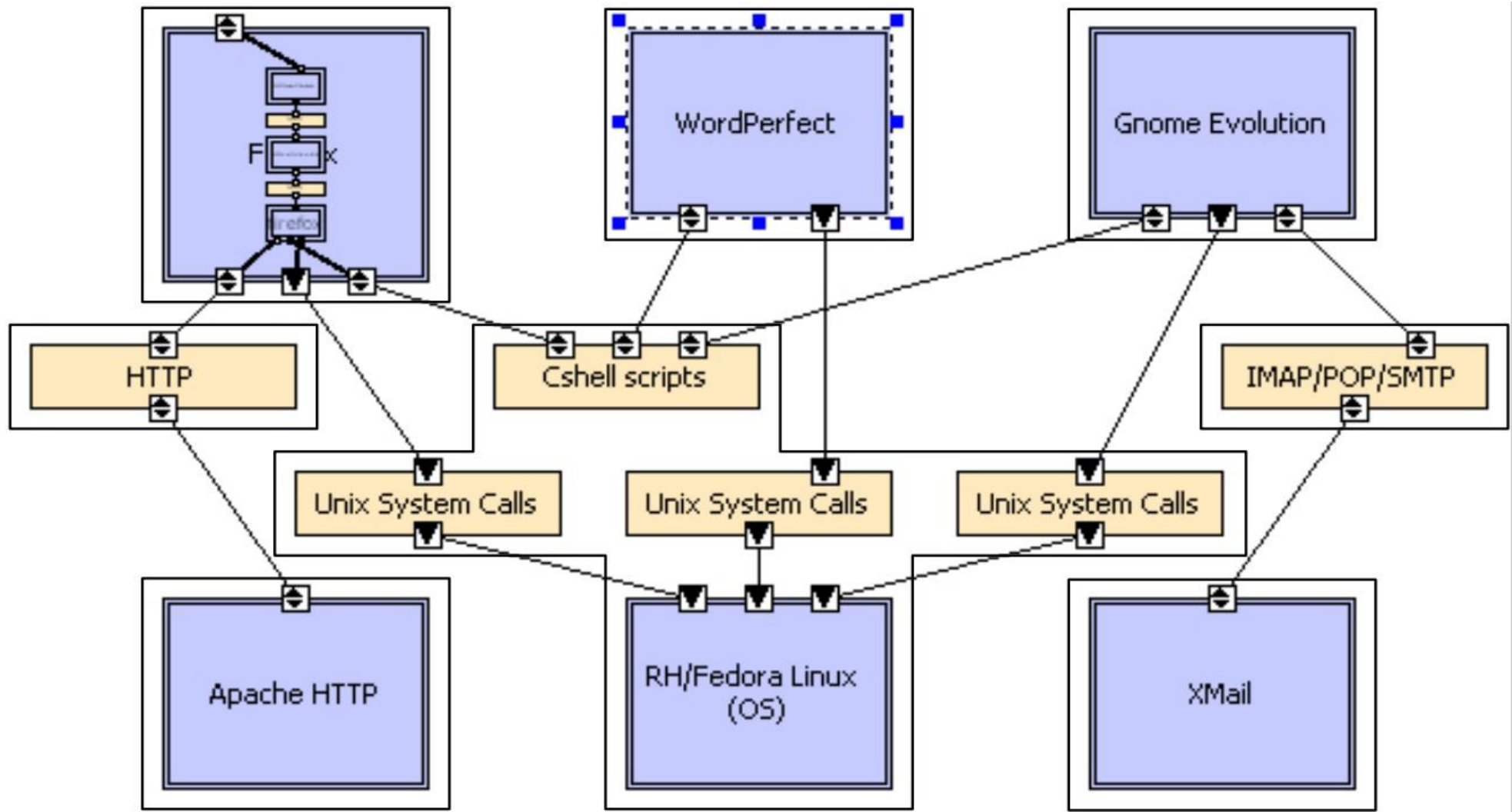
Software product line that provides alternative, functionally similar components compatible with the reference design-time architecture



A *build-time deployment selection* among alternative components that produce an integrated enterprise system within the product line



A security capability specification encapsulating the run-time configuration instance via multiple virtual machines (e.g., using *Xen*)



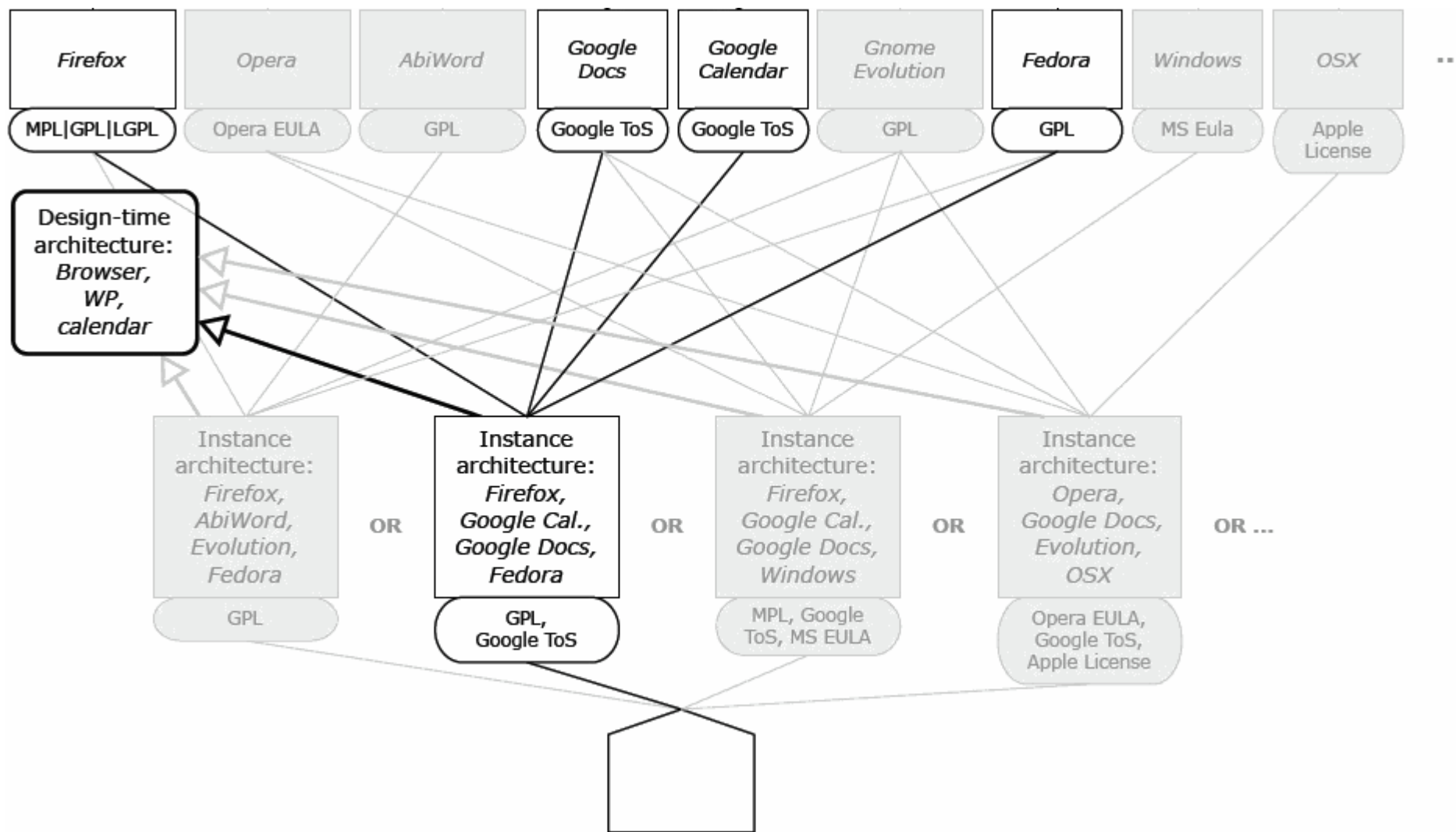
An end-user *run-time deployment* version of selected components within enterprise system product line utilizing security library, **SELinux**, for enforcing mandatory obligations and rights.

The screenshot displays a desktop environment with several applications open:

- Web Browser (Mozilla Firefox):** Shows the website for the "GAME CULTURE & TECHNOLOGY LAB". The page includes a navigation menu, a search bar, and a "Missions" section. The mission statement reads: "The mission of the Game Culture & Technology Lab is to play with how game metaphors, design principles, and technologies can be utilized for alternative content and context delivery. The focus is on the next generation Internet and beyond." It also mentions that the approach combines theory and practice, art and science, education and entertainment, to create an environment that supports diverse forms of expression in a wide range of applications. The methods include sampling, reuse, hacking, appropriation, reverse engineering, and custom creation in the interest of open-source innovation and critical intervention. Since its inception in 1999, the Game Lab has been physically housed in the *Clare E. Foye School of the Arts (SOA)*. In 2010, an established a co-located facility in The California Institute for Telecommunications and Information Technology (CITe).
- Calendar Application (Evolution):** Shows a calendar for Monday, April 26, 2010. There are two tasks listed: "1:00pm Proposal review meeting" and "3:00pm Work on JSS paper draft".
- Document Editor (LibreOffice Writer):** Displays a slide titled "A Composed Open Architecture Software System at Run-Time". The slide content includes a terminal window showing SELinux commands and output. The terminal output is as follows:

```
liveuser@localhost:~$ cd /selinux
liveuser@localhost selinux]$ ls
access      checkreqprot  compat_net    deny_unknown  initial_contexts  nls                policyvers    user
avc         class         context       disable        load               reject_unknown    reject_unknown
booleans    commit_pending_pools  create        enforce        member             policy_capabilities  relabel
liveuser@localhost selinux]$
```

Updated *post-deployment system configuration*, using alternative but functionally similar components within enterprise system product line



An end-user view of the updated alternative run-time system configuration

The image displays a user's desktop environment with several open applications:

- Browser (Mozilla Firefox):** Displays the Game Culture & Technology Lab website. The page title is "GAME CULTURE & TECHNOLOGY LAB". The content includes a navigation menu, a search bar, and a "Mission" section. The mission statement reads: "The mission of the Game Culture & Technology Lab is to play with how game metaphors, design principles, and technologies can be utilized for alternative content and context delivery. The focus is on the next generation Internet and beyond." It also describes the approach as combining theory and practice, art and science, education and entertainment, and lists methods like sampling, misuse, hacking, appropriation, reverse engineering, and custom creation.
- Google Docs:** A document titled "A Composed Open Architecture Software System at Run-Time" is open. The document content is partially visible, showing a diagram or flowchart related to the architecture.
- Google Calendar:** Shows a calendar for Monday, April 26, 2010. A meeting titled "Proposal review meeting" is scheduled for 1:00 PM to 2:30 PM. Another meeting titled "Work on ISS: paper draft" is scheduled for 3:00 PM to 5:30 PM.
- Terminal:** A terminal window shows the user "liveuser@localhost" running the "ls" command in the "/selinux" directory. The output lists various files and their permissions, including "access", "avc", "booleans", "class", "commit_pending_bools", "create", "deny_unknown", "disable", "enforce", "initial_contexts", "load", "member", "modinfo", "modprobe", "mount.fuse", "mount.nfs", "mount.nfs4", "mount.nfs-fuse", "mount.nfs-3g", "pppoe-sniff", "pppoe-start", "pppoe-status", "pppoe-stop", "ppp-watch", "pvchange", "pvck", "pvcreate", "stdisk", "vgremove", "shutdown", "vgrename", "vgscan", "vgsplit", "weak-modules", "ypbind", "policyvers", "reject_unknown", "relabel", and "user".

Discussion and conclusions

Discussion

- Our goal is to develop and demonstrate a new approach to address challenges in the acquisition of secure OA software systems.
- Program managers, acquisition officers and contract managers will increasingly be called on to provide review and approval of security measures that are employed during the design, implementation, and deployment of OA systems.
- We seek to make this a simpler, more transparent, and more tractable process.

Conclusions (1)

- Our research demonstrates how complex OA systems can be designed, built, and deployed with alternative components and connectors into functionally similar system versions, to realize for overall system security.
- We described a scheme to specify and realize OA system configurations that are compatible with existing security mechanisms.
 - Our scheme does not assume that individual system elements must be secure before inclusion into the secured OA system's configuration.
- Central to our scheme are software product line concepts integrated with security mechanisms.
- We provided a case study that reveals how to specify a secure OA enterprise system product line accommodating diverse needs of software producers and developers, system integrators, users and acquisition managers.

Conclusions (2)

Next steps:

- Articulate the *process* how to simply and transparently specify and assess OA system security using streamlined security policies.
- Develop and demonstrate a prototype *automated environment* that can support the modeling and analysis of OA system security policies and alternative version OA system configurations, in ways that address the diverse needs of acquisition managers, system integrators and end-users.

Acknowledgements

Research described in this presentation was supported by grant #N447602-12-1-0004 from the Acquisition Research Program at the Naval Postgraduate School, and from grant #0808783 from the National Science Foundation. *No review, approval, or endorsement implied.*