Enterprise Computing: Collaboration System Example

Jim Fawcett

Software Modeling

Enterprise Computing Combines Structures

- Enterprise computing binds together a business with its partners, suppliers, and customers.
- May integrate many functions:
  - Inventory control, order processing, product disclosure, product design collaboration.
- Likely to be peer-to-peer with “distinguished” peer that coordinates activities.
  - Partners work together through a collaboration subsystem.
- Uses web-based service-oriented architecture.

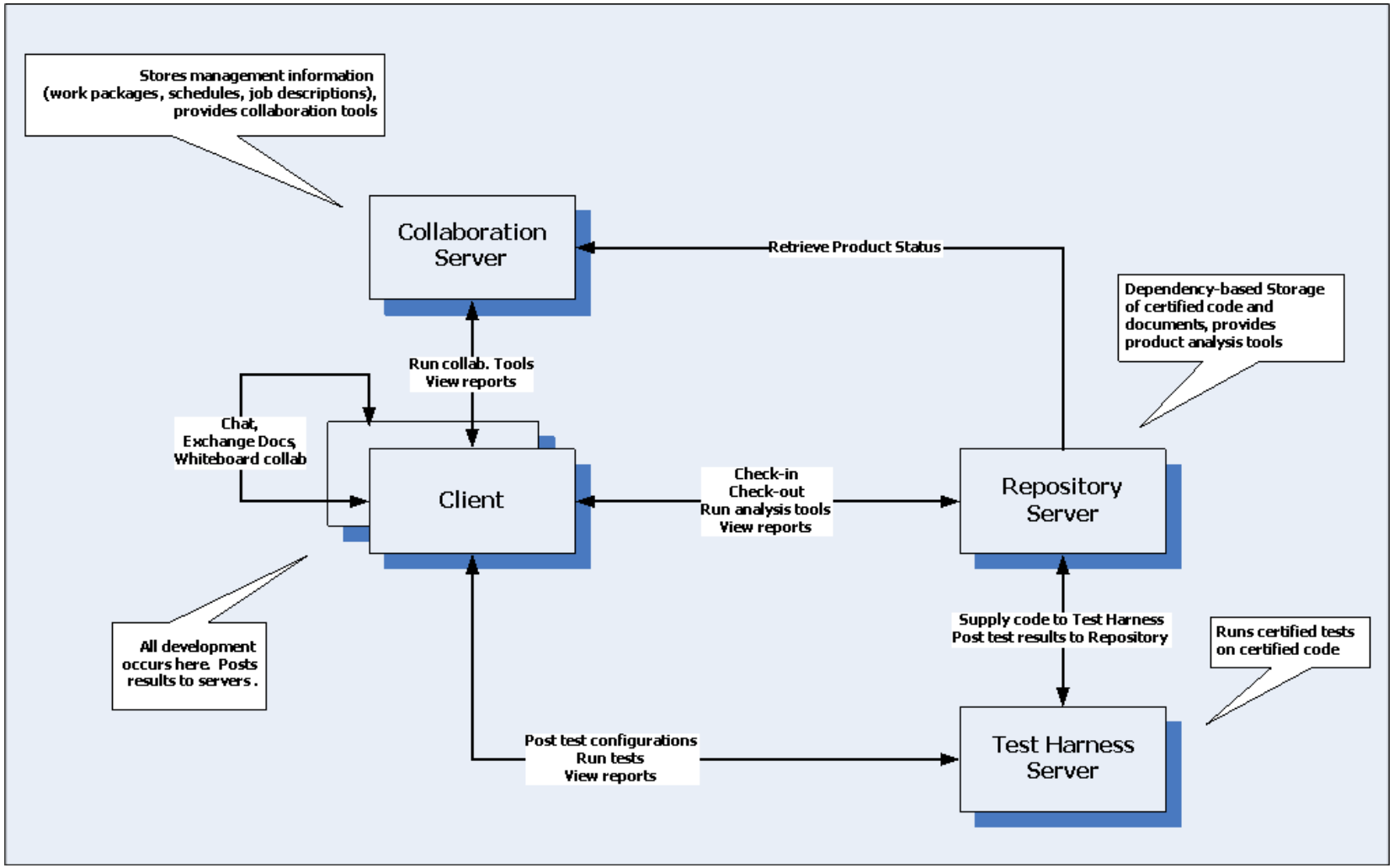
Collaboration System

- System that focuses on sharing of processes and products among peers with a common set of goals.
  - Primary focus is organizing and maintaining some complex, usually evolving, state:
    - Software development baseline
    - Set of work plans and schedules
    - Documentation and model of obligations
    - Communication of events
- Example:
  - Collab – CSE784, Fall 2007, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/CServ.htm

Virtual Collaboration-Repository-Testbed Server System (VCRTS)

- Servers:
  - **Collaboration**
    - Holds work package definitions, schedules, job descriptions, collaboration tools (whiteboard, chat, …)
  - **Repository**
    - Holds the developing project baseline, e.g., code, test drivers, documentation, test results, …
  - **Test harness**
    - Performs all certified tests, only on repository products.
  - **Clients**
    - Code development, test development, local testing, chatting, whiteboard collaboration, …

Example Collaboration System



Virtual Servers

- Not defined by machine boundaries
  - May have multiple servers on one machine
  - May have multiple machines implementing one server, e.g., repository, testbed
  - Can be easily replicated
    - Download installer
    - Select desired contents from source
    - Create server
- All servers derive from abstract virtual server
  - Virtual server is one of the core services.

Virtual Server Uses

- Project has VCRTS
  - Manages all certified project products
    - Code baseline
    - Test code and results
    - Documentation
- Teams have VCRTS
  - Local management for each team
- Company has VCRTS
  - Manages company's reusable code base

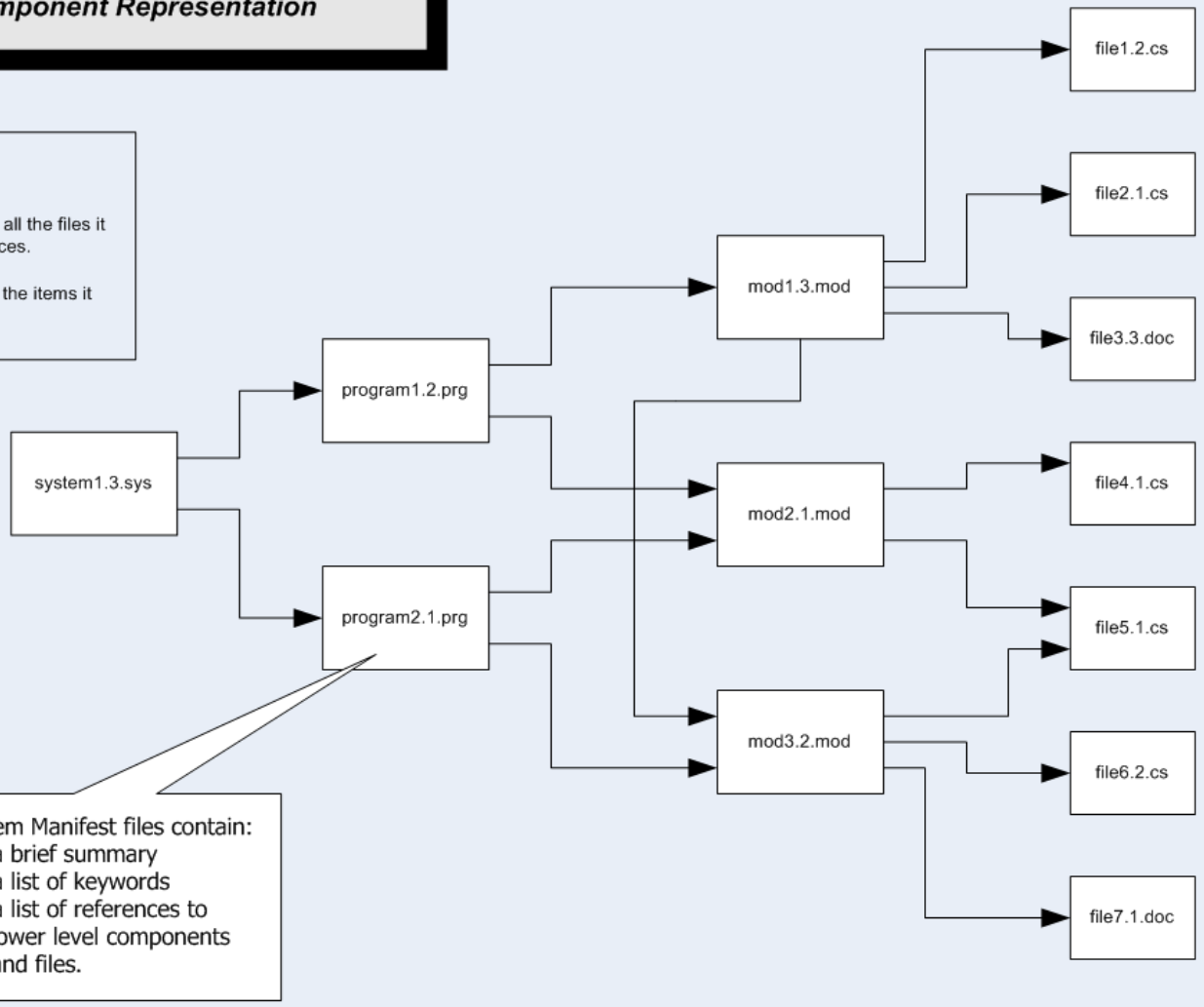
Layered Structure

- Provides a structure based on:
  - System services—things the user doesn't think about
    - Communication, storage, security, file caching, …
  - User services—things the user manipulates as part of the use of the system
    - Input, Display, Check-in/Check-out, …
  - Ancillary—things that are not part of the system mission but are necessary
    - Logging, extension hooks, test hooks, …

Component Representation

Definitions

- **Item:**
  A manifest and all the files it directly references.
- **Component:**
  An item and all the items it references.



Item Manifest files contain:
- a brief summary
- a list of keywords
- a list of references to lower level components and files.

Versioning Concept

New versions caused by change in file F2:

- F2.2 is the new version of file F2.1
- M2.2 is new version of module manifest M2.1, resulting from referring to new version of F2. Note that it still refers to the same files, F1.3 and F3.2 as M2.1
- Module M1.2 is new version of M1.1 resulting from referring to new version, M2.2 It still refers to F4.1.
- The RI for Program P1 has not decided to use the new version of M1 yet.

RI for a module may link a new version of her manifest to any file or lower level manifest. The RI may NOT link a higher level manifest to the new version. That is allowed only by the RI for the higher level module.

The versioning of M1.2 is open – indicated by dashed lines – meaning that its RI may change links in that manifest without generating a new version.

However, M1.2 may not be checked-out for modification until its versioning is closed. Also, it may only be a part of a test configuration that does not have modules linking to it, until its versioning is closed.

Older versions:

Older versions are retained in the Repository. This supports two critical activities:

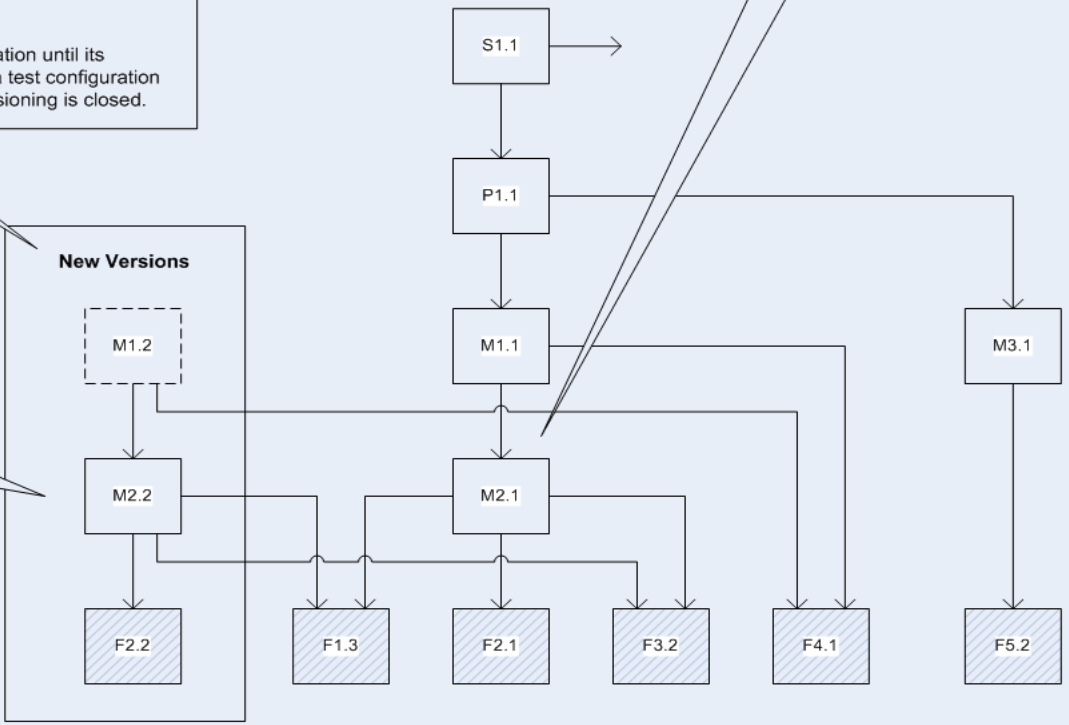
- Developers can access complete configurations for older products that are still in service to provide support for customers.
- A configuration can be easily rolled back should an earlier change prove to be incorrect or lead to other problems in the developing system.

Manifests and Files:

Manifests are XML files that define Systems, Programs, and Modules, simply by linking to lower level manifests and files. Files are shown with hatched pattern, manifests have a solid background.

All links are dependency relationships. Thus, Both modules M2.1 and M2.2 depend on file F1.3. If two modules have no dependency on each other, they are not linked.

Note that the Repository need make no distinction between Systems, Programs, and Modules. That is simply a developer's design distinction.



Peer-to-Peer

- Distribution of parts that cooperate on a mission by sending each other commands and messages.
  - Parts may or may not be identical but probably have identical layered system services
  - Usually part of a collaboration system
  - May have a “distinguished” peer
  - Development attempts to provide one set of core services and build peer personalization on top of that
- Example:
  - Software Matrix, Gosh M.S. Thesis http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/softwarematrix.htm

Service Oriented

- System composed of
  - Set of autonomous services
  - Software glue that binds the services together
- Focus on
  - Reliability, availability, composability
- Example:
  - VRTS – CSE784 Project, Fall 2008, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/Vrts.htm

Agent Based

- System uses software agents
  - Semiautonomous, mobile, task-oriented software entities
  - May be scheduled
  - Provide scriptable user-specific services
    - Collect information from a large set of data
    - Perform analyses on changing baseline and report
    - Conduct specific tests
    - Make narrowly specified modifications to baseline
- Example:
  - CSE681 Project #5, summer 2009, http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/Projects/Pr5Su09.doc

Project #5

- Peer-to-peer?
  - May initiate analyses from client
  - May schedule analyses and notify users of results
- Collaborative?
  - QA, management, developers, and architects all care about the analyses and results.
  - How do we overtly support collaboration?
- Service oriented?
  - Communication and notification are probably service based.
- Layered?
  - If we extend by sending libraries to remote machines to be run from tool holster, we may want to have the holster provide execution services—a sandbox—to enhance security.
- Agent based?
  - We probably want to schedule tests, tailored to specific users, e.g., QA, team lead, architect.

ENGINEERING@SYRACUSE