Project Center Use Cases

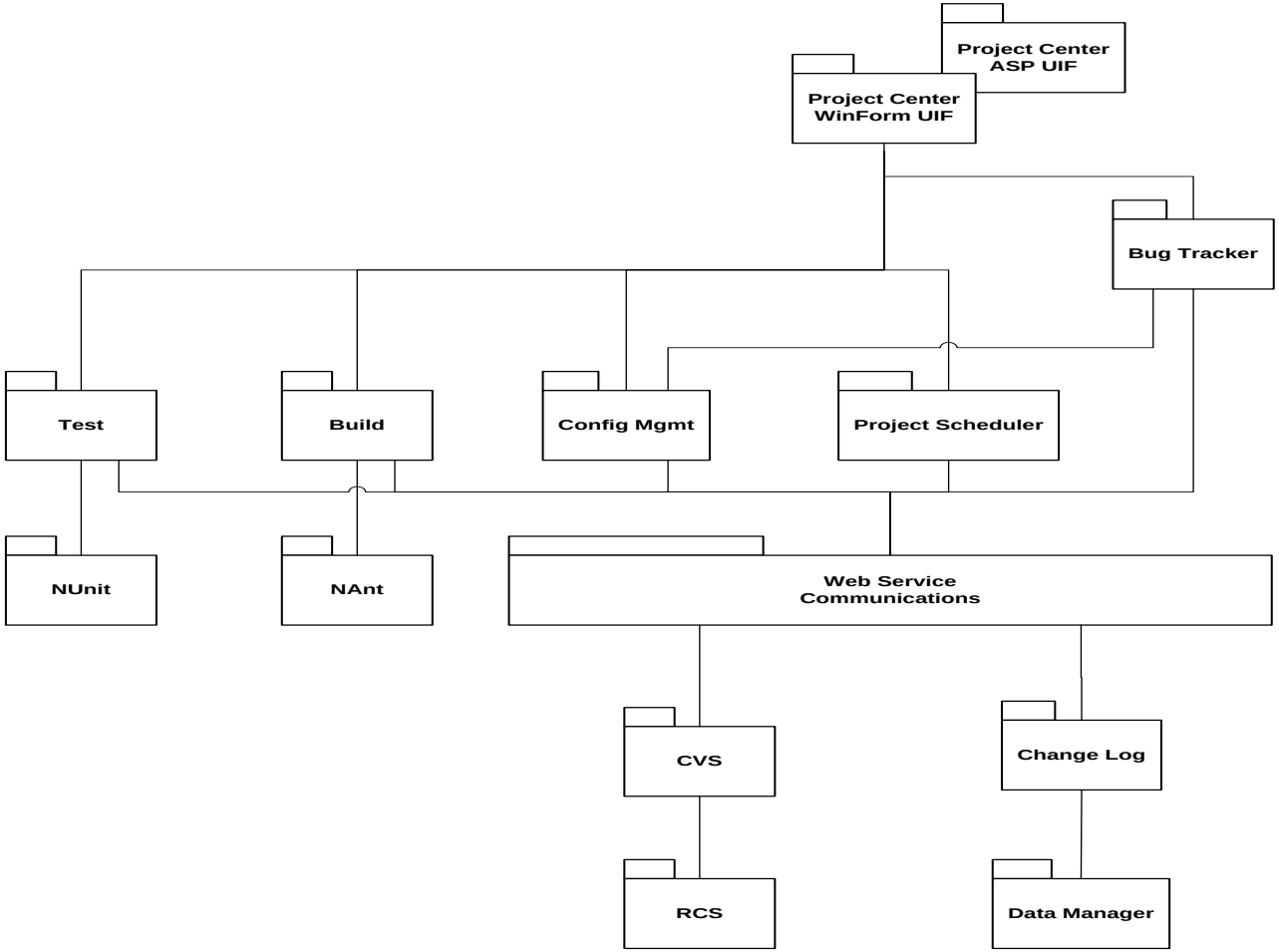
Jim Fawcett

Software Modeling

Project Center

- Project Center was a project assigned to another course, CSE784—Software Studio, but these slides are also relevant to our projects #1 and #4.
- Project Center is a software development collaboration system, built from a number of open-source projects with code wrappers to enable them to communicate and collaborate effectively.

Project Center Packages



Developer's Daily Use

- Look at Project Center schedule, notices pages, alerts
- Get today's work from Project Center:
  - Get latest source of my code from CVS including NAnt build script.
  - Get latest build of other team's code on which my code depends
  - Make modifications or additions to my code.
  - Build my source, incorporating libraries on which my code depends, using NAnt.
  - Run NUnit on my source and iterate, recording and working off bugs.
  - Commit changes to CVS. That automatically results in change log entry. Any components frozen cannot be committed to CVS.
- Send libraries of my latest code that others need to Project Center via CVS.
- Project Center tools used:
  - CVS, NAnt, NUnit, Schedule, Change Log, Bug Tracker

Developer at Customer's Site

- Walk customer through requirements issues, demo part of code, record customer issues in P.C. from customer site.
  - Login to P.C. via browser
  - View Requirements Database
  - Open CVS web interface from browser
  - Extract demos from CVS and run at customer's site
    - Can modify and rebuild onsite if developer takes laptop with P.C. installed.
  - Walk through bug reports and change logs to discuss progress
- Project Center tools used:
  - CVS, Requirements Database, Bug Reports, Change Log, all via Asp—possibly NAnt and NUnit run on laptop.

Use of Project Center for Qualification

- All Qualification builds—typically four or five—ready to go in CVS with NAnt scripts to rebuild should the customer want to peek at internals.
  - Usually extract just executables
  - But may rebuild any of the test builds with single NAnt command
- NUnit set up to run each of the Qualification Tests, showing, by default, only what is necessary for qualification.
  - Each test procedure captured in help module
  - Requirements database synchronized to qual test showing B-Spec requirement for this test and A-Spec requirements it maps to.
- Project Center tools used:
  - Requirements Database, CVS, NAnt, NUnit, Help

Customer's Use of Project Center for Maintenance

- We deliver Project Center along with product
  - CVS, NAnt, NUnit all set to run regression tests on delivered product
- Project Center help has inserted module that documents product code—a supplement to delivered documents
- Customer can now immediately do modifications and builds without studying the product packaging for weeks.
- Project Center tools used:
  - CVS, NAnt, NUnit, Help

Manager's Use of Project Center

- Review status of builds and tests through schedule-based status reports. What is important here is clarity of the information transfer, not having a pretty or fancy calendar. You are not asked to reinvent Outlook or Microsoft Project Manager.
  - Schedule says Display team has scheduled integration build to integrate with Data Editing team.
  - Schedule shows that Display team has not installed the required build the day before integration.
  - Checks Quality Assurance report for last display build and reviews (from CVS).
  - Checks bug tracker reports.
- Checks to make sure that the notification for scheduled integration has been posted (a month ago).
- Sends notice to Display team leader that there will be a meeting in half hour in manager's office.
- After meeting manager posts action items associated with that meeting, assigned to the Display team leader.
  - Note that much of this functionality is fairly close to that supplied with the requirements database and other tools.
- Project Center tools used:
  - Schedule, CVS, Bug Tracker, Schedule Alerts, Action Item Database (Bug Reporter with a different name?)

Architect's Use of Project Center

- Reviews all interfaces held by CVS against the OCD.
- Reviews CVS holdings for implementation and test of each interface's implementations.
- Extracts a team's source and NAnt build script, builds executable and reviews functionality by running NUnit.
  - Each team is required to deliver test drivers with their libraries.
  - Each obligation of the team's code is either demonstrated or a message is stubbed stating its current status.
  - All of this runs under NUnit.
- Architect reviews team's view of its obligations using this process from the beginning of development.
  - Each team is asked to declare its assigned interfaces and provide a fully stubbed implementation at the beginning. It then replaces each stub as the real code is developed.
  - Each stub announces what it will be delivering.
- Project Center tools used:
  - CVS, NAnt, NUnit, Requirements DataBase

Quality Assurance Use of Project Center

- QA member assigned to Display team extracts source from CVS, including NAnt build script.
- Uses NUnit to run series of code standards conformance tests on source.
- Builds executable or library with test drivers, supplied by Display team.
- Notes warnings.
- Runs QA build and notes functionality supplied.
  - Each team is required to supply NUnit tests that display what works and have stub messages for what does not yet work.
- Writes QA report and stores in CVS, associated with the Display build.
- Project Center tools used:
  - CVS, NAnt, NUnit with special QA tests

Some Observations about Design

- Most of the custom tools are minor variants of a single design
  - Requirements Database
  - Bug Tracker
  - Change Log
- It would be extremely useful to have web service interfaces to add and modify entries in any of the databases.
- Examples:
  - When modified code is committed to CVS it would be simple to have Project Center user interfaces insert the change record to Change Log using its service interface.
  - When Qual Testing, it would be simple to synchronize NUnit test with display of B-Spec requirement and A-Spec requirement in separate windows using web service access to Requirements Database to search for requirement by number.
  - Meetings and reviews could be scheduled using web service interface to scheduler.
- It may also be useful to provide a command line interface for insertion and modification of database entries. Will make our tools consistent with the open-source tools, which all have command line interfaces.

Observations about Design

- Should designate Project Center server
- Users have Project Center WinForms interfaces on their client machines.
- Users can access most of the Project Center functionality through a browser, viewing Asp pages from server.
  - All persistent data resides on Project Center server
    - CVS/RCS code and document storage
      - Should support private and public storage for each team
    - NAnt build scripts (in CVS)
    - Requirements
    - Bug Reports, Change Logs
    - Schedule and Tracking information
  - Tools may reside on client or server. Architect will choose with help of team leaders.

Prototyping

- The best way to decide how to glue all this together is to use the open-source tools before committing to the Project Center structure.
  - Suggest we download all of them and use them with a couple of small example projects, perhaps CSE784, Project #1 for this year.
  - Since the custom tools are entirely under our control, they can fit into the same structure needed for CVS, NAnt, and NUnit.

Distributing Workload across Teams

- Database and Security:
  - Designs queries for all accesses to any of the databases, providing interfaces with insertion, update, and extraction
- Open-Source Tools team:
  - Prototype use of open-source tools with Project #1
  - Responsible for help subsystem design and implementation
  - Provides help contents for open-source tools
- Communication team:
  - Provides web service message-passing for custom tools
  - Provides web service message-passing wrapper for open-source tools.
    - All open-source tools have command line interfaces so this should be straightforward.
    - Tutorial links—later slide—indicate how command line interfaces work.

Distributing Workload across Teams

- Scheduling and Tracking team
  - Design and implement Scheduler, Requirements Database, Bug Tracker, Change Log.
  - Design and implement support for inserting new tools.
- WinForms Interface team
  - Will have plenty of work with interface.
  - Main problem is getting early access to code to call.
- Asp Interface team
  - Plenty of work with interface pages
  - Same problem as WinForms team
- Test team—plenty of work already
- To get a quick start, User Interface teams could work out detailed use cases
- Then start hooking up open-source tools using communication stub
  - Comm stub is just post-message, get-message interfaces used in the local process as a stand-in for commlink to another machine until that becomes available.

Derived Requirements

- User authentication
- Interoperation between tools
  - Bug Tracker cites CVS entries
  - CVS writes to Change Log
  - Project Scheduler reads CVS and/or Change Log, NUnit log

Open-Source Tool Tutorials

- Tutorial links
  - CVS: http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/
  - RCS—used by CVS: http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/rcs/
  - NAnt: http://nant.sourceforge.net/help/index.html
  - NUnit: http://www.nunit.org/getStarted.html
- Notes:
  - You will find, looking at these links, that all these open-source tools provide command-line interfaces.
  - That means that accessing them through a web service interface should be straightforward.

ENGINEERING@SYRACUSE