

Services, groups and project transition

Robert Doiel

rsdoiel@usc.edu

Software Engineer

ITS Web Services

USC

Topics

- My organization (WS)
- Generally available resources
- Some things to think about
- Strategies
- Problems in project transition

ITS Web Services

- 3 Programmers
- 3 Graphic designers
- 2 Content specialist
- Project Management/Management

Typical team

- Project manager
- Information Architect
- Graphic Designer (UI Specialist)
- Programmer
- Content Specialist

Services available developer

- Your Unix user account
- POP, IMAP, LDAP
- .htaccess
 - Basic Authentication
 - Shibboleth
- cwis.usc.edu

Client side

- XHTML/CSS/Web forms
- Javascript (AJAX)
- Flash (has issues of ADA compliance but improving, getting less popular then Ajax)

Server side (cwis)

- PHP 4 (migrating to 5)
 - Access to IMAP, POP, LDAP, databases, etc.
 - Problems and security issues
 - Pear (PHP Modules)
- SSL now available for cwis.usc.edu (www.usc.edu)
- .htaccess basic and shib

Some things to think about

- Development/Testing/Production
 - Asbed's definition: Any service which is required to perform ones job should be a production service.
 - Production means different things to different groups
- ITS is Unix (Sun) focused for servers and services.

ITS Organizational constraints

- Recently reorganized
- Limited budget for hardware, software and personnel
- ITS is still driven by competing agendas

ITS Technology constraints

- PHP 4.3.x not 5.x yet
- Java servlets are theoretically possible, only if your client is the ITS production guys
- Ruby lots of talk, no ITS wide production systems available yet

ITS Technology constraints

- No central DB support, `web-mysql.usc.edu` is pretty much restricted to WS.
- Oracle is for big iron projects with budget
- No central Subversion (though CVS is installed)

The Maintenance Issue

- Who
- Where's the funding
- Who has authority to commit resources
- Organization churn

Typical biases among IT staff

- not invented here
- one more problem...
- not enough time, budget, people

Some observations

- applications live longer than planned
- inheriting software is a drag
- no good deed goes unpunished in some groups

And more issues...

- who has time to read documentation?
- who has time to read source code?
- its not supported by a company therefore ...
- but the company support is clueless ...

Strategies

- Take advantage of existing infrastructure and do not require changes in the infrastructure itself (build on top of it)
- Package your application, including test code
- Write your software with maintenance in mind
- Use version control

Don't forget

- OOP works in web applications too
- isolating UI, engine and storage also works in web applications
- take advantage of design patterns
- Write to APIs
- Outside services for Mashups

Remember to

- Make sure your code is readable
 - limit mix language code
 - clear comments go a long way
 - clear code is even better
 - make it easy on your eyes, use white space!
- Small and simple if beautiful
 - limit the number of lines of code
 - limit the number of files involved

Also recall

- Write (unit) tests first!
- You can write code to generate code (and unit tests)
- Take advantage of existing services where appropriate

Problems in project transition

- Champion has vanished
- Reorganization
- Priority shift
- Product quality and trust

Some of the big problems

- No 'Extra time' for new projects
- New projects are often seen as new problems by IT folks
- 'not invented here' (NIH) problem
- No budget, no champion

Bridging these problems

- Building trust for your solution
- Importance of documentation
 - Write to your audience(s)
- Pick known technologies in your customer's solution

Communication is critical!

- It is really, really important to be a good writer, communicator, and translator
- You will have to deal with both engineers and non-engineers in your career!
- Understand what drives the budget and who runs the budget

Moving from a development machine to an ITS server

- Problems in software versions
- Problems in software location
- Problems in getting basic access and permissions
- Problems with account quotas

Approaches

- Automate tests scripts to verify installation
- Automate configuration
 - See if you have the installed version
 - See if you have the correct permissions
 - Set sane defaults
- Short and clear installation instructions and trouble shooting tips

Some concerns around maintenance

- Things live longer than you'd expect
- Who gets called when things go wrong?
- Who gets called when the underlining software environment has changed and the software you developed needs to be fixed?

Avoid Ugly Things

Ugly installation

- You have to modify source code to install
- config file is complicated and undocumented and is generated by hand
- App has hard coded directory, file or login expectations

Software is read more than written

Ugly Code

- Mixed language coding (common in web apps)
- No design, unclear design, shifting design
- No comments
- Doesn't follow design
- No tests showing how things work and that they actually do work

Ugly re-deployment

- No cleanup/uninstall script
- No installation script
- No verification (tests)
- No doc, no references
- No upgrade script or procedure

Testing helps insure trust

- Unit testing
- Performance testing
- Check your software on more than on machine!
- Don't be afraid to have several client drive throughs during the development process

Packaging

- sh and tar can go a long way
- expected contents
 - readme.txt/README
 - INSTALL
 - COPYING
 - HACKING/DEVELOPER.txt
- document dependencies and copying issues (including license issues)

Useful web links:

- ITS page : <http://www.usc.edu/its>
 - (click on Guide to services and support)
- Use USC's search entry off the home page
- You can send questions to me at rsdoiel@usc.edu

Thank you

✉ rsdoiel@usc.edu

