Collaborations

How to succeed

Why collaborate?

- General benefits:
  - Sharing resources is more efficient
  - Sharing views helps to define the right product
  - Same product everywhere
- For our users:
  - Stable well designed software that is the same at every institute
  - New features available at the same time everywhere
  - Data can be shared between institutes
- For us:
  - Overcome our limited resources
  - Share our knowledge to improve our skills
  - Have a stable well designed software
    - to start with
    - or to replace old applications that come to an end

What is a Collaboration?

- "Collaborate: to work jointly with others or together especially in an intellectual endeavor"
  Merriam-Webster's Online Dictionary
- "A team which shares a common purpose, there's mutual trust and everyone uses agreed-upon approaches for the work." *Requirements by Collaboration*, Ellen Gottesdiener
- Concerned by external instead of internal collaboration

Types of Collaborations

- Information sharing (share ideas and problems)
- Common rules (same interfaces, share same data, same workflows…)
- Common development
  - Outsourced development
  - Internal (distributed) development
- Cloning a system is no collaboration
  - It helps other teams to get started faster
  - Almost no resources needed for hand-over
  - But almost no further benefit

Consequences

Information sharing	• Time necessary for collaboration
Common rules	• Common understanding necessary • Partial loss of ownership at each institute • No quick changes possible (agreement necessary) • Resources are allocated with “external” tasks
Common development	• Common development approach necessary • Specificities are hard to be taken into account • Bugs can be introduced by partners

- Which type do we target?
  - My two cents:
    - Minimum: Information sharing
    - Optimum: Common development
  - Decision on type of collaboration at the end of the workshop?

Prerequisites for Success

- Trust
- Clear common goal
- Define a collaboration agreement
  - Be ready to invest resources for partners!
- Define ownership of product
- Get management buy-in … also for consequences!
- Define resources and commit on their availability
- Define the project management
  - Decision making process
  - How to measure success, control and steer
- Define product scope (what's in and what's out)
- Define how to include end user
- Define how to deal with specificities!
- Agree on common development approach (technologies, style, tests…)

Best Practices

- Communication, communication and communication
  - friendly atmosphere
  - kick-offs to get to know each other
  - regular meetings
  - communication tools
- Encourage participation through positive feedback
- Protect resources from internal overloading
- Define steering group with high level management from each institute
- Define lightweight iterative process and review regularly
- Define common functionalities / functional scope through workshops

Best Practices continued

- Include end user representatives during whole iteration
- Make use of (independent) facilitators for workshops
- Choose common technologies and ramp up knowledge to same level
- Most important issue: how to handle specificities?
  - either harmonize workflows
  - or have massive configuration options
  - or extension mechanisms
  - or …
- Start small and grow once the collaboration is stable

Best Practices for Development

- Again: communication, communication and communication (inform others about own actions)
- Initial architecture design through workshops
- Modular structure to ease separate development and limit impact on other modules
- Shared code base
- Bug and feature management
- Test-driven development
- Code reviews for essential parts
- Publish code as open source

Tools

- “The biggest challenge of getting people to work together isn't a technological problem--it is a cultural and organizational one.

  The key to good collaboration in any situation, big or small, is to first define the environment within which you want to operate, and then apply technology--where feasible--as an enabler, and not the other way around.” *Collaboration: A Closer Look* by George Ball

- Web platform incl. cms, wiki, forums (joomla…)
- Video conferences incl. whiteboard and application-sharing (marratech…)
- Configuration management (CVS, Subversion, SourceForge…)
- Bug and feature tracking (Bugzilla, JIRA…)
- Project mgt. (tracks…)
- Build tools (ant, maven, make…)
- Test tools (junit, cactus, cocoon…)
- IDEs (Eclipse…)

Don’ts

- Collaboration partners with separate goals (e.g. improving vs. only running a system)
- Missing mutual trust
- Fight for resources
- Quick-fixes (non-agreed changes)
- Blaming
- Parallel work on same issue due to
  - lack of communication
  - shortage of time
  - no common agreement on requirement details

Manage Risks!

- Analyze risks to minimize them
- Risk assessment in collaboration to set priorities
  - Priorities for requirements
  - Priorities for technical solutions to be tested
  - Priorities for resources
- Steering group checks state of project regularly
- What happens if we (partially) fail?
  - How can the collaboration react?
  - How can our institute react?
- Exit strategies

Don’t worry: It works!

Successful Collaborations at ESRF:

TACO	Framework for device servers
TANGO	Framework for device servers
FABLE	GUI for 3D crystallographic analysis
ISPyB	Information system for protein crystallography
DNA	Online data analysis for macromolecular crystallography
EDNA	Online data analysis for macromolecular crystallography

See Andy’s and Olof’s presentations tomorrow!

Conclusion

Let’s work happily together ☺

Questions ???