Collaboration Patterns for Software Development

Introduction

Software development is a collaborative effort, as such effective communication and coordination is essential. All the time that is spent on unnecessary communication and coordination, leaves less time for the core activities of developing and testing software. The larger a team gets the bigger the problem of collaboration gets. The number of ways a team can collaborate and the ways they can communicate grows exponentially to the number of team members. The team needs a common strategy for how the collaboration should be done. The point with this strategy should be to minimize the overhead of coordination and communication between team members and focus it on what's important.

There are two important aspects of software development. Software development produces software products. Products that have versions, variants, milestones and release schedules etc. This is all part of the high level product tactics. Software development is also about collaborative problem solving. Developers and testers work together to solve problems, this is a creative, social and dynamic activity. These two aspects of software development needs to work efficiently on their own and together.

The technical solution for coordinating the collaborative software development effort is mainly branches in version control tools. The technical nature of branches includes more or less advanced concepts from Software Configuration Management. Neither developers or management can be expected to be SCM experts. Developers and management need to translate their desired tactics and collaboration into SCM concepts they are not familiar with.

Collaboration Patterns

My thesis presents Collaboration Patterns that can be used to efficiently streamline high level product tactics with low level collaborative development. Collaboration Patterns provides high level concepts that are familiar to both management and developers, providing conceptual tools that does not require SCM knowledge.

Split and Regroup Pattern

One of the patterns presented in the thesis is the split and regroup pattern (Fig. 1). The point with this pattern is to create a collaborative concept of a traditional method of working within software teams.

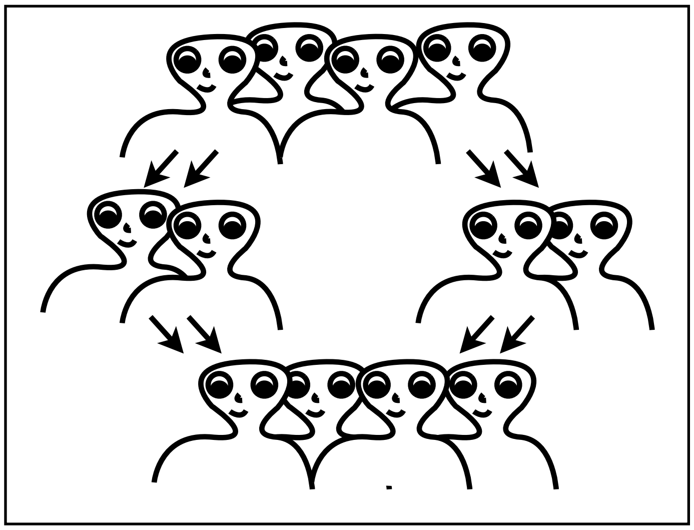
Figure 1. Showing a team splitting into smaller groups and then merging back together.

By splitting the team into smaller groups the subgroups will be smaller and development inside them can be more efficient. When the different groups have solved their respective tasks they can come together into the bigger team.

How do we implement it?

Within each organization these patterns need to be implemented in the SCM tools that are available. Each implementation will be unique to each tool, but the thesis outlines how the different patterns can be implemented in traditional centralized version control tools and distributed version control tools.

Future Work

Future work that can be done is to adapt version control tools, e.g. by writing wrapper scripts around them, that will map the underlying technical concepts of branches etc. to the high level concepts of Collaboration Patterns presented in the thesis.