Issues to Consider for Setting up Distributed Development Model

- An Anshinsoft Whitepaper

Anshinsoft Corp.
5201 Great America Parkway, Suite 320
Santa Clara, CA 95054

www.anshinsoft.com

1. Introduction

In software development geographically distributed development models are very common these days. With offshore development becoming a dominant paradigm today, it's very realistic to have a combined onshore/offshore delivery mechanism. This implies a virtual 24 hour work cycle at a substantially reduced cost leading to a higher ROI for the stakeholders.

A distributed working model means adopting a set of tools and practices that fit the paradigm. This ranges from project management procedures to the specific chain of tools that each individual developer needs to use on a regular basis. And we are seeing a proliferation of such tools in the market today being used by organizations that endorse this model. With more and more tools like Git (the distributed version control system), Mingle[1] (the distributed agile project management tool) and Hudson[2] (the continuous integration server) getting popular, distributed development and deployment of software is becoming the norm of the day. The success of these models is also exemplified by the fact that a rich community of open source software is also using these tools.

However in order to ensure a successful deployment of such a model, we also need to be aware of all issues that may arise in course of time. This document highlights some of these issues based on our experience in adopting the same model with multiple client engagements.

2. Know your Toolset

This is the first and foremost criteria to ensure a success in a distributed development environment. You need to finalize your toolset upfront so that all team members are well aware of the ecosystem. If you are planning to deploy tools which are open source, you need to decide on the specific versions as well. The toolset includes the following:

- Development Environment including
  - IDE
  - Unit testing tool
  - Functional testing tool
  - Local version control systems (if any)
- Distributed Version Management system
- Software build system
- Continuous Integration system (if any)
- Issue tracking and Reporting system
- Project portal
- Database Management Systems
- Messaging Systems
- Software deployment servers along with access control

[1] http://www.thoughtworks-studios.com/mingle-agile-project-management
[2] http://hudson-ci.org/

3. Choose Proper Deployment Options

When you have a host of tools identified, you need to standardize interoperability guidelines between them. A critical aspect of this exercise is to identify deployment options for servers like build servers, continuous integration servers and version control servers. Again there are quite a few options depending on the tools that you choose:

1. One option may be to go for hosted models like Github that offer source code control systems for development.
2. Another alternative may be to have your own installation hosted on Amazon like cloud infrastructure.
3. And the third alternative is to host everything privately over proprietary network with https access to all the participating teams.

4. Streamline the Build Process

You need to choose a build process that supports distributed development. If you have a development stack centered on Java/JavaEE then tools like Maven[3] is a good option. These tools are capable of handling versioned snapshots from remote repositories ensuring consistency of jar versions.

5. Decide on Distributed Version Management Tool

This is one of the very important decisions that must be made at a very early stage of the project. Git is one such tool which offers flexibility of customization and support for a true distributed development model. Have a look at the following structure of code control that is typical of a software development project. Unless you have the proper tool support, it can very quickly become a complete mess.

[3] http://maven.apache.org

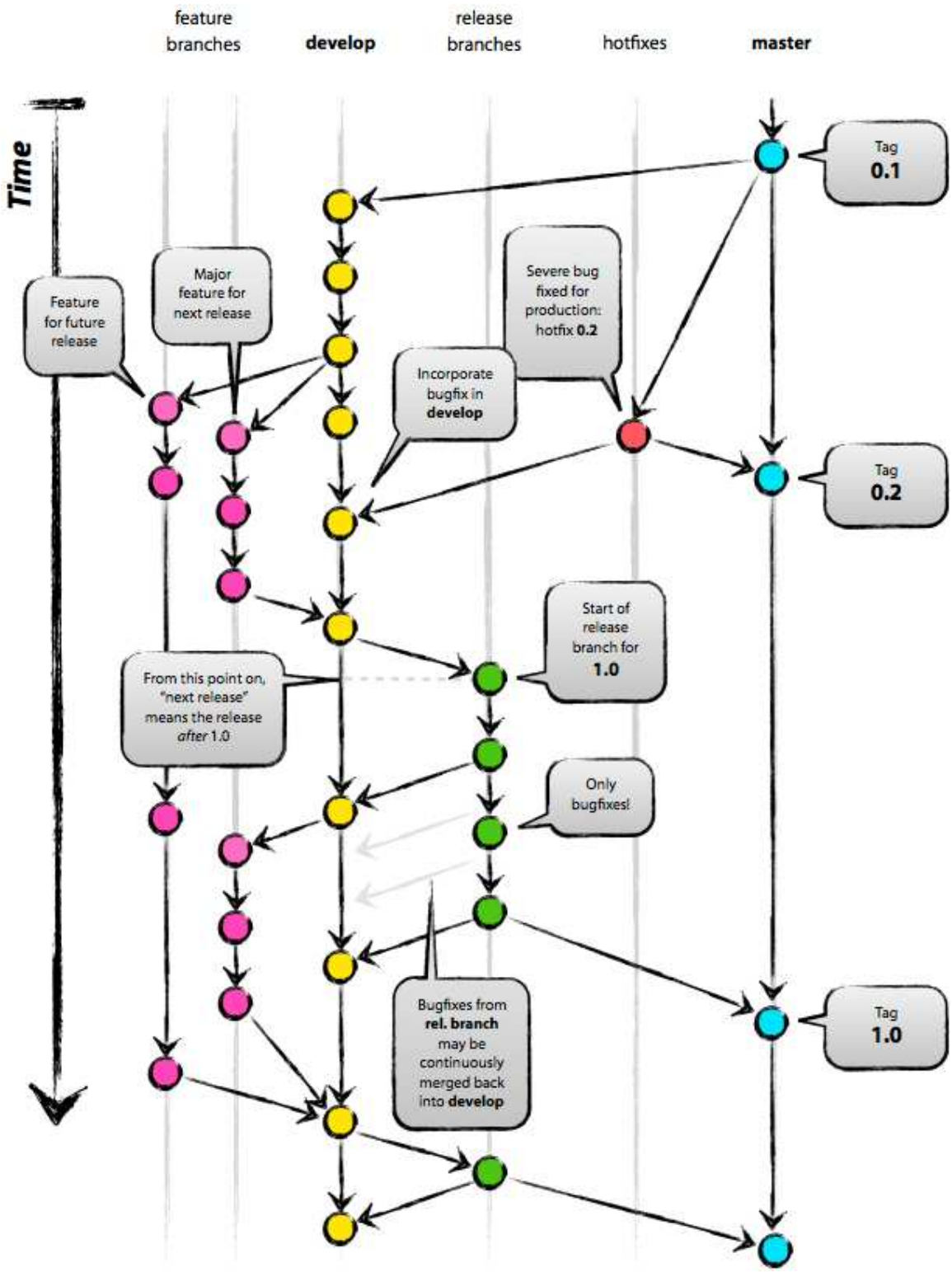

Figure 1 Branching and Tagging in a Version Management Tool

6. Streamline your QA process

QA and testing processes need to be transparent across the team structures. Conventions need to be in place for proper versioning so that the QA team can do all testing against the proper versions. All testing artifacts like test cases, test results and reports need to be versioned and archived. In case you decide for any automated testing tool, the proper server version of the tool needs to be deployed and made available to all members of the testing team.

7. Decide on a Review Process

Proper communication is the cornerstone of successful delivery. For any development project, you need to have agile processes in place for communicating across teams. Consider a typical SDLC phase diagram for a development project:

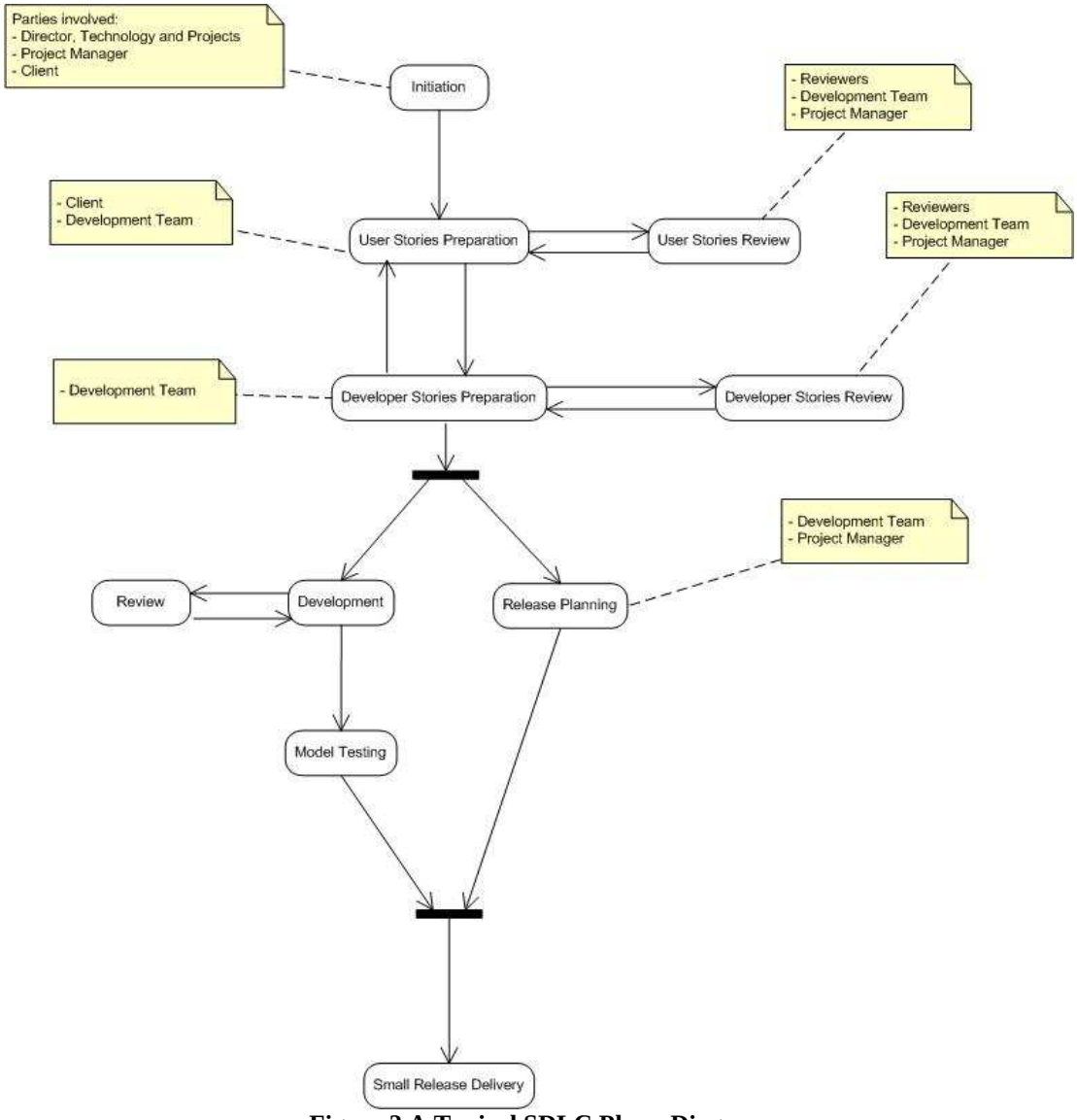

Figure 2 A Typical SDLC Phase Diagram

At almost every phase of the development lifecycle, we have iterations. If we cannot set up a proper communication network across the various participants it will lead to chaos. For iterations and reviews you need to have proper infrastructure in place like WebEx, SharedView and other collaboration toolsets.

8. Minimize Iteration Size

For distributed development, one of the key factors is to decide upon the size of the sprint or iteration. Based on our experience working with clients in a distributed model, we found that shorter sprints deliver more ROI. In fact sprints can be as small as 2 weeks, after which the milestone goes for a review. The review process takes place over the Web and then the issues flow back into the next iteration cycle.

9. Use a proper Agile Project Management Tool

The biggest challenge for a distributed working model of a project is to adapt to changes in requirements and have them transmitted seamlessly across the team structure. A well-designed project management tool offers strong collaboration capabilities across the team, the stakeholders and the client. Select one such tool so that you have an integrated view of the entire snapshot for all the phases of the project. Mingle, from ThoughtWorks is one such tool.