Modern Development Processes and Tools

Christian G. Warden

September 14, 2015

Summary

[Software development teams] can become more productive by adopting improved processes and tools to support these process changes. The proposed changes cover the stages of a project from project planning, during development, and through deployment. They also cover maintenance.

This was originally written for a client, currently using Trac and Subversion, looking to improve their development process.
I decided to share it since other teams might find it useful.

The areas to improve include *Task Prioritization*, *Estimating Project Duration*, *Tracking Project Status*, *Maintaining Code Quality*, and *Reducing Deployment Time*. Specific process changes are proposed to address each of these, along with tools to support the process changes.

Table 1: Expected Costs

Tool	Monthly	Annual
LiquidPlanner	$290	$3,480
Git	N/A	n/a
GitHub	$25	$300
TravisCI	$129	$1,548
Balsamiq Mockups	$12	$144
Total	$456	$5,472

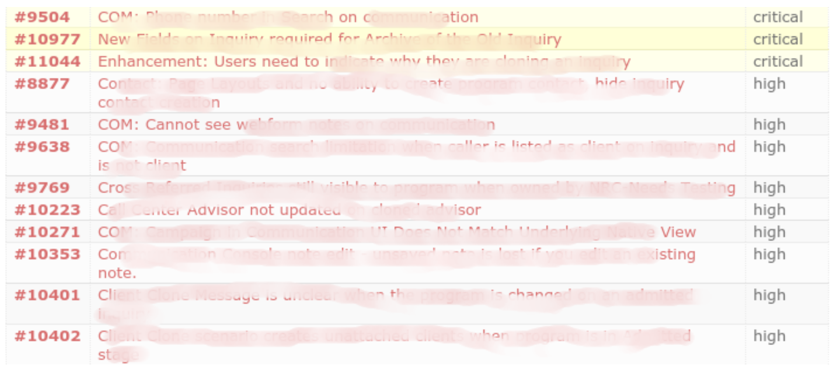
Project Planning

Two process changes are recommended during the planning phase of a project: adopting ordinal priorities for tasks and estimating the work required to complete each task at a granular level.

Task Prioritization

Trac supports a categorical task prioritization scheme. Tasks can be assigned a priority of high, medium, low, etc. The problem with this method of prioritization is that it doesn't clearly identify the precedence of tasks. When deciding which tasks should go into which release, it is important to be able to unambiguously prioritize the tasks. If one of two tasks cannot be completed for a release, it should be clear which has higher priority. During development, each individual should be clear about the next task on which they should be working.

Categorical prioritization provides some guidance, but one tends to end up with a long list of "high priority" tasks.



#9504	COM: [redacted] on c[redacted]	critical
#10977	New [redacted]	critical
#11044	Enhancement: Users need to i[redacted]ry	critical
#8877	Con[redacted] hide inquiry con[redacted]n	high
#9481	COM: Cannot see w[redacted]n	high
#9638	CO[redacted]nd is [redacted]	high
#9769	Cros[redacted]g	high
#10223	Ca[redacted]nter Advisor not updated [redacted]visor	high
#10271	CO[redacted]e View	high
#10353	Con[redacted]isting note.	high
#10401	Cli[redacted]	high
#10402	Cli[redacted] st[redacted]	high

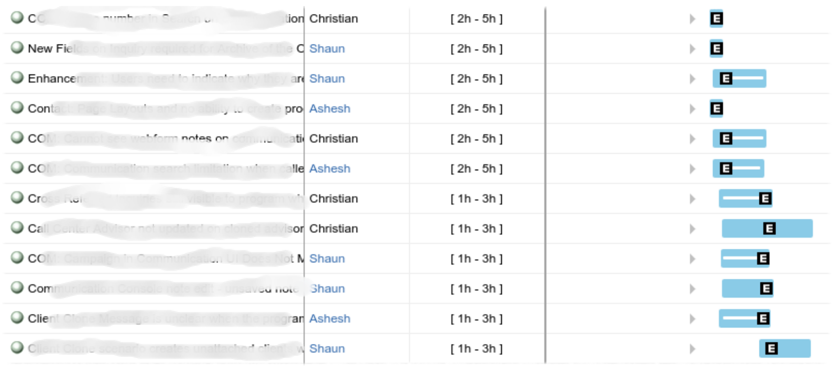
Figure 1: Typical Trac ticket list

USING ORDINAL TASK PRIORITIES simply means that there is always a single highest priority task, then the second highest priority one, etc. The priorities are both unambiguous and easy to change by dragging tasks up and down.

All project management tools that support a kanban-style board for tasks, including LiquidPlanner, implicitly use an ordinal prioritization scheme.

A valuable by-product of ordinal task prioritization is that, when combined with task estimates, a schedule for task completion can be generated automatically.

LIQUIDPLANNER uses ordinal prioritization.



C[redacted]ion	Christian	[2h - 5h]
New Fie[redacted] C	Shaun	[2h - 5h]
Enhance[redacted]	Shaun	[2h - 5h]
Conta[redacted] pro	Ashesh	[2h - 5h]
CO[redacted]	Christian	[2h - 5h]
CO[redacted]lle	Ashesh	[2h - 5h]
Cro[redacted]	Christian	[1h - 3h]
Cal[redacted]sor	Christian	[1h - 3h]
CO[redacted]t M	Shaun	[1h - 3h]
Com[redacted]	Shaun	[1h - 3h]
Clie[redacted]ar	Ashesh	[1h - 3h]
C[redacted]	Shaun	[1h - 3h]

Figure 2: Liquid Planner tasks with estimates and anticipated completion dates

Estimation

Task estimation is essential for planning a project to ensure it comes in on-time and within budget. LiquidPlanner provides a novel approach to task estimation in which both best-case and worst-case estimates [1] are entered for tasks. This allows the uncertainty in the

Estimating is hard, and most people are over-confident. But one can become better at estimating through practice. Douglas Hubbard's How to Measure Anything (`http://amzn.to/1LoFZrA`) is a good resource on estimating. The author's company, Hubbard Decision Research, also offers a three-hour calibration course for about $600 per person.

[1] `http://www.liquidplanner.com/support/articles/estimating-in-ranges/`

estimate to be captured. This range is used as a confidence interval, and a statistically correct timeline [2] for the project is generated based on the individual task estimates.

[2] http://www.liquidplanner.com/support/articles/reading-the-liquidplanner-schedule-bars/

With estimates on the assigned tasks for a project, the workload of each of the developers can be reviewed, and tasks can be reassigned if necessary.

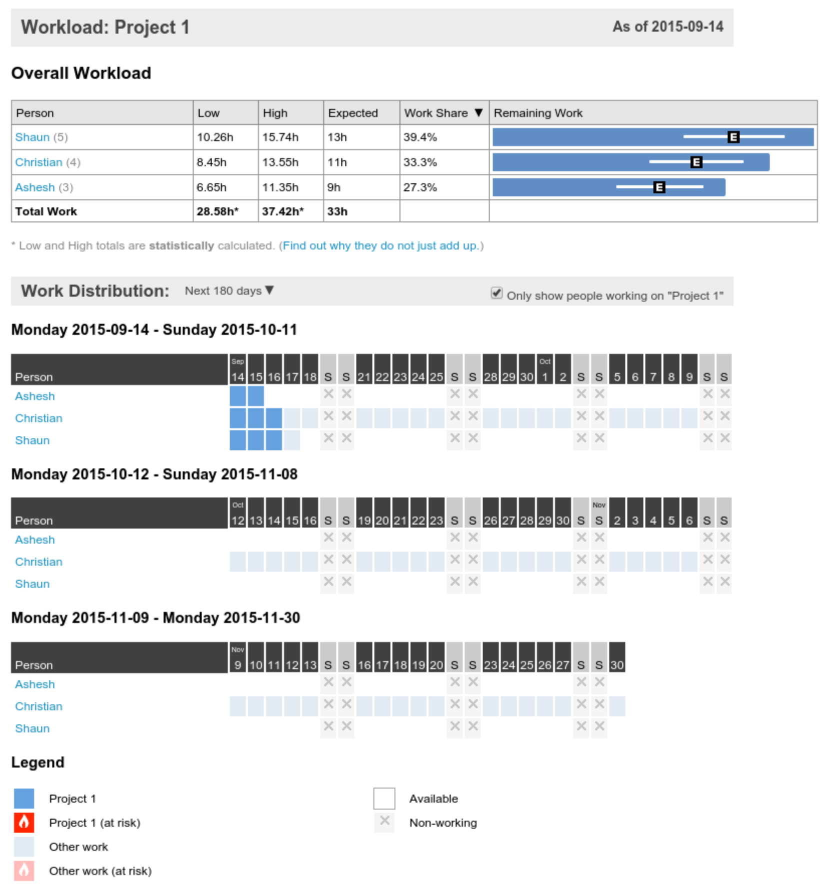


Figure 3: Workload in LiquidPlanner

Workload: Project 1 — As of 2015-09-14

Overall Workload

Person	Low	High	Expected	Work Share ▼	Remaining Work
Shaun (5)	10.26h	15.74h	13h	39.4%	
Christian (4)	8.45h	13.55h	11h	33.3%	
Ashesh (3)	6.65h	11.35h	9h	27.3%	
Total Work	28.58h*	37.42h*	33h		

* Low and High totals are **statistically** calculated. (Find out why they do not just add up.)

Work Distribution: Next 180 days ▼ ☑ Only show people working on "Project 1"

Monday 2015-09-14 - Sunday 2015-10-11

Monday 2015-10-12 - Sunday 2015-11-08

Monday 2015-11-09 - Monday 2015-11-30

Legend

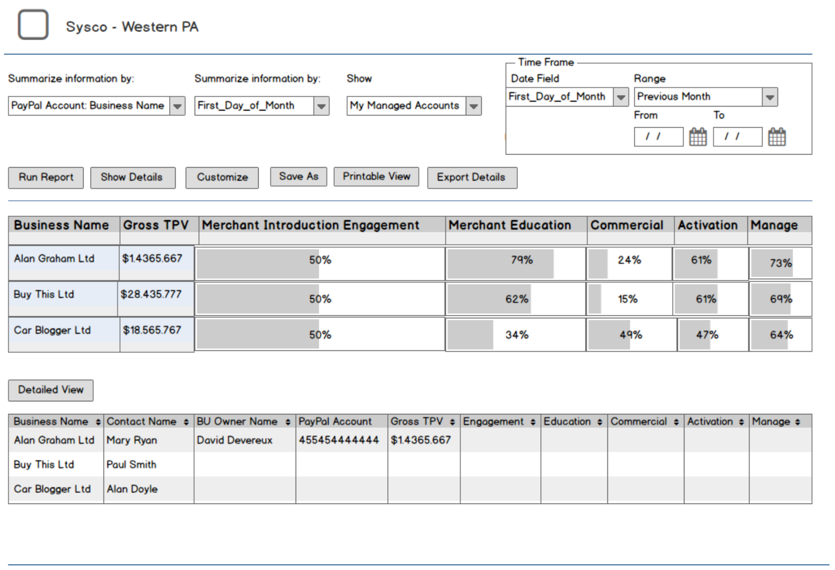
🟦	Project 1	☐	Available
🟥	Project 1 (at risk)	✕	Non-working
▨	Other work		
▨	Other work (at risk)		

Requirements Gathering

Task estimation is not very valuable if the work being estimated does not accurately reflect the requirements of the project. It is necessary for the requirements to provide sufficient detail so that the developer can provide accurate estimates. (Alternatively, estimates with a very wide confidence interval can be provided, but a project plan that estimates project duration being between one month and three years, for example, is generally not acceptable to the project sponsor.)

Depending on the scope of the work that needs to be done, different ways of providing requirements may work. For large, new features or significant user interface redesigns, mockups [3] can be invaluable.

[3] Balsamiq Mockups is a good, inexpensive tools for mockups



Sysco - Western PA

Summarize information by: PayPal Account: Business Name

Summarize information by: First_Day_of_Month

Show: My Managed Accounts

Time Frame
Date Field: First_Day_of_Month
Range: Previous Month
From: / /
To: / /

Business Name	Gross TPV	Merchant Introduction Engagement	Merchant Education	Commercial	Activation	Manage
Alan Graham Ltd	$1.4365.667	50%	79%	24%	61%	73%
Buy This Ltd	$28.435.777	50%	62%	15%	61%	69%
Car Blogger Ltd	$18.565.767	50%	34%	49%	47%	64%

Business Name	Contact Name	BU Owner Name	PayPal Account	Gross TPV	Engagement	Education	Commercial	Activation	Manage
Alan Graham Ltd	Mary Ryan	David Devereux	455454444444	$1.4365.667					
Buy This Ltd	Paul Smith								
Car Blogger Ltd	Alan Doyle								

Figure 4: Sample Balsamiq mockup

THE IMPORTANT THING is that developers end up with requirements of sufficient detail so as to be able to break down the work into tasks with acceptably narrow confidence intervals. Generally, this means tasks that are between two and eight hours long.

The recommendations in this document do not provide a full solution for perfecting the requirements gathering process, but once adopted should help illustrate where further improvements could be made.

For example, it might be worthwhile to have a someone with user interface or technical writing expertise be part of a project to help ensure that new features are intuitive to users. That said, a more frequent release schedule made possible by some of these changes will support a more iterative approach to development in which user feedback can be incorporated into future development relatively quickly.

During Development

Throughout the duration of a project, it is important that the project manager and the whole team are in sync on the project status.

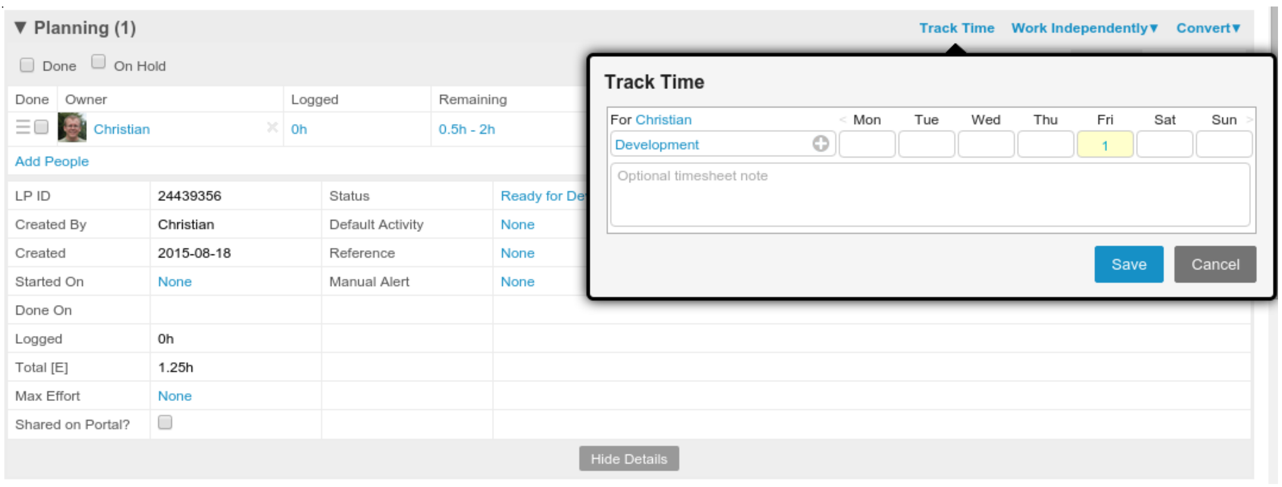
Tracking Project Status

As development progresses, it should be easy to track the remaining amount of work remaining. Developers must adopt a process of keeping their tasks up to date with the amount of remaining work. LiquidPlanner makes this easy with a built-in time tracker, which automatically updates the remaining time estimates as time is recorded.

THE SCHEDULE IS UPDATED AUTOMATICALLY as each task's estimate is updated.

If time passes without any updates, the schedule also updates automatically, pushing the expected completion date into the future.

If a schedule starts to slip, tasks that are at risk of not being completed by the deadline are flagged in red. The tasks in the critical



▼ Planning (1)

Done · On Hold

Done	Owner	Logged	Remaining
	Christian	0h	0.5h - 2h

Add People

LP ID	24439356	Status	Ready for De
Created By	Christian	Default Activity	None
Created	2015-08-18	Reference	None
Started On	None	Manual Alert	None
Done On			
Logged	0h		
Total [E]	1.25h		
Max Effort	None		
Shared on Portal?	☐		

Track Time

For Christian	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Development					1		

Optional timesheet note

Save · Cancel

Figure 5: Recording Time in LiquidPlanner

path can be highlighted, and the problems can be addressed early, while there's still time for mitigation.

Kanban

In addition to the standard view of tasks with anticipated completion dates, LiquidPlanner also provides a kanban-style board. This view is helpful in clearly identifying what each person is working on, what's next up to be worked on, whether any tasks are getting held up moving through the development pipeline, or whether too much work-in-progress is piling up.

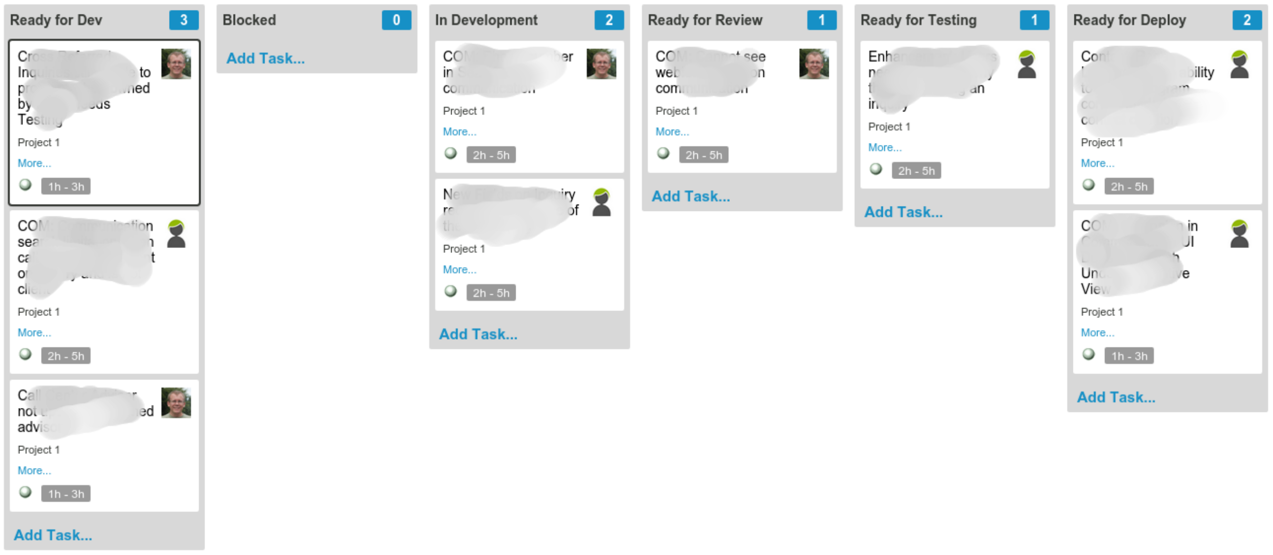


Figure 6: Card View in LiquidPlanner

Limit Work-in-Progress

Work-in-progress describes a task on which work has started, but which has not made it through development, code review, and testing yet. To make the development process more efficient and minimize delivery risk, it is important to minimize work-in-progress. Because we have prioritized the most important work to be done first, we want to ensure that this work makes it through the development pipeline, and it ready to be deployed. We do not want work piling up, waiting to be code reviewed or tested.

Maintaining Code Quality

In addition to optimizing the processes for managing tasks, the time spent developing new features, fixing bugs, and getting these changes into production can be reduced through improved tooling and processes as well.

The overall goal is to keep release branches stable with an understandable history.

Develop on Feature Branches

The first step is do all development on feature branches. The work for each task should be developed on a new branch. Git makes it quick and easy to create branches, and track which branches have been merged.

Pre-Merge Code Review

The next step is to do code reviews on feature branches, prior to merging. This keeps the branch onto which the feature branch will be merged stable, and allows updates made during the code review process to be made on the feature branch.

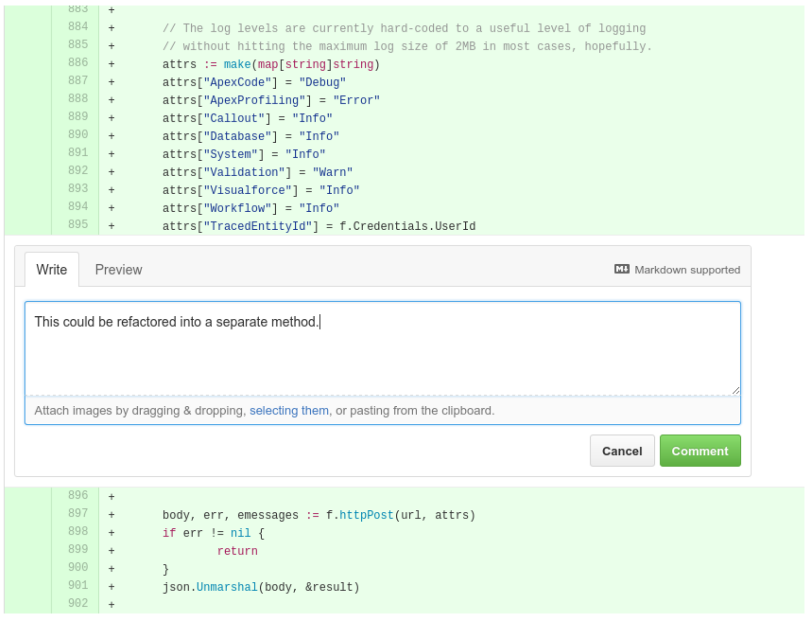
GITHUB ALSO SUPPORTS INLINE COMMENTS on commits, which make code review comments clearer and more concise.

Automated Testing

We can also automate some of the code review process with automated testing. When using git and GitHub for development, once development has been completed for a task, the developer opens a pull request [4].

[4] `https://help.github.com/articles/using-pull-requests/`

TRAVIS CI IS A CONTINUOUS INTEGRATION TOOL that integrates with GitHub which allows us to have tests run automatically when a



```
+
+       // The log levels are currently hard-coded to a useful level of logging
+       // without hitting the maximum log size of 2MB in most cases, hopefully.
+       attrs := make(map[string]string)
+       attrs["ApexCode"] = "Debug"
+       attrs["ApexProfiling"] = "Error"
+       attrs["Callout"] = "Info"
+       attrs["Database"] = "Info"
+       attrs["System"] = "Info"
+       attrs["Validation"] = "Warn"
+       attrs["Visualforce"] = "Info"
+       attrs["Workflow"] = "Info"
+       attrs["TracedEntityId"] = f.Credentials.UserId
```

This could be refactored into a separate method.

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

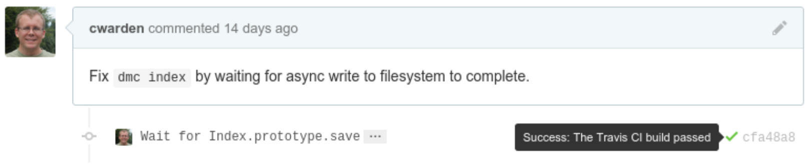
```
+
+       body, err, emessages := f.httpPost(url, attrs)
+       if err != nil {
+               return
+       }
+       json.Unmarshal(body, &result)
+
```

Figure 7: Inline Comment Example

pull request is opened. The version of the code on the feature branch is deployed to testing org using the `checkOnly` flag [5] and all of the tests are run.

[5] `http://sforce.co/1OQXkwu`

The status of the test run is automatically displayed within GitHub with a link to the full test results.



cwarden commented 14 days ago

Fix `dmc index` by waiting for async write to filesystem to complete.

`Wait for Index.prototype.save …`

Success: The Travis CI build passed ✓ cfa48a8

Figure 8: Pull Request after tests pass successfully in Travis CI

Commit Clean History

Finally, the commits on the feature branch can be squashed prior to merging. This allows for a clean history, which eases subsequent maintenance when one needs to figure out why changes were made to the code and identify the related task.

Git provides features for rewriting history that are very useful `http://bit.ly/1OQUaZB`), but can cause headaches if not used correctly. Fortunately, GitHub offers a protected branches feature to mitigate the most common problem associated with rewriting history (`http://bit.ly/1OQUV4O`)

Deployment Process

Limiting work-in-progress, keeping code changes for work-in-progress on feature branches, doing pre-merge code reviews, and automating testing all work to simplify the deployment process.

It is still necessary to perform manual, functional testing to ensure that the changes made meet the requirements, but our release branches are almost always in a deployable state, to be deployed to a QA org for full end-to-end testing or to production for hot-fixes.

Separate Patch Branch

This new model also makes it easier to maintain one branch for mainline development on the next major release and a separate branch for any urgent bug fixes to the last release.

By convention, the mainline branch is called ‘trunk‘ in subversion, and ‘master‘ in git.