Bugflow Docs Documentation

Release 0.0.0

Chris Gough

March 27, 2016

Contents

Contents:

CHAPTER 1

Vision

I use GitHub tickets extensively to manage development with a distributed multidiciplinary team.

My aspiration is a tool that:

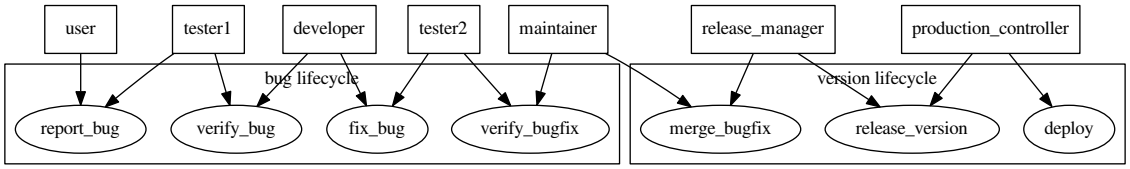
- automates some of the work I do in the ticket system
- streamlines ticket creation tasks, and improves the quality of information captured on initial ticekt creation
- allows me to have different processes for different projects, and supports easy continuous improvement of processes in a project

Bugflow is my name for that tool.

CHAPTER 2

Concepts

Imagine the following "bugflow" made sense for your project. The point of bugflow is to support arbitrary distributed, multi-diciplinary, ticket-based development processes, so this is only an example for demonstrating concepts.



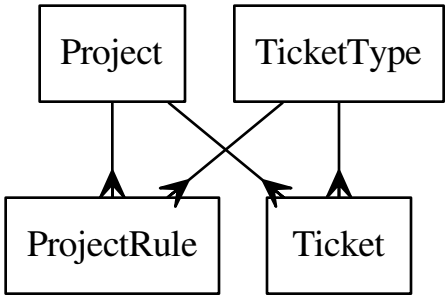
Let's ignore most of the "version livecycle" for now and focus on the "bug lifecycle". We can do this because it's just more of the same, just one of the endless variety of development processes that we might want to support. We only need to focus on one to learn the concepts.

The first thing to notice is that this diagram looks like an over-simplification. The diagram implies that "verify bugfix" never sends the job back to the developer with "it's still not fixed". It implies a developer doesn't recieve a "verified bug" from a tester and tell them "no, it's better like this, I refuse to change it the way you are asking becuase it will make the product worse". These sorts of things are all real and healthy kinds of conversation.

So, what is bugflow?

Bugflow is: * a configuration model for defining project rules * a set of interfaces for streamlining tasks * a swarm of artificially intelligent ticket grooming agents that work for freeon your project

The configuration model is the conceptual heart of the system, so let's start there.



This logical ER diagram means:

- Projects are configured with Project Rules
- Project Rules are about TicketTypes
- Tickets belong to a Proeject
- Tickets are of a TicketType

Projects and Tickets are those things that exist in Github.

TicketTypes are a convention. They mean "Tickets that carry a specific tag". In bugflow, we create and manage some special ticket tags. You can have other tags too, and bugflow will ignore them. You can manipulate tags on tickets however you like, and BugFlow will follow it's configuration rules. This may or may not be a problem depending on the collective stupidity (of you and your rules).

The ProjectRules are actual logical rules. Yes, like prolog. No, you don't have to write prolog! (in fact there is no prolog anywhere in the system, who mentioned prolog?).

When you configure your rules, you essentially make a collection of "if FOO and BAR, then do BAZ" type statement with a modern leak-proof rat-free user-friendly and developer-empathetic rule specification component. I can say that with confidence because I haven't built it yet.

There are a few different types of rules, a bunch of different actions (BAZ), and a bunch of different conditions (FOOs and BARs) that should all be very familiar to you if you use a ticket system to manage a distributed, multi-diciplinary team. More about that in subsequent chapters (that I haven't written yet).

The types of rules are:

- questions that need to be asked (using a dynamic form)
- behaviors that change tickets (using the GitHub API)

The conditional logic acts on:

- information about tickets that can be obtained through the API
- answers to questions, provided through a dynamic form

If you are familiar with python pyke, you might see where I am heading with this.