Designing a Workflow Engine Database

exceptionnotfound.net/designing-a-workflow-engine-database-part-1-introduction-and-purpose

April 22, 2015

Our company was noticing that a lot of our internal processes were relatively similar: one person requests that something happen (e.g. a purchase, an approval to start building something, etc.), then a series of people approve that request until, finally, the item has been reviewed and approved by anyone that needed to review it.

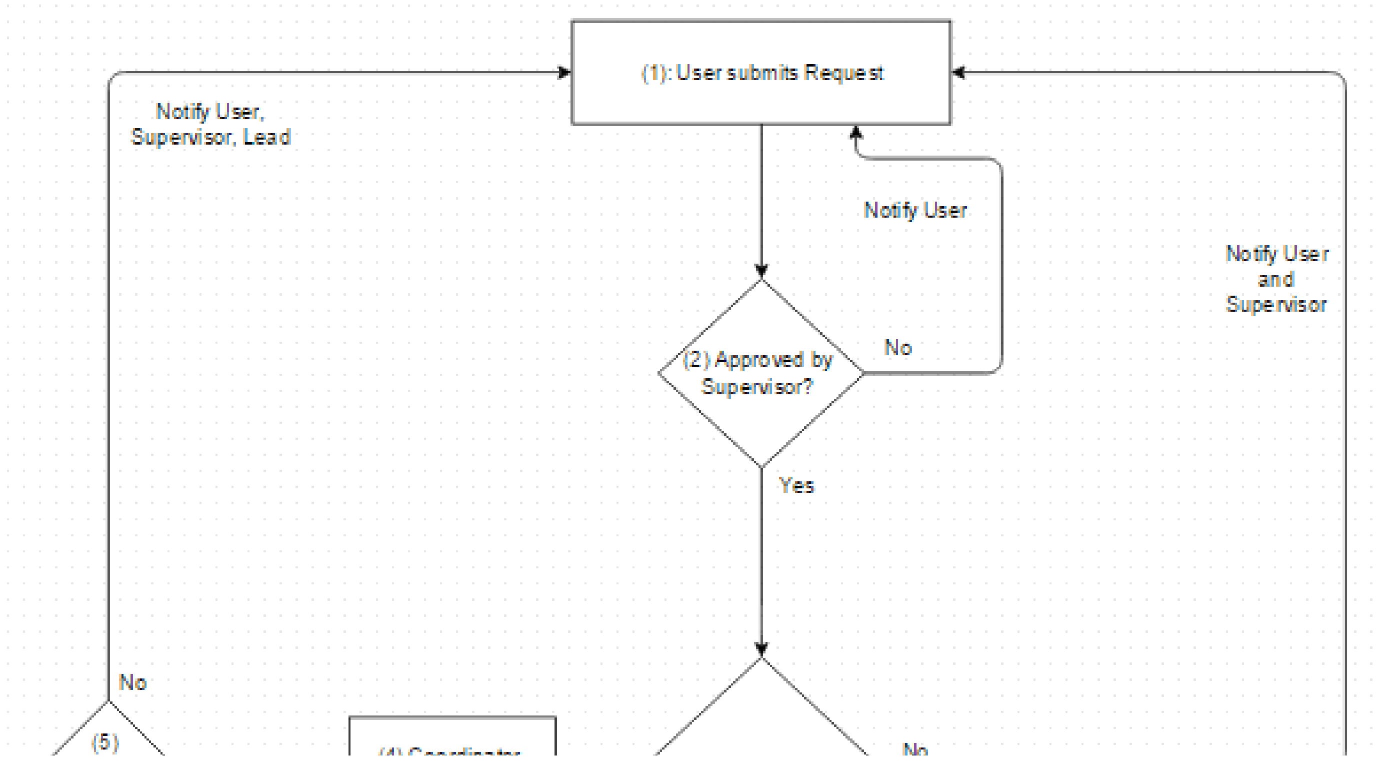
Our team (particularly our management group, who have to deal with these very similar applications nearly every day) started thinking that, since we had so many approvals processes that were so similar, could we build a database-driven tool to manage all of them?

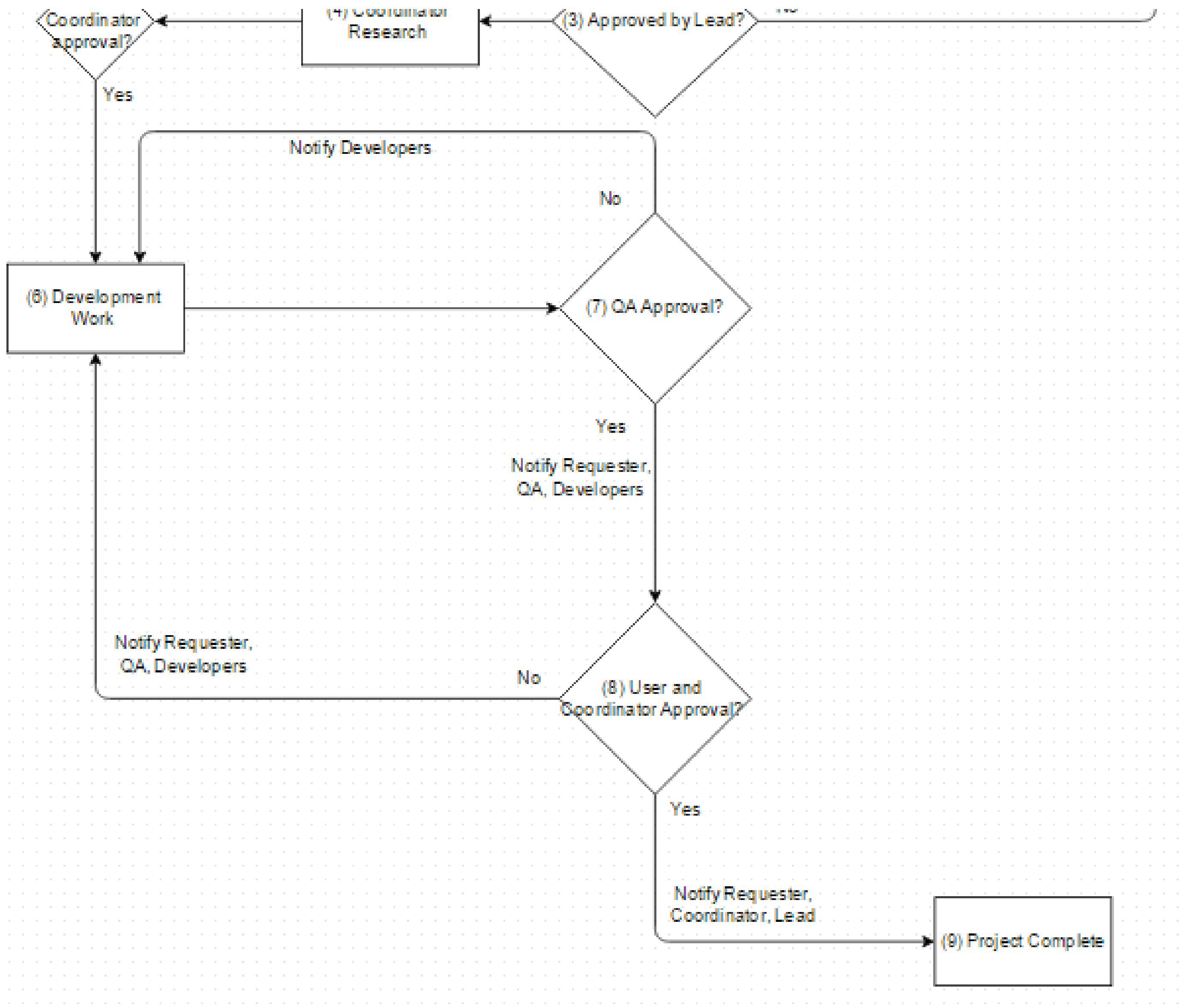
I was assigned to investigate these similarities, and in the process of doing such I realized that these processes were essentially finite-state machines which had a Request in a given State at any particular time, and which also defined how to travel from one state to another (Transitions) and what needed to happen in order to invoke that travel (Actions).

Upon further research, what I discovered was that these collections of Requests, States, Transitions, Actions, and so on had a name: they were Workflows.

What is a Workflow?

A Workflow is a series of decisions made by different people that determines what happens to a particular request that one of those people made, according to a defined and repeatable process. An example of this process is shown in this flowchart:


(1): User submits Request
Notify User, Supervisor, Lead
Notify User
Notify User and Supervisor
(2) Approved by Supervisor?
No
Yes
No
(5)
No



Example Steps in a Workflow

Let's illustrate that flow chart another way, by enumerating the steps:

1. John Doe submits a request that says "I need a site that would allow me to have my employees log their time off so I can quickly review it."
2. John submits that request to his supervisor Victoria, who agrees with him and approves the request.
3. Victoria's approval sends the request to Nate, a lead developer, who looks over the initial request and decides that there is a workable idea here that he and his team can implement as a website. He approves the request.
4. Nate's approval sends the request to Jenna, the coordinator, who starts conducting research (probably involving John, Victoria, and Nate) to determine how much effort this request would require.
5. If Jenna approves the request, it is assigned back to Nate with orders to begin

development.
6. Nate and his team do the development work. This step could take a variable amount of time.
7. Once development work is done, the request is sent to the QA team for approval, lead by Charles.
8. If Charles and his team find no testing issues, he approves the project and sends it to John for final approval (essentially all of teams are together asking John, "this is what you wanted, right?").
9. If John approves the project, it is marked complete, and no more action can be taken against it.

Lots of Similarity

We had many different kinds of workflows in our organization, and the management group wanted us programmers to determine if we could build a generic, database-driven service that could represent many (if not all) of these processes in one central database.

After reviewing several of the currently-existing workflows, my team and I discovered that they all had similar components:

- A **Request object** that is reviewed, approved, or implemented by various people.
- A **set of highly-variable data** that was associated to each Request.
- A **series of decisions (called a Process)** that determine who was next going to review the Request. These decisions must always be submitted to the engine by an individual person, but that individual could be a person related to the Request (e.g. the Requester or a stakeholder) or one of a group of people.
- A **set of notifications** that could go to various groups of people.
- A small number of people that were **in charge of the Process itself**.

In this series, we're going to walk through the database design of our Workflow app and show each part of the solution was implemented, and finally how they were all wired together. We're going to do this in eight parts (this post is Part 1):

This isn't the only design that can implement Workflows, and it has been simplified from our actual implemented design. But I believe that putting my design ideas down on paper (as it were) will help both me understand my design better and, hopefully, any other budding DBAs out there when they are trying to design a database for their own non-trivial problems.

IMPORTANT NOTE: I am not a DBA, and this design has not been thoroughly reviewed by someone who is. DBAs of the world, let me know how I can improve this design in the comments! All suggestions are welcome!

One final note: in this series we're only concerned with showing how to design this engine, not actually implement it in code. What this means is that we won't cover certain details, such as:

- How the Users get into our database OR
- How the Emails are actually sent to recipients.

With those caveats out of the way, come along with me as we explore how to design a Workflow Engine database, starting with the Process Table and Users!