Chapter 2

Detailed Problem Overview

1. Problem Knowledge

Within this section, context is given to the problem by means of providing domain knowledge, which is necessary before describing the problem itself. Reusable code is discussed, as well as the use of repositories to store such code.

Code Reuse

Writing reusable code is the act of developing (usually) modular code with two goals in mind: how it fits in to the current project, and how it can be used in future projects.

Therefore, code reuse is specifically using existing code to produce new software, and reusability is the indicator of how likely it is that a section of code can be reused [1].

Following the mind-set of reusable code allows for stable subsystems to be used as the foundations on which more complex systems can be built on top, allowing them to develop faster [4].

Ideal reusable code would have already been developed and tested for accuracy and completeness, allowing the developer to trust in the code and not need to re-develop or test their own version of this code [7]. Therefore, software reuse can improve on the final quality of the software, as well as the developer’s productivity.

*[I want to write more here] – problems, benefits, how companies use it, how individuals use it. Contextualise soft eng, the way code can be written and never used again. A brief history of code reuse – when did it start (1980 – fell away a bit, until the start of modularised code – object orientated programming, etc).*

A Brief History of Code Reuse

Talk about SVN and Git here, introduce how we got to them so they can e referred to later on. (Data on how popular Git is, how popular SVN is, how many servers/users/projects/commits per day etc – How many people don’t use it?)

It is generally understood that code reuse has been around since programming began: Programmers have been swapping code for as long as there was code to swap; but research into the field can be mostly traced to Douglas McIlroy in 1968, and his proposal for the software industry to be based on reusable components [5, 6].

Modern day reuse environments have a focus on repurposing existing software assets, and writing or creating those assets to be as reusable as possible. These assets extend further than just code, and include models, requirements, designs and tests [7], or they can be as simple as README files.

Cross Project Code Reuse

Collect evidence or citation for this section. Survey? Show there’s a problem (how easy is it to use/to find/how often do you use…/why not more?). Use questions to get requirements for the system. (Which system do you normally go to? Why? Any suggestions to improve systems like that?) – potentially go back after system is in place and show them, observe, let them use while doing specific tasks (6/8 people). Get more ideas for improvement (perfect loop). Evaluation!!

Software developers, notably those that work on smaller day-to-day projects such as web development, are often faced with repeatedly writing similar or identical code when beginning new projects, or creating congruent modules. Furthermore, developers often have resources they wish to access and use regularly, such as normalise.css in web development (for forcing the same default behaviour between all modern browsers). A typical example would be for web developers to use specific code at the start of a project to normalise the default behaviour between all modern browsers.

Despite this commonality, some developers continue to write the same code, wasting development time and effort on each occasion they reproduce this code.

Others store this code in files on their local machine or in a cloud service, often categorising snippets by use of named folders. This code then remains static, un-shareable and not available for peer review.

With the ever rapid advancements in software development and individual language evolutions, code stored in this way is prone to going stale and obsolete. Code repositories aim to manage or mitigate this problem.

Code Repositories

Code repositories are databases tasked with the management of source code, and can be modelled in a variety of ways, such as relational or object-oriented [8].

These repositories are often project orientated, such as with an SVN or Git, where source code is uploaded in entirety and act as a version control for, or snapshots of, a project.

Other repositories are used to store modularised code for reusing in later projects. This paper focuses more on these types of repositories, specifically when using them for short, cross-project, recurrent snippets.

1. Problem Description

Include data to back up this section as well. Find evidence of who does and doesn’t reuse code for a number of projects. Internal/External reuse.

Increase this section and talk more about each point.

When looking at these cross-project repositories, a number of issues stand out in relation to how the developers use and interact with the repositories and the reusable code itself. The first and foremost is when developers do not reuse code at all, and instead opt to continually rewrite it each time.

Next, then, are personal repositories: A user may write a piece of code and store it for reuse, but this code may not be reviewed in the future, leading to stale code. Furthermore, there is no visibility of the code to peers, removing the possibility of peer review or improvement.

The ability to effectively search and sort within the repository is key to its effectiveness, and often personal repositories do not have adequate features for this. This dampens the possibility of finding code even without knowing it exists in the repository.

Finally, the issue of the repository evolving needs to be addressed. As languages evolve, so too must the repository to adapt to the needs of the developer. Often this priority takes a backseat and the repository itself may become inefficient. Therefore, we have identified five problems with developers, code reuse and the repositories itself.

1. The developer may not reuse at all, and so waste time *rewriting code*.
2. The developer may not update code in line with language advancements, leading to *stale code*.
3. The reusable code may have a lack of *peer review* in a personal or limited use repository.
4. The repository may not employ effective searching and sorting methods, so the developer may not find the code they require.
5. *Maintaining/modifying the repository* itself in response to the evolving needs of the software developer(s).

Similar Solutions:

Google Code (Discontinued?)  
GitHub (Single project orientated, stores code, not designed for reuse)  
Atom.js snippet storage (No focus on collaboration, peer review or social code. Personal repo)

Moocs for reuse – classcentral (reuse in search bar)

Ideas:

* Files as simple as readme files - complex as Java and .NET components
* “Searchable repositories of software metadata and use history.”
* “Programmers have been swapping code for as long as software has existed. What's often lacking are procedures, disciplines and tools for tracking, managing, searching and distributing software assets.”
* “…development tools and environments, version-control software, tools for wrapping or transforming legacy code, and messaging tools that can access reusable code where it sits.”
* “Indeed, although the practice is called "software reuse," much more than code can be carried in reuse libraries. Assets can include things such as business-process rules, best practices, interface specifications, test cases, images, documentation, models, patterns, XML schemas and code at all levels -- virtually anything that can be placed in a file.”  
  (<http://www.computerworld.com/article/2571102/app-development/code-reuse-gets-easier.html>) - July 2003 – Gary Anthes, Computerworld