**Project Proposal**

1. **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).

1. **A Brief History of Code Reuse**

It is generally understood that code reuse has been around since programming began, 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].

1. **Software Developers and Code Reuse**

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 may have resources they wish to commonly use and refer to, such as normalise.css in web development (for forcing the same default behaviour between all modern browsers).

Despite this common use case, many developers continue to write the same code, wasting from minutes to hours of development each time they reproduce this code. Others store this code in files on their local machine or in a cloud service, categorising by use of named folders. While this means they are able to use the code, it remains static, un-shareable and not available for peer review. Therefore, with the ever rapid advancements in software development and individual language evolutions and new customs, the stored code may go stale and obsolete.

1. **Problems with Code Reuse**

As covered in the previous section, there are two main problems with current code reuse:

1. Wasting time *rewriting code*.
2. Not updating code in line with language advancements, leading to *stale code*.

There are also two further, subtle issues: [consider rewriting – repo exclusive, on top of those two, issues relating to x y z]

1. *Maintaining/modifying the repository* itself in response to the evolving needs of the software developer(s).
2. Lack of *peer review* of code in personal repository.

As languages evolve, so too do their practices, how they are thought-of and how they are used. As software development on a whole evolves, the repository must evolve with it. This task is often overlooked, or seen simply as a chore by those single developers that maintain them. Over classification within repositories can remove fluidity, making it difficult to maintain and adapt throughout its lifetime. Therefore, this is a final problem that needs addressing. [Change]

It is ‘promising because complex systems evolve faster if they are built upon stable subsystems. Empirical studies have also concluded that software reuse can improve both the quality and productivity of software development.’ (<http://l3d.cs.colorado.edu/~gerhard/papers/icsr6-2000.pdf)>.

[1] Frakes, W.B. and Kyo Kang, (2005), "Software Reuse Research: Status and Future", IEEE Transactions on Software Engineering, 31(7), July, pp. 529-536  
( <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1492369> )

[2] [An evolutionary approach to constructing effective software reuse repositories](http://dl.acm.org/citation.cfm?id=248233.248242&coll=DL&dl=ACM&CFID=720184950&CFTOKEN=38469218) (1997) – Reference for modifying repo as requirements evolve  
(<http://delivery.acm.org/10.1145/250000/248242/p111-henninger.pdf?ip=138.38.246.174&id=248242&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E85B475708465C551%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=720184950&CFTOKEN=38469218&__acm__=1444390582_f852ec952f1d3635685b03f4819bbbde>)

[3] Role and Relevance of Reuse Repository Facilitating Software Development

(2014) – Reference for how reuse repo’s affect development and their role and relevance (<http://delivery.acm.org/10.1145/2640000/2632445/p36a-rathi.pdf?ip=138.38.246.174&id=2632445&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E85B475708465C551%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=720184950&CFTOKEN=38469218&__acm__=1444390676_e06011e7531fb1ab65abb3f5e3180d86)>

[4] Promoting Reuse with Active Reuse Repository Systems (2000) Reference for helping developers navigate a repository and find components to reuse that they may not have known existed  
(<http://l3d.cs.colorado.edu/~gerhard/papers/icsr6-2000.pdf)>

[5] Doug McIlroy – Conference sponsored by NATO

McIlroy, Malcolm Douglas (January 1969). ["Mass produced software components"](http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF) (PDF). *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968*. Scientific Affairs Division, NATO. p. 79.

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>

[6] Jacobson, I., Griss, M. and Jonsson, P. Software Reuse: Architecture, Process and Organization for Business Success. ACM Press, New York, NY, 1997.

[7] From Local to Global Coordination: Lessons from Software Reuse - Rebecca E. Grinter

<http://delivery.acm.org/10.1145/510000/500309/p144-grinter.pdf?ip=138.38.165.211&id=500309&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E85B475708465C551%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=720184950&CFTOKEN=38469218&__acm__=1445334655_6f3823adccd64d29c4759d22f567ee58>

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