Vision and Scope Document

for

Snip

Version 1.0 approved

Prepared by Steven Atkinson

Personal Project

9/15/2004

Copyright © 2004 by Steven Atkinson.

Table of Contents

Revision History

Name	Date	Reason For Changes
Steven Atkinson	08/13/03	First draft.
Steven Atkinson	09/15/04	Revamped to clean it up. Maybe I will start it now!

1. Business Requirements

The Snip project is a personal project used to store code snippets in any computer programming language, to classify those snippets according to many different classification schemes and to allow the user of the system to create, update and search for code snippets.

The project is to be used as a vehicle to investigate a number of different implementation technologies, to provide experience with this wider range of technologies and potentially to be used as a basis from which to make simple comparisons of the technologies. Proposed technologies include a C++/COM/WindowsAPI thick client, a Java/JSP/J2EE web application using Tomcat/Jboss, and a suite of command-line client scripts (PERL, Python).

The ultimate value this project will provide is a flexible, classifiable, searchable and modifiable collection of code snippets, accessible using a wide variety of technologies.

1.1. Background

Over the last few years, it has become apparent as a professional software developer that knowledge leaks from grey matter. To save some key techniques becomes useful in future efforts. Traditional snippet collections can provide some useful pieces of debugging code, an algorithm here or there or some small piece of useful functionality. However the effort involved in searching for that functionality often is larger than writing the snippet from first principles, rendering many code snippet collections a frustrating experience for the moderately experienced programmer.

The advent of design patterns has also somewhat deviated the discussion of software engineering into a design and architecture realm. This realm is useful, but to the practicing programmer, code-level snippets for small, defined tasks are also eye-opening, educational and can also be used to "fill in the template methods" of design patterns.

Previous work on software libraries[1] and classification scheme research is also providing a technical background for classification and search scheme development.

1.2. Business Opportunity

Snip is a code snippet management system. The problem these systems try to solve is to share knowledge of how to solve problems using software code. Many popular software development environments have support for code snippets in the form of code templates that can often be included into source code under development. These template-instantiation features are certainly useful for development; however there is no self-organization of those templates, which are usaully hand-classified and only accessible for the lifetime of the project, not beyond. There is a need for a persistent code snippet library from which to easily and quickly select code snippets in a meaningful and useful manner to the practising software engineer.

The market for code snippets is limited to developers who are interested in learning new techniques (browsers) or developers who have a very specific need (searchers). Other potential users include quality assurance professionals, and perhaps there will be a role for code snippet maintenance.

---

[1] Formal Engineering of Software Library Systems Steven Atkinson. Ph.D. thesis, submitted to *The School of Information Technology* , at *The University of Queensland* , October 1, 1997.

This market is very small; the key to a code snippet utility is the ability to easily find interesting snippets. Microsoft has a Code Librarian Viewer utility that supports a notion of a code library without editing, importing or rearrangement of code snippets. Snip will support the creation, modification and searching for code snippets in multiple ways. Another competitor is snippet.org, which stores many different snippets from different languages in a classified hierarchy. The entry criteria for snippets into snippets.org is less restrictive than Snip. Snip will have defined standard rules to ensure that most snippets have passed a strict set of requirements, in a similar way to requirements for design patterns (well defined, seen multiple times).

Snip will solve the problem of searching for useful software by ensuring that only quality snippets enter the collection and that those snippets can be found using a number of different classification techniques. The notion of a classification technique will be generalized to allow for additional techniques to be defined dynamically. In doing so, Snip will help overcome a large problem in software development – it will finally be easier to search for quality code than to re-invent that code from scratch.

1.3. Business Objectives and Success Criteria

Snip will provide value to the business through improved developer knowledge sharing. Shared knowledge contained in a code snippet library is persistent over time, reducing startup costs for nely hired engineers. Feature development time could be decreased by reusing code techniques used earlier.

The success of the Snip project will be determined by two factors: firstly, how successful can people be using Snip to find code of interest, and secondly, how easily can they take advantage of code found in their new context?

A number of factors can affect the success of the Snip project

- *Feasibility of formulating search queries in different ways that effectively find software snippets*
- *Feasibility to define and generalize the notion of a classification scheme*
- *Time available for task completion*
- *Overhead of learning new technologies*

Snip can be said to be successful if the following criteria are met:

- *Snip can allow for the creation/import of code snippets with a notion of quality control in place*
- *Snip supports searching for code snippets using a sensible classification scheme*
- *Snip can allow for editing of code snippets once they are in the library*
- *Snip has been implemented using at least 2 different software development languages*
- *Snip has been implemented with both web and thick-client interfaces*

1.4. Customer or Market Needs

Existing Customer Needs

Software Developers need to find reusable code for a number of purposes:

- *to expand technical knowledge*
- *to solve small idiomatic problems at hand*

Good software developers also often contribute code to professional forums, and/or open source projects. Additionally, many software professionals contribute articles to magazines and newsletters.

In summary, to search for and contribute code and explanatory snippets of information can be a useful tool in a software developer's process.

Currently many developers meet these needs in a number of ways:

- *performing Google searches through reference materials*
- *using and contributing professional development / forum web site subscriptions*
- *using and contributing pre-existing code from their past experience*
- *communicating with more experienced software development team members*

Unmet Customer Needs

All the above methods of creating and finding code snippets can be slow and yield imperfect results. The time taken to find a useful piece of code may be less than the time it takes to find a book and write the code from scratch.

Snip provides an amalgamation of these services within classification schemes that allow more specific searches more readily Snip reduces the time to find useful information, and promotes sharing of knowledge amongst developers.

Customers would be able to not only flexibly search, but also submit snippets for review and inclusion.

Operating Environment

Customers can access Snip using a thick-client web services approach, and also through a personalized web application. Ultimately Snip should offer its services as a live web service so it can be integrated into IDEs and other applications.

.

Critical Requirements

1. *Ability to search for snippets of information using multiple classification schemes*
2. *Ability to find useful information quickly without too many searches*
3. *Ability to submit a code snippet for review and inclusion into the library*

1.5. Business Risks

The major risk is that the searching mechanism is of little practical use, rendering the system interesting but not commonly accepted amongst the user base. To mitigate this risk, one can use freely available search engines to at minimum provide reasonable-quality keyword-based searches.

This project is for the purposes of validating some of the ideas in the field of sotware reuse, and so failure of users to accept the system may indicate infeasibility of the goals or poorly structured systems.

2. Vision of the Solution

2.1. Vision Statement

Snip is a searchable, shareable software library that can assist the daily software development process. Snip provides the ability to carry personal code collection with you wherever you go to develop software, and to allow you to share snippets amongst co-developers to increase corporate engineering knowledge.

2.2. Major Features

Code snippets are fragments or complete sections of code, used to demonstrate a technique that may be useful to other professionals using the same tools. Snippets may be dependent upon other snippets, third-party libraries, external hardware, software architectures and other environmental factors as yet unknown. These dependencies are considered part of a snippet.

1. *Ability to define a model for code snippets*
    a. *Ability to extend the code snippet model*
2. *Ability to insert code snippets*
    a. *Ability to certify code snippets*
    b. *Ability to add code snippet to the store (multiple classification schemes)*
3. *Ability to search for code snippets*
    a. *Ability to search using multiple classification schemes*
        i. *Ability to query using different query types*
    b. *Ability to search extended snippet models seamlessly*
4. *Ability to edit code snippets*
5. *Ability to browse code snippets by categories*

2.3. Assumptions and Dependencies

Snip depends upon Java technology for an initial implementation..
Snip depends on the generic classification scheme defined in my thesis.

3. Scope and Limitations

3.1. Scope of Initial Release

The initial set of features to be implemented are enough to define the product as a proof-of-concept vehicle to validate the snippet search mechanism. The ability to search for snippets using multiple classification schemes is the most important initial feature and addresses the major product risk directly.

To enable search for code snippets, the initial feature set of the Snip project must contain those features designed to model, store and search for snippets.

1. *Ability to store code snippets*
    a. *Ability to extend the code snippet model*
2. *Ability to search for code snippets*
    b. *Ability to support multiple classification schemes*
    c. *Ability to search extended snippet models seamlessly*
3. *Ability to browse code snippets by categories*

3.2. Scope of Subsequent Releases

Secondary features to be implemented later are the ability to insert snippets into the library using a set of certification / acceptance criteria, and the ability to edit existing snippets in the library Both these features are important from a cohesive snippet library perspective, but are secondary to the major purpose of having a library – the ability to search for information..

3.3. Limitations and Exclusions

Snip will not store arbitrary pieces of code; the intent is for there to be entrance criteria.

4. Business Context

4.1. Stakeholder Profiles

Stakeholder	Major Value	Attitudes	Major Interests	Constraints
Junior software developer	*improved productivity*	*Want to write their own code; product can help them learn without overhead*	*Learning new techniques from others; storing cool tricks they develop*	*Should be very simple to initially use*
Senior software developer	*reduced rework*	*Receptive but demand much of the search system*	*Like to remember their own techniques; eager to share knowledge*	*Must integrate well with existing toolsets*
Software Project Manager	*automation of previously manual tasks*	*Snip sharing can automate knowledge transfer, reduce need for meetings*	*Team cohesiveness, better quality of software output*	*Must be a small part of a larger development toolset*
Executive	*improved productivity*	*Sees a happier work environment*	*Audit/Review*	

4.2. Project Priorities

Dimension	Driver (state objective)	Constraint (state limits)	Degree of Freedom (state allowable range)
Schedule	*release 1.0 to be available by 1/31/2004, release 1.1 by 3/31/2004*		
Features	*Store, search and share features must be included in release 1.0*		*Sharing features can be dropped if necessary*

Quality	*Useful tool with sharp clean HTML/GUI interface. Test-based design should be included at the start*		*Tests can be designed as documents without being implemented.*
Staff		*maximum team size is 1 developer for initial release*	
Cost	*To be developed in spare time*	*Spend $500 on software tools for development*	*Don't spend unless really needed.*

4.3. Operating Environment

The environment in which Snip will be used is an interactive software development environment. The tool needs to be available and reliable.

Users of Snip will be geographically dispersed across many time zones.
Each user will have a local snippet store to which they have access at any time.
Snippets enter the system as instigated by any particular user.
Snip users can share selected snippets from their own library with other (selected) Snip users
Sharing of snippets should occur using a secure connection protocol