Technical Communication in Your Field

STEVEN PAUSTIAN, New Mexico Institute of Mining & Technology

TODO [the absract goes here, dummy]

Categories and Subject Descriptors: C.2.2 [**Technical Communication**]: Computer Science

General Terms: Comments, Communication, Issue Tracking, Technical Language, Version Control

Additional Key Words and Phrases: Technical Communication, ACM Small Format

ACM Reference Format:
Steven Paustian 2015. Technical Communication in Your Field *ACM Ex. Journ.* 0, 0, Article 00 (March 2015), 4 pages.
DOI:http://dx.doi.org/00.000/0000000.0000000

1. INTRODUCTION

Computer Science (CS) is the study of computation and its applications. CS majors are employed in a wide variety of positions post-graduation; prominent job titles include Application Developer, Information Technology Architect, and Systems Administrator.[Payscale.com 2015]

Many technical communications in CS are comparable to other disciplines, including peer reviewed journals, memos, and technical reports. In addition, it employs a number of communication tools that are unique. Examples include issue tracking and code commenting.

To further explore technical communication in the field, a recent graduate and a seasoned professional programmer were interviewed. A discussion of these interviews can be found in [2]. Three examples technical documents were gathered and will be discussed in [3]. Two secondary sources regarding technical communication were also found and will be discussed in [4].

2. INTERVIEW ANALYSIS

The following is brief summary of the information collected from the interviewees.

2.1. Professional Programmer

Frank Germano, a professional programmer of 30 years, was a wealth of information regarding technical communication. As a former vice president of a software engineering firm and current lead developer, he noted technical communication requirements may vary widely through one's career.

According to Frank, entry level programmers spend the least time on technical communication. Code commenting, software documentation, and issue tracking (verbose

This work was intended, designed, and written for ENGL341-02, Spring 2015.
Author's address: S. Paustian, Computer Science Department, New Mexico Institute of Mining & Technology, Socorro, NM.

DOI:http://dx.doi.org/00.0000/0000000.0000000

descriptions of buggy code) are some examples of communication by programmers; they rarely interact with lay management.

Lead developers and project managers serve as a bridge between programmers and non-technical management, and their communications reflect this. They tend to handle state diagrams, flowcharts/UML diagrams, memos, project presentations, and (sometimes) contracts. He revealed that larger firms will typically include a Technical Writer in teams working on a significant project. This person's sole responsibility is creating, editing and interpreting technical documention for the software the team is currently writing.

The bulk of a programmer's non-coding time is spent on internal communication, namely emails. As one advances in the corporate structure, more and more of a programmer's time is used in technical communication. Frank spends around four hours a day on e-mails, issue tracking, memos, and software documentation (excluding code comments). He reveals that the amount of time he spends coding is small, but as a vice president, was even smaller.

2.2. Recent Graduate

Jon Zingale is a recent graduate from The University of Texas as a math major with a computer science minor. He was hired by a security firm in Silicon Valley. He reports that a large portion of his time is spent on communication; mostly emails, memos, and documenting security problems both in code and an internal issue registration system.

He is in nearly constant contact with members of his team; they communicate through Skype, e-mails, and team meetings. According the Jon, the ability to quickly communicate technical concepts on a whiteboard are essential to his position. He rarely needs to convey technical information to non-technical personel, though he was recently asked to give a 20 minute presentation to a client regarding security holes discovered by his team.

He expects the amount of time he spends on communication to increase as he advances in his career. He concedes that his communication requirements are different from many of his peers due to the nature of his employment, none of the courses he took as a student were germaine to this aspect of his job.

3. EXAMPLE DOCUMENTS

Three example documents were obtained from Frank and Jon: an example of properly commented code, an example of issue tracking documention, and a monthly memo sent to clients. An analysis of these documents is presented in Table I

Table I. **Analysis of Sample Documents**

Document Type	**Audience**	**Purpose**	**Style**	**Formating**
Issue Tracking	Management tracking progress on a bug fix. Internal programmers assigned to fix the issue.	Provide specific technical and tracking information regarding a bug that needs to be fixed.	Sentences are terse and paragraphs are short. Stars (*) are used to separate sections.	Conforms to internal specifications designed for clear communication of technical information.
Customer Memo	Current customers and the general public. Company employees.	Provide customers and employees with "state of the union" information about the company.	Sentences are narrative and readable. There is an active voice and the present tense is used.	Colorful and easy to read. Different font weights and colors highlight important information and what it relates too. Information is regarding new customers and new software functionality.
Code Comments	Programmers and other employees working directly with the code.	Explicitly explain code parameters, usage, and how the code works.	A variety of special characters are used as no "markup" is allowed. Comments are extremely terse. Whitespace is used to great effect.	Conforms to internal commenting standards.

4. SECONDARY DOCUMENTS

The following academic documents discuss two vital communication tools in computer science: software documention, and team communication. In [4.1] the importance of software documentation is explored. In [4.2], a software development process is discussed which stresses face-to-face communication amongst development team members.

4.1. Comments Are More Important Than Code

Comments inside code are a unique communication tool of programming. Programmers often talk of making code "self-commenting," but this is nearly impossible to do. Code is, by definition, complicated and difficult to understand. Comments and documentation are essential tools for communicating an algorithm's intent, usage, and implementation. In other words, good comments explain how and most importantly, why.[Raskin 2005]

Raskin argues that code comments are not only important, but essential for code to be reliable and maintainable. Enforcing strict commenting guidelines is cumbersome and limits the amount of code a programmer can write in a given unit of time. Raskin argues that despite this, the benefits of rigorous comments far outweight the costs. Rebuilding code with good documentation is far easier than without. In fact, writing any code is far simpler if documention exists. In addition, good documentation provides "background and decision information that cannot be derived from code." [Raskin 2005]

For these reasons, software documention should be a priority for all programmers.

4.2. Agile Development

Agile software development (ASD) is a software development technique that stresses, among other things, effective person to person communication over documents and lengthy response times.[Highsmith and Cockburn 2001]

Written documents and e-mails tend to lead to long response times. For software developement to be effective and responsive, these wait times must be minimized. To effect this change, team collaboration should happen in person using whiteboards and face to face interaction. In addition, a sponsor, customer representative, and user should be embedded within the development team. Without their participation, software projects tend be slow in responding to changes in software requirements.[Highsmith and Cockburn 2001]

Software documentation should be generated as code is written. Pre-planned documentation tends to lead to unresponsive code. In the modern development world, plans tend to go out of date within days of being written. Therefore, priorities must adjust according to changing requirements. To this end, daily customer interaction and end user feedback is preferred over contract negotations and end-of-cycle user input. [Highsmith and Cockburn 2001]

5. CONCLUSION

TODO[fill this out]

ACKNOWLEDGMENTS

The author would like to thank Frank Germano and Jon Zingale for their interviews.

REFERENCES

Jim Highsmith and Alistair Cockburn. 2001. Agile software development: The business of innovation. *Computer* 34, 9 (2001), 120–127.

Payscale.com. 2015. Popular Jobs for Computer Science Majors - College Salary Report. (2015). http://www.payscale.com/college-salary-report-2014/choosing-a-major/popular-jobs-by-major/computer-science-and-math

Jef Raskin. 2005. Comments are more important than code. *ACM Queue* 3, 2 (2005), 64.