CSE471 - Free and Open Source Software

Dayanand V
v_dayanand@cb.amrita.edu

Dept. of Computer Science and Engineering
Amrita School of Engineering
Amrita Vishwa Vidyapeetham

January 19, 2016

Credits

The credits for the contents of this presentation goes to Karl Fogel and his book “Producing Open Source Software”. You are as free to reuse and redistribute the contents of this presentation as I am in using Fogel’s book’s content.

Contents

Introduction

- About 90-95% free software projects fail. Why?

Introduction

- About 90-95% free software projects fail. Why?
  - Unrealistic requirements
  - Vague specifications
  - Poor resource management
  - Insufficient design phases

Introduction

- About 90-95% free software projects fail. Why?
  - Unrealistic requirements
  - Vague specifications
  - Poor resource management
  - Insufficient design phases
- Aren't these the same reasons why closed-source projects fail?

Introduction

- About 90-95% free software projects fail. Why?
  - Unrealistic requirements
  - Vague specifications
  - Poor resource management
  - Insufficient design phases
- Aren't these the same reasons why closed-source projects fail?
  - Yes, and there's more.

Introduction

- About 90-95% free software projects fail. Why?
  - Unrealistic requirements
  - Vague specifications
  - Poor resource management
  - Insufficient design phases
- Aren't these the same reasons why closed-source projects fail?
  - Yes, and there's more.
- Sometimes, the developers do not appreciate the problems unique to open source development.

Introduction

- About 90-95% free software projects fail. Why?
  - Unrealistic requirements
  - Vague specifications
  - Poor resource management
  - Insufficient design phases
- Aren't these the same reasons why closed-source projects fail?
  - Yes, and there's more.
- Sometimes, the developers do not appreciate the problems unique to open source development. Like?

Introduction

- More reasons why open source projects fail.
  1. Expecting hordes of oompa-loompas to magically volunteer for you.
  2. Expecting that releasing the source will be a cure of all its ills.
  3. Skimping on presentation and packaging to develop "more important stuff".
  4. Expecting that the same management practices used for in-house development will work equally well on an open source project.
  5. Failures of "Cultural Navigation".

"Open source does work, but it is most definitely not a panacea. If there's a cautionary tale here, it is that you can't take a dying project, sprinkle it with the magic pixie dust of "open source," and have everything magically work out. Software is hard. The issues aren't that simple." -Jamie Zawinski, The Mozilla Project

Contents

History

- At the dawn of commercial computers, ‘Software’ sounded closer to an accessory than a business asset.

History

- At the dawn of commercial computers, ‘Software’ sounded closer to an accessory than a business asset.
- Customers (scientists, technicians etc) distributed patches to other customers and it was encouraged by the manufacturers.

History

- At the dawn of commercial computers, ‘Software’ sounded closer to an accessory than a business asset.
- Customers (scientists, technicians etc) distributed patches to other customers and it was encouraged by the manufacturers.
- There was no hardware standardization and the manufacturer wanted machine-specific code and knowledge to spread as widely as possible.

History

- At the dawn of commercial computers, ‘Software’ sounded closer to an accessory than a business asset.
- Customers (scientists, technicians etc) distributed patches to other customers and it was encouraged by the manufacturers.
- There was no hardware standardization and the manufacturer wanted machine-specific code and knowledge to spread as widely as possible.
- There was no Internet, and widespread frictionless sharing as we know it today, was not possible.

The Rise of Proprietary Software and Free Software

- The advent of better ‘hardware’ and ’high level’ programming languages.

The Rise of Proprietary Software and Free Software

- The advent of better ‘hardware’ and ’high level’ programming languages.
- Selling software began looking like a good strategy.

The Rise of Proprietary Software and Free Software

- The advent of better ‘hardware’ and ’high level’ programming languages.
- Selling software began looking like a good strategy.
- If the user had freedom to modify the software and improve it, the ‘added value’ patches would be of lesser significance.

The Rise of Proprietary Software and Free Software

- The advent of better ‘hardware’ and ’high level’ programming languages.
- Selling software began looking like a good strategy.
- If the user had freedom to modify the software and improve it, the ‘added value’ patches would be of lesser significance.
- Shared code can reach competitors.

The Rise of Proprietary Software and Free Software

- The advent of better ‘hardware’ and ’high level’ programming languages.
- Selling software began looking like a good strategy.
- If the user had freedom to modify the software and improve it, the ‘added value’ patches would be of lesser significance.
- Shared code can reach competitors.
- Ironically, the Internet was getting off the ground around the same time.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- “Richard Stallman” happened.

  “We did not call our software ”free software”, because that term did not yet exist; but that is what it was. Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.” -Richard Stallman

The Rise of Proprietary Software and Free Software - Conscious Resistance

- “Richard Stallman” happened.

  “We did not call our software ”free software”, because that term did not yet exist; but that is what it was. Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.” -Richard Stallman

- He went ahead and started the GNU Project and Free Software Foundation (FSF).

The Rise of Proprietary Software and Free Software - Conscious Resistance

- “Richard Stallman” happened.

  “We did not call our software ”free software”, because that term did not yet exist; but that is what it was. Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.” -Richard Stallman

- He went ahead and started the GNU Project and Free Software Foundation (FSF).
- GNU was aimed at developing a completely free and open computer operating system and body of application s/w.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- “Richard Stallman” happened.

  “We did not call our software ”free software”, because that term did not yet exist; but that is what it was. Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.” -Richard Stallman

- He went ahead and started the GNU Project and Free Software Foundation (FSF).
- GNU was aimed at developing a completely free and open computer operating system and body of application s/w.
- “The GNU General Public License (GPL) says that the code may be copied and modified without restriction and that both copies and derivative works must be distributed *under the same license as the original with no additional restrictions.*”

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.
- GNU Emacs came into being.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.
- GNU Emacs came into being.
- The compiler collection GCC.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.
- GNU Emacs came into being.
- The compiler collection GCC.
- By 1990s, GNU had produced most parts of a free OS except for the kernel.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.
- GNU Emacs came into being.
- The compiler collection GCC.
- By 1990s, GNU had produced most parts of a free OS except for the kernel.
- Linus Torvalds enters the scene with Linux, a kernel completed with the help of volunteers around the world.

The Rise of Proprietary Software and Free Software - Conscious Resistance

- GNU slowly began gaining ground and was able to produce free replacements for many critical components of an OS.
- GNU Emacs came into being.
- The compiler collection GCC.
- By 1990s, GNU had produced most parts of a free OS except for the kernel.
- Linus Torvalds enters the scene with Linux, a kernel completed with the help of volunteers around the world.
- Becomes GNU Linux, the world's first free and open source operating system.

The Rise of Proprietary Software and Free Software - Accidental Resistance

- Berkeley Software Distribution (BSD) - University of California, Berkeley.

The Rise of Proprietary Software and Free Software - Accidental Resistance

- Berkeley Software Distribution (BSD) - University of California, Berkeley.
- Non ideological practice of free software development.

The Rise of Proprietary Software and Free Software - Accidental Resistance

- Berkeley Software Distribution (BSD) - University of California, Berkeley.
- Non ideological practice of free software development.
- X Window System developed at MIT allowed proprietary extensions over free software, but by itself was free.

The Rise of Proprietary Software and Free Software - Accidental Resistance

- Berkeley Software Distribution (BSD) - University of California, Berkeley.
- Non ideological practice of free software development.
- X Window System developed at MIT allowed proprietary extensions over free software, but by itself was free.
- TEX, developed by Donald Knuth offering free publishing quality typesetting system was released open source with just some naming restrictions on derivatives.

The Rise of Proprietary Software and Free Software - Accidental Resistance

- Berkeley Software Distribution (BSD) - University of California, Berkeley.
- Non ideological practice of free software development.
- X Window System developed at MIT allowed proprietary extensions over free software, but by itself was free.
- TEX, developed by Donald Knuth offering free publishing quality typesetting system was released open source with just some naming restrictions on derivatives.

This presentation is built using 'Beamer' which is an extension of LATEX, built on TEX.

Free Versus Open Source

- Free? As in *free food*?

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.
- Freedom to share and modify the source code for any purpose.

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.
- Freedom to share and modify the source code for any purpose.
- Case study : Battle of browsers (1990s), Netscaps vs IE.

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.
- Freedom to share and modify the source code for any purpose.
- Case study : Battle of browsers (1990s), Netscaps vs IE.
- Free software - Morality vs Profitability.

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.
- Freedom to share and modify the source code for any purpose.
- Case study : Battle of browsers (1990s), Netscaps vs IE.
- Free software - Morality vs Profitability.
- Conflict of opinion.

Free Versus Open Source

- Free? As in *free food*?
- Free as in *freedom*.
- Freedom to share and modify the source code for any purpose.
- Case study : Battle of browsers (1990s), Netscaps vs IE.
- Free software - Morality vs Profitability.
- Conflict of opinion.
- 1998 - the word "open source" is coined by Open Source Initiative (OSI) to disambiguate "free" and aimed at giving 'free software' a good marketing.

The Situation Today

- Free software has become a culture of choice.

The Situation Today

- Free software has become a culture of choice.
- So what persuades all these people to stick together long enough to produce something useful?

The Situation Today

- Free software has become a culture of choice.
- So what persuades all these people to stick together long enough to produce something useful?
- The feeling that their connection to a project, and influence over it, is directly proportional to their contributions.

The Situation Today

- Free software has become a culture of choice.
- So what persuades all these people to stick together long enough to produce something useful?
- The feeling that their connection to a project, and influence over it, is directly proportional to their contributions.
- Clearly, projects with corporate sponsorship and/or salaried developers need to be especially careful in this regard.

Contents

Look Around

"Every good work of software starts by scratching a developer's personal itch."
-Eric Raymond, 'The Cathedral and the Bazaar'.

- A good software results when the programmer has a personal interest in seeing the problem solved.

Look Around

"Every good work of software starts by scratching a developer's personal itch."
-Eric Raymond, 'The Cathedral and the Bazaar'.

- A good software results when the programmer has a personal interest in seeing the problem solved.
- Today, we also have the phenomenon of organizations-for-profit, corporations, governments, non-profits, etc starting large, centrally-conceived open source projects from scratch.

Look Around

"Every good work of software starts by scratching a developer's personal itch."
-Eric Raymond, 'The Cathedral and the Bazaar'.

- A good software results when the programmer has a personal interest in seeing the problem solved.
- Today, we also have the phenomenon of organizations-for-profit, corporations, governments, non-profits, etc starting large, centrally-conceived open source projects from scratch.
- Case Study : Kuali Foundation

Look Around

"Every good work of software starts by scratching a developer's personal itch."
-Eric Raymond, 'The Cathedral and the Bazaar'.

- A good software results when the programmer has a personal interest in seeing the problem solved.
- Today, we also have the phenomenon of organizations-for-profit, corporations, governments, non-profits, etc starting large, centrally-conceived open source projects from scratch.
- Case Study : Kuali Foundation
- Identifying potential problems that are faced by a large community.

Look Around

"Every good work of software starts by scratching a developer's personal itch."
-Eric Raymond, 'The Cathedral and the Bazaar'.

- A good software results when the programmer has a personal interest in seeing the problem solved.
- Today, we also have the phenomenon of organizations-for-profit, corporations, governments, non-profits, etc starting large, centrally-conceived open source projects from scratch.
- Case Study : Kuali Foundation
- Identifying potential problems that are faced by a large community.
- So, first, *look around*! ¡github.com, openhub.net, sourceforge.net, directory.fsf.org¿

Start from What You Have

- Understanding clearly what the project is and what its not.

Start from What You Have

- Understanding clearly what the project is and what its not.
- Need for Documentation : Founders Vs New Comers.

Start from What You Have

- Understanding clearly what the project is and what its not.
- Need for Documentation : Founders Vs New Comers.
- Investments : *hacktivation energy*.

Start from What You Have

- Understanding clearly what the project is and what its not.
- Need for Documentation : Founders Vs New Comers.
- Investments : *hacktivation energy*.

Start from What You Have

- Choose a Good Name.

Start from What You Have

- Choose a Good Name.
  - Gives an idea about what the project does.

Start from What You Have

- Choose a Good Name.
  - Gives an idea about what the project does.
  - Easy to remember.

Start from What You Have

- Choose a Good Name.
    - Gives an idea about what the project does.
    - Easy to remember.
    - Good manners, good legal sense.

Start from What You Have

- Choose a Good Name.
    - Gives an idea about what the project does.
    - Easy to remember.
    - Good manners, good legal sense.
    - Getting top-level domains with forwarding to a central home site.

Start from What You Have

- Choose a Good Name.
    - Gives an idea about what the project does.
    - Easy to remember.
    - Good manners, good legal sense.
    - Getting top-level domains with forwarding to a central home site.
    - Handle availability in Twitter, FB.

Start from What You Have

- Clear mission statement.

Start from What You Have

- Clear mission statement.
- State that the Project is ‘Free’.

Start from What You Have

- Clear mission statement.
- State that the Project is ‘Free’.
- A brief list of features and requirements.

Start from What You Have

- Development Status

Start from What You Have

- Development Status
  - Should reflect reality.

Start from What You Have

- Development Status
    - Should reflect reality.
    - Alpha, Beta releases.

Start from What You Have

- Development Status
  - Should reflect reality.
  - Alpha, Beta releases.
  - Availability for download.

Start from What You Have

- Development Status
  - Should reflect reality.
  - Alpha, Beta releases.
  - Availability for download.
  - Version Control and Bug Tracker Access.

Start from What You Have

- Development Status
  - Should reflect reality.
  - Alpha, Beta releases.
  - Availability for download.
  - Version Control and Bug Tracker Access.
    - GitHub.com - based on Git.

Start from What You Have

- Development Status
  - Should reflect reality.
  - Alpha, Beta releases.
  - Availability for download.
  - Version Control and Bug Tracker Access.
    - GitHub.com - based on Git.
    - Rate of bug filing.

Start from What You Have

- Development Status
  - Should reflect reality.
  - Alpha, Beta releases.
  - Availability for download.
  - Version Control and Bug Tracker Access.
    - GitHub.com - based on Git.
    - Rate of bug filing.
  - Communication Channels
    - Mailing List.

Start from What You Have

- Development Status
    - Should reflect reality.
    - Alpha, Beta releases.
    - Availability for download.
    - Version Control and Bug Tracker Access.
        - GitHub.com - based on Git.
        - Rate of bug filing.
    - Communication Channels
        - Mailing List.
        - Chat room.

Start from What You Have

- Development Status
    - Should reflect reality.
    - Alpha, Beta releases.
    - Availability for download.
    - Version Control and Bug Tracker Access.
        - GitHub.com - based on Git.
        - Rate of bug filing.
    - Communication Channels
        - Mailing List.
        - Chat room.
        - IRC Channel.

Start from What You Have

- Development Status
    - Should reflect reality.
    - Alpha, Beta releases.
    - Availability for download.
    - Version Control and Bug Tracker Access.
        - GitHub.com - based on Git.
        - Rate of bug filing.
    - Communication Channels
        - Mailing List.
        - Chat room.
        - IRC Channel.
    - Developer Guidelines.

Start from What You Have

- Documentation.

Start from What You Have

- Documentation.
  - Minimal criteria:

Start from What You Have

- Documentation.
  - Minimal criteria:
    - Should tell the reader clearly how much technical expertise they're expected to have.

Start from What You Have

- Documentation.
  - Minimal criteria:
    - Should tell the reader clearly how much technical expertise they're expected to have.
    - Tells clearly how to setup, and make sure that they've installed the software correctly.

Start from What You Have

- Documentation.
  - Minimal criteria:
    - Should tell the reader clearly how much technical expertise they're expected to have.
    - Tells clearly how to setup, and make sure that they've installed the software correctly.
    - Give one tutorial-style example of how to do a common task.

Start from What You Have

- Documentation.
  - Minimal criteria:
    - Should tell the reader clearly how much technical expertise they're expected to have.
    - Tells clearly how to setup, and make sure that they've installed the software correctly.
    - Give one tutorial-style example of how to do a common task.
    - Label the areas where the documentation is known to be incomplete.

Start from What You Have

- Documentation.
  - Minimal criteria:
    - Should tell the reader clearly how much technical expertise they're expected to have.
    - Tells clearly how to setup, and make sure that they've installed the software correctly.
    - Give one tutorial-style example of how to do a common task.
    - Label the areas where the documentation is known to be incomplete.
  - Developer documentation.
- Demos, Screenshots, Videos, and Example Output.

Start from What You Have

- Documentation.
    - Minimal criteria:
        - Should tell the reader clearly how much technical expertise they're expected to have.
        - Tells clearly how to setup, and make sure that they've installed the software correctly.
        - Give one tutorial-style example of how to do a common task.
        - Label the areas where the documentation is known to be incomplete.
    - Developer documentation.
- Demos, Screenshots, Videos, and Example Output.
- Hosting.

Choosing a License and Applying It

Choosing a License and Applying It

- The “Do-Anything” licenses (MIT License).

Choosing a License and Applying It

- The “Do-Anything” licenses (MIT License).
  - Asserts nominal copyright (without actually restricting copying).

Choosing a License and Applying It

- The “Do-Anything” licenses (MIT License).
    - Asserts nominal copyright (without actually restricting copying).
    - Code comes with no warranty.

Choosing a License and Applying It

- The “Do-Anything” licenses (MIT License).
  - Asserts nominal copyright (without actually restricting copying).
  - Code comes with no warranty.
- The GNU General Public License (GPL).

Choosing a License and Applying It

- The “Do-Anything” licenses (MIT License).
  - Asserts nominal copyright (without actually restricting copying).
  - Code comes with no warranty.
- The GNU General Public License (GPL).
  - Translates loosely to - “If you use my code, make sure you let others use yours.”

Choosing a License and Applying It

Choosing a License and Applying It

- Applying a license - Steps :

Choosing a License and Applying It

- Applying a license - Steps :
  - State the license clearly on the project's front page.

Choosing a License and Applying It

- Applying a license - Steps :
  - State the license clearly on the project's front page.
  - Put the full license text in a file called COPYING (or LICENSE) included with the source code.

Choosing a License and Applying It

- Applying a license - Steps :
  - State the license clearly on the project's front page.
  - Put the full license text in a file called COPYING (or LICENSE) included with the source code.
  - Put a short notice in a comment at the top of each source file, naming :
    - Copyright date
    - Holder
    - The kind of license
    - Where to find the full text of the license.

Setting the Tone

- Avoid private discussions ($1 \langle n$).

Setting the Tone

- Avoid private discussions ($1 \langle n$).
- Zero tolerance to rudeness.

Setting the Tone

- Avoid private discussions ($1 \langle$ n).
- Zero tolerance to rudeness.
- Practice conspicuous code review.

Setting the Tone

- Avoid private discussions ($1 \langle n$).
- Zero tolerance to rudeness.
- Practice conspicuous code review.
- Be open from day one!

Setting the Tone

- Avoid private discussions ($1 \langle n$).
- Zero tolerance to rudeness.
- Practice conspicuous code review.
- Be open from day one!
- Opening a formerly closed project.

Setting the Tone

- Avoid private discussions $(1 \langle n)$.
- Zero tolerance to rudeness.
- Practice conspicuous code review.
- Be open from day one!
- Opening a formerly closed project.
- Announcing.

Setting the Tone

- Avoid private discussions $(1 \langle n)$.
- Zero tolerance to rudeness.
- Practice conspicuous code review.
- Be open from day one!
- Opening a formerly closed project.
- Announcing.

Contents

Overview

- Website
- Mailing lists / Message forums
- Version control
- Bug tracking
- Real-time chat

Website

- Canned Hosting Services.

Website

- Canned Hosting Services.
- Github.com

Website

- Canned Hosting Services.
- Github.com
- Setting up a Github account.

Website

- Canned Hosting Services.
- Github.com
- Setting up a Github account.
    1. Go to *https://github.com/*

Website

- Canned Hosting Services.
- Github.com
- Setting up a Github account.
  1. Go to *https://github.com/*
  2. Sign up - Get yourself an account.

Website

- Canned Hosting Services.
- Github.com
- Setting up a Github account.
  1. Go to *https://github.com/*
  2. Sign up - Get yourself an account.
  3. Go to *https://guides.github.com/activities/hello-world/*

Website

- Canned Hosting Services.
- Github.com
- Setting up a Github account.
  1. Go to *https://github.com/*
  2. Sign up - Get yourself an account.
  3. Go to *https://guides.github.com/activities/hello-world/*
  4. Learn how to :
     1. Create a ‘repository’.
     2. Open an ‘issue’.
     3. Create a ‘branch’.
     4. Make a ‘commit’.
     5. Open a ‘pull request’.
     6. ‘Merge’ a pull request.