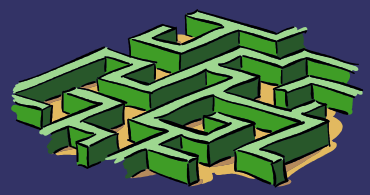
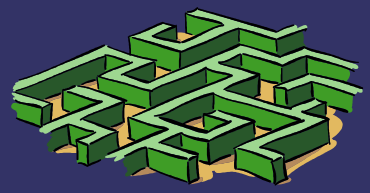
OPEN SOURCE TECHNOLOGY

CS 321



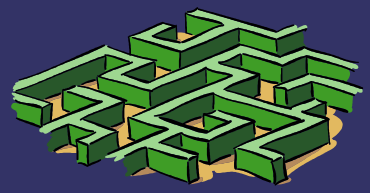
The Good, the Bad and the Ugly of Open Source

- Advantages of using OSS include our ability to:
  - View source code
  - Change and redistribute source code
  - Buy from different vendors and adopt new platform
  - Avoid proprietary information formats
  - Allow integration between products



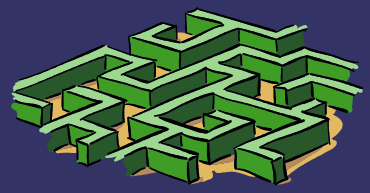
cont..

- Reduce software licensing cost and effort
- Develop and deploy effectively internationally
- Draw from a large pool of skilled professionals



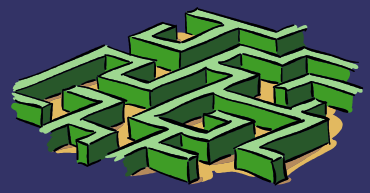
Is Open Source enough by itself?

- Once we have open source product we need to use, we will want to review at least the following elements:
  - Platform portability (hardware, OS)
  - Database portability (stored procedures, APIs)
  - Language
  - Data architecture
  - Software quality (modular, documented)



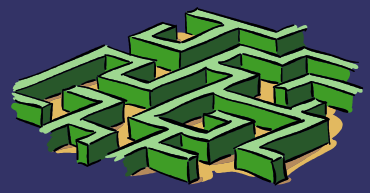
1. Deployment Platform

- Applications have to be deployed on standard, scalable computer hardware available from different vendors and employing internal standard components obtainable in an open market process.
- Applications have to support a range of operating systems.
- As a general rule, if an application is not now available for multiple platforms it is probably not going to be in the future.



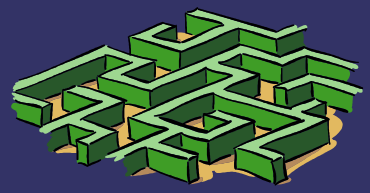
2. Database Platform

- If the application uses a database, we would like it at a minimum to use a standards-based (SQL) database and a consistent, well-designed current interface to that database's APIs and stored procedures.
- Unfortunately, the SQL standard is not sufficient to ensure that an application does not have dependencies on a particular database. We need to recognize the limits of SQL portability and review database design and access strategy to maximize it.



3. Software Language, Architecture, and Implementation

- Application have to be written in an open source language, preferably one that is portable between platforms. This set of languages includes C/C++, Perl, Python, PHP, and UNIX shell. Java is not source language.
- Products need to be specifically reviewed for modularity, since we are looking to see if the system can be adapted to our anticipated needs, and for quality of documentation in the context of our expected users and scenarios of use.

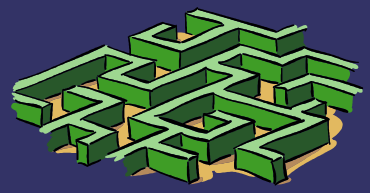


4. Data Architecture

- This allows external access to all data through well-understood standards – for example, SQL or XML – and that this access is documented with examples and is free of restrictions on use.
- Question to ask to test the openness of a system include:
  - For each vendor, what do I do if it goes bankrupt, tries to overcharge me, or ceases to support the product?
  - What will it take to move the system to a platform an order of magnitude larger or smaller?
  - How easily can I obtain a package or build a new application to add or replace functionality?

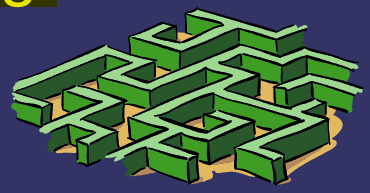
Choosing Open Source

- Open source software does not have the direct sales force, the marketing budgets, or the incentive to bundle product.
- Open source has a less complete level of sales support. What is sold is most a hybrid involving a closed code or some other proprietary solution on top of or alongside open source, and this can pay for expensive of the sale. Examples: WebSphere on Apache, Oracle on Linux, or SAP on Linux and MySQL.



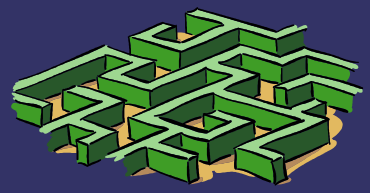
Open Source Products

- Are not bundled, branded or integrated
- Bundling is when several products come packed together
- There is generally some, often limited integration
- Branding is when several products are named similarly so that all can benefit from the reputation of some
- Different open source software products often work well together because they tend to use the same standards, but that is not always communicated by product naming or by any sort of marketing.



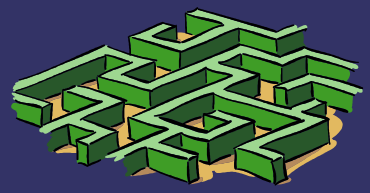
Cont..

- Open source can be harder for customers to integrate, because it is more likely to achieve “working together” by making separate parts work using open standards.
- Integration between database and development tools is generally high. Because source code is available, you should be able to make things work at some effort level.



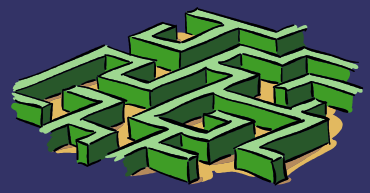
Can vendors make money from Open Source

- Many vendors are making significant revenues from Linux today. Hardware and systems integration are content.
- In 2002, HP reported $2B in 'Linux-related revenue'
- IBM reported $1.5B from Linux in 2002 and expected to report over $3.5B in 2004.
- Among software-only companies, Red Hat is profitable selling only Linux-related products and services, and MySQL is profitable selling only SQL-related products and services.



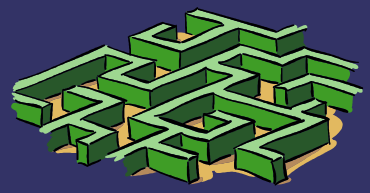
Cont..

- There are many ways to run a software-related business, the most common is to offer a mix of consulting, training, integration, and support services in addition to license or distribution fees. This is a modal of open source companies MySQL and Red Hat.



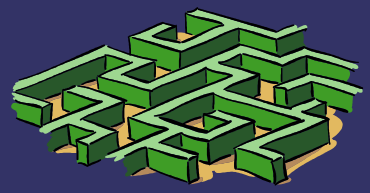
How OSS Is Developed

- Several models that are in use, these are:
  - An individual working largely alone
  - A “bazaar”, or large loosely knit dispersed group
  - A conventional collocated product team
- Most often, a product starts with an individual or a product team and later moves toward the bazaar, often retaining strong central ownership.
- Some OSS has been built on the bazaar model, where a loosely structured, large network of people with little formal organization cooperates.



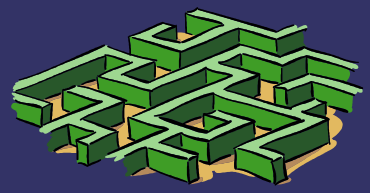
How OSS Is Developed..

- There as to be a seed product first, that can be run, distributed, and tested to get the process going, and that is generally created by an individual. Example all Apache, Linux and BSD were all tightly structured around a small central leadership team.
- First example of the bazaar could be the Berkeley effort to get AT&T files out of BSD, which was led by Eric Bostic in 1990. Others could include Linux, starting from around 1992 and Apache, starting around 1995, led by Brian Behrendorf.



How OSS Is Developed

- Some OSS is built by tightly structured teams that are modeled in a way similar to conventional software development organizations. Examples include Gnome and MySQL.
- Some OSS was built by conventional software companies, which have then converted the product to open source. Examples include OpenOffice, Mozilla, Eclipse, and Firebird SQL.
- Some open source was built by individuals working alone, but is now maintained by a loose team. Examples, which date back to the earliest open source still in use, include Sendmail, GNU Emacs, and C, Samba, and Perl.

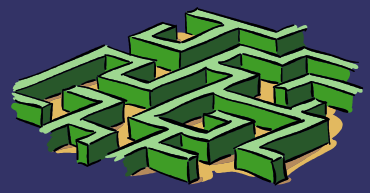


How OSS Is Developed

- The earlier vision and planning phases are not exposed in open source, which only speaks to the development and testing activity compared to closed code.
- Vision and planning are done by an individual or small team, development and testing by a larger one.
- Much OSS is infrastructure, and in infrastructure software (and all matured software) most changes are defined by bug lists, not ideas for new use.
- Every successful open source project has a very modular structure?

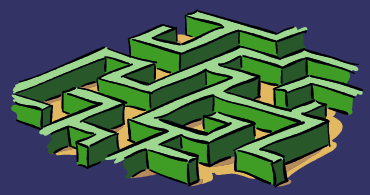
How OSS Is Developed..

- What is unique to open source is that the large pool can see the code, a method known as “with enough eyeballs.” Eric Raymond argues that software that requires high reliability should be open source.
- Comparing open source development to corporate development:
  - Both open source and closed code software developers pay a very high for failure. Corporate developers can fail and live again, and analysis shows that most do fail on measures of time, cost and quality.

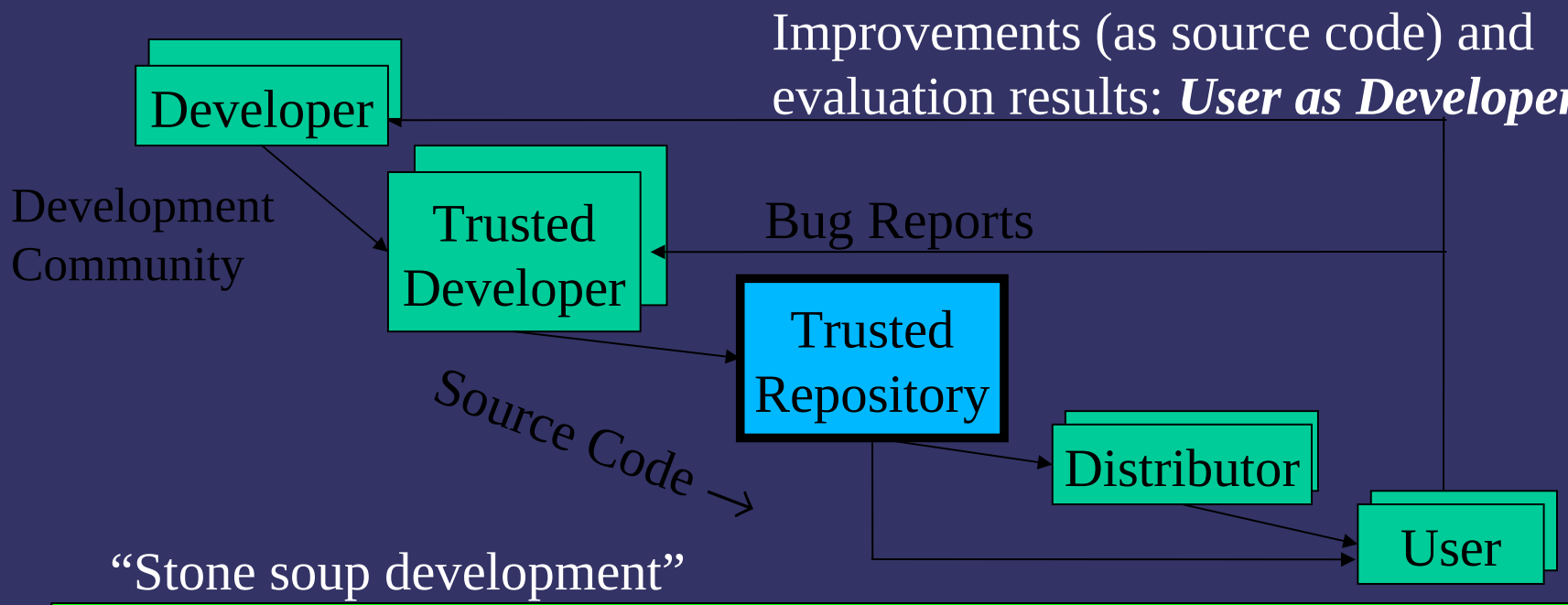


How OSS Is Developed..

- Comparing open source development to corporate development:
  - Corporate development is not usually maintained over the same time frames and with the same levels of staffing and consistency as open source.
  - The software house toolkit is similar to open source (mostly C/C++ with other languages used for peripheral development and front-end scripting), the corporate developers favors proprietary toolkits that have shifted over time, currently usually Java based.



Typical OSS development model

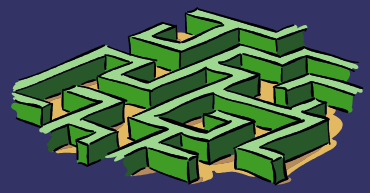


"Stone soup development"

- OSS users typically use software without paying licensing fees
- OSS users typically pay for training & support (competed)
- OSS users are responsible for paying/developing new improvements & any evaluations that they need; often cooperate with others to do so
- Goal: Active development community (like a consortium)

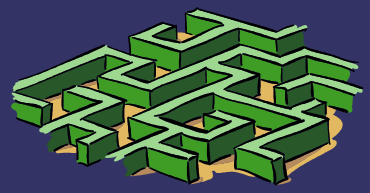
Languages used to develop Open Source Products

- The language used on most projects are:
  - C/C++ 41%
  - Perl 14%
  - Java 13%
  - PHP 11%
  - Shell, SQL, Tcl 8%
  - Python 6%
  - All other 7%
- Most of the successful open source products are written in C/C++.



Languages used to develop Open Source Products..

- **Perl** is a practical, portable language. A major asset of Perl is its database of reusable code, CPAN. Using CPAN, it is possible to for Perl users to quickly find solutions to problems that are new to them but have been met and solved by others. More used that PHP or Python. Its applications include several system administration tools, including Webmin, the very popular SpamAssassin tools.
- **Python** is an object-oriented language that is easy to learn and use. It has extensive libraries available to allow development of pretty much many kind of application, including games, networking programming, etc.



Languages used to develop Open Source Products..

- Python is used to develop complex client applications such as OSAF's Chandler (personal information manager) and Red Hat's Anaconda (installer), as well as server-based applications such as Mailman and BitTorrent.
- **Java** is a popular language, but is used less for open source development. It is less of a general-purpose scripting language with built-in functions for common tasks like Perl or Python, but it does have a strong available set of classes.