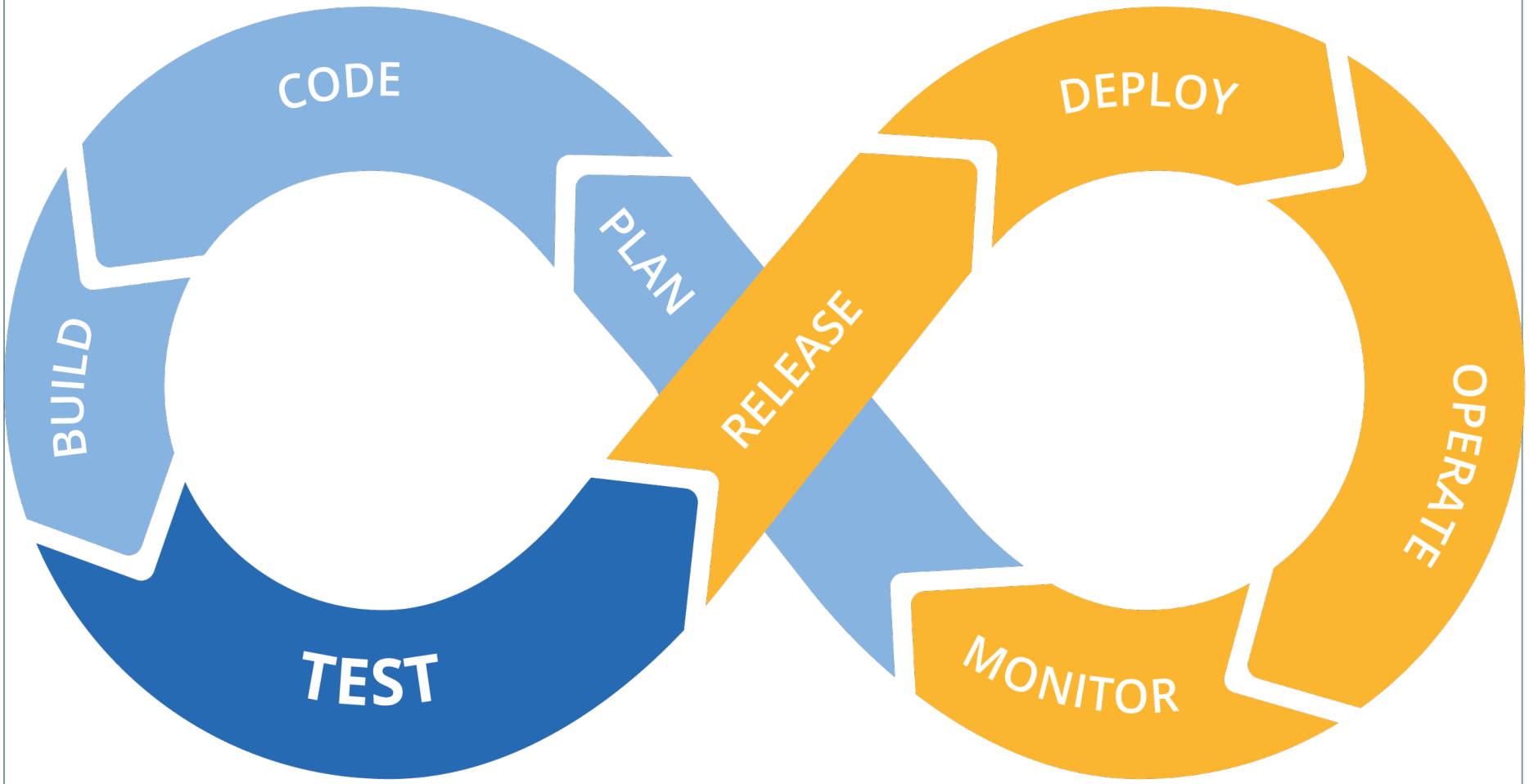
Chương 4: PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

NGUYỄN VĂN HÒA

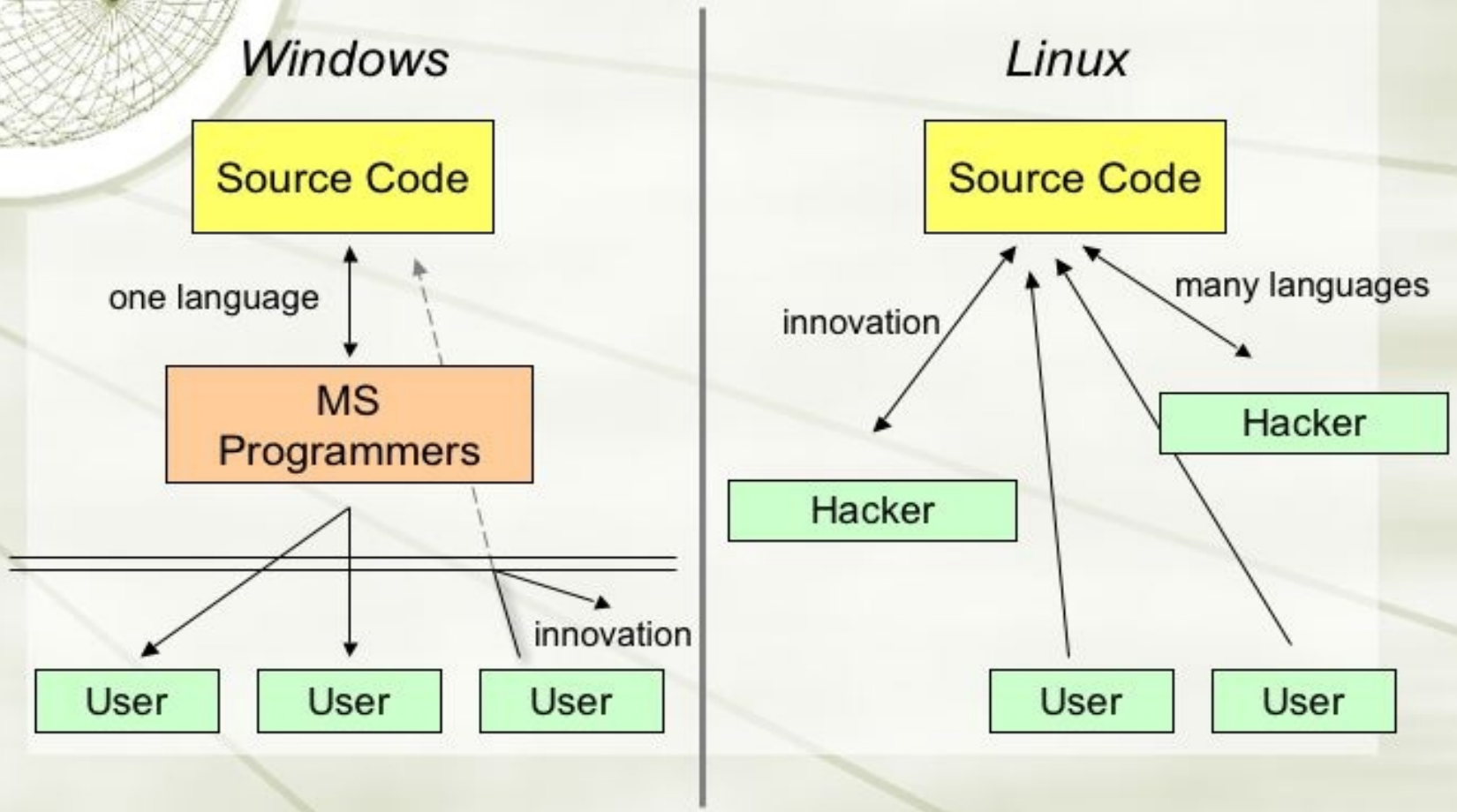
EMAIL: NVHOA@AGU.EDU.VN

Tiến trình phát triển phần mềm

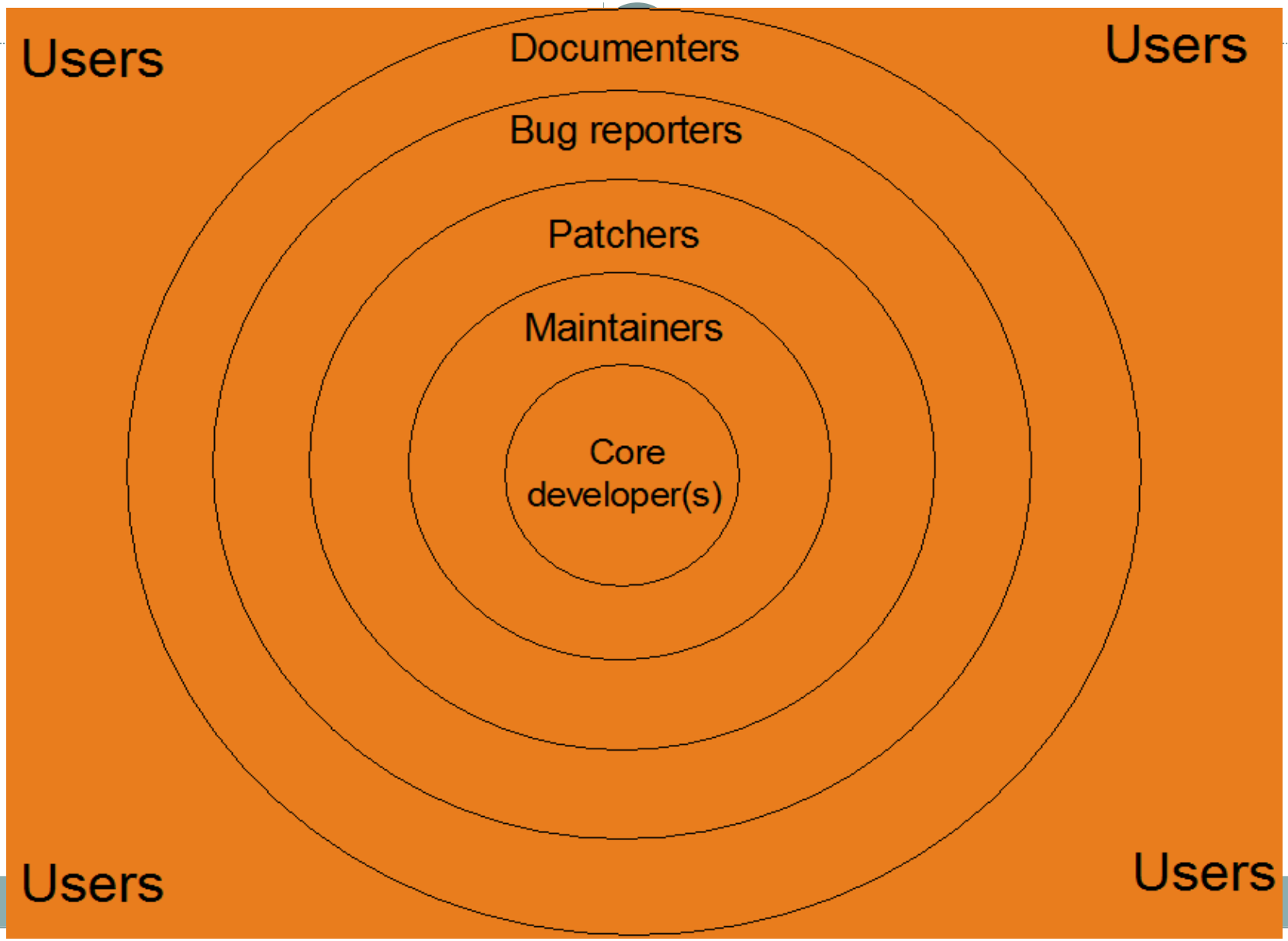


.NET OS

Software: Closed vs Open



OSS Development Process



OSS Development Model

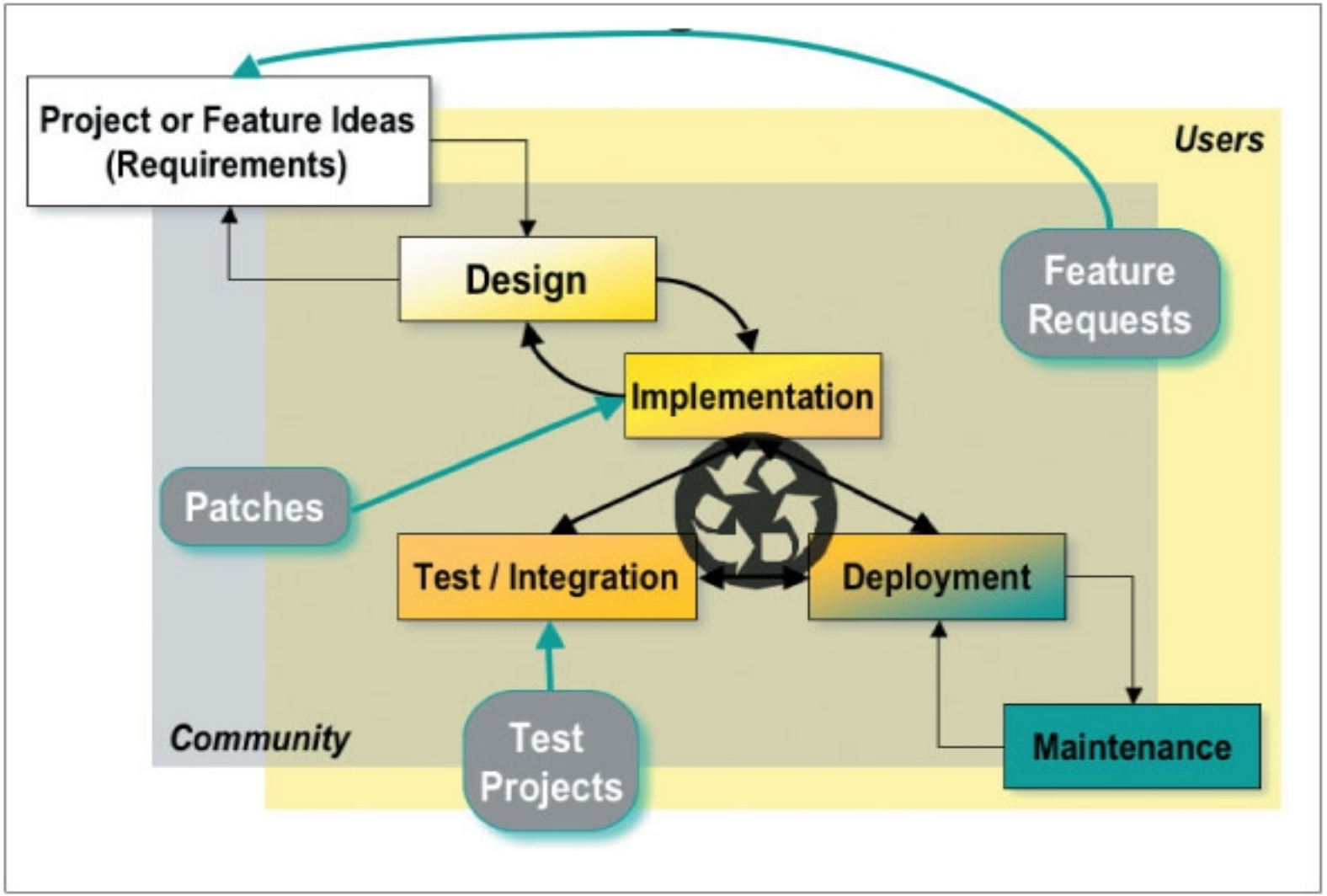
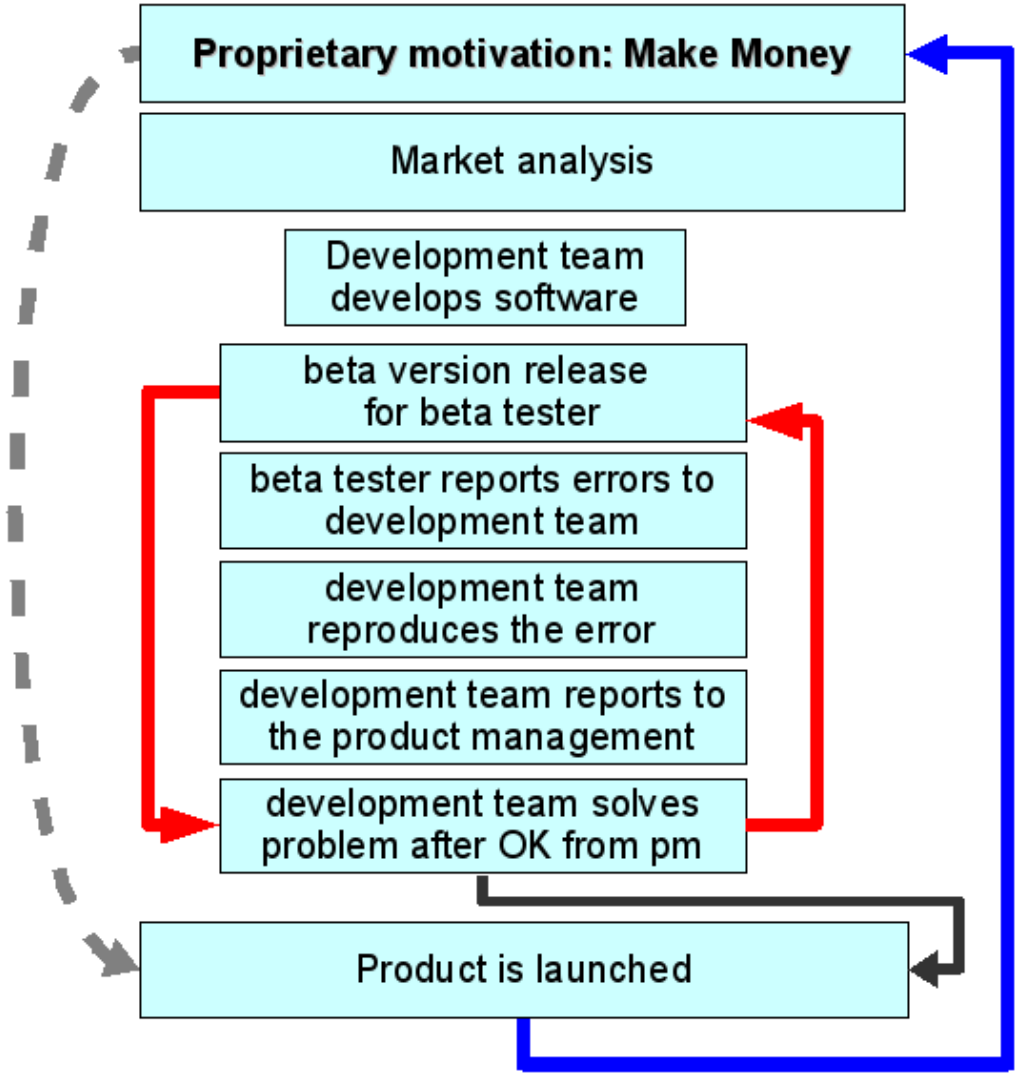
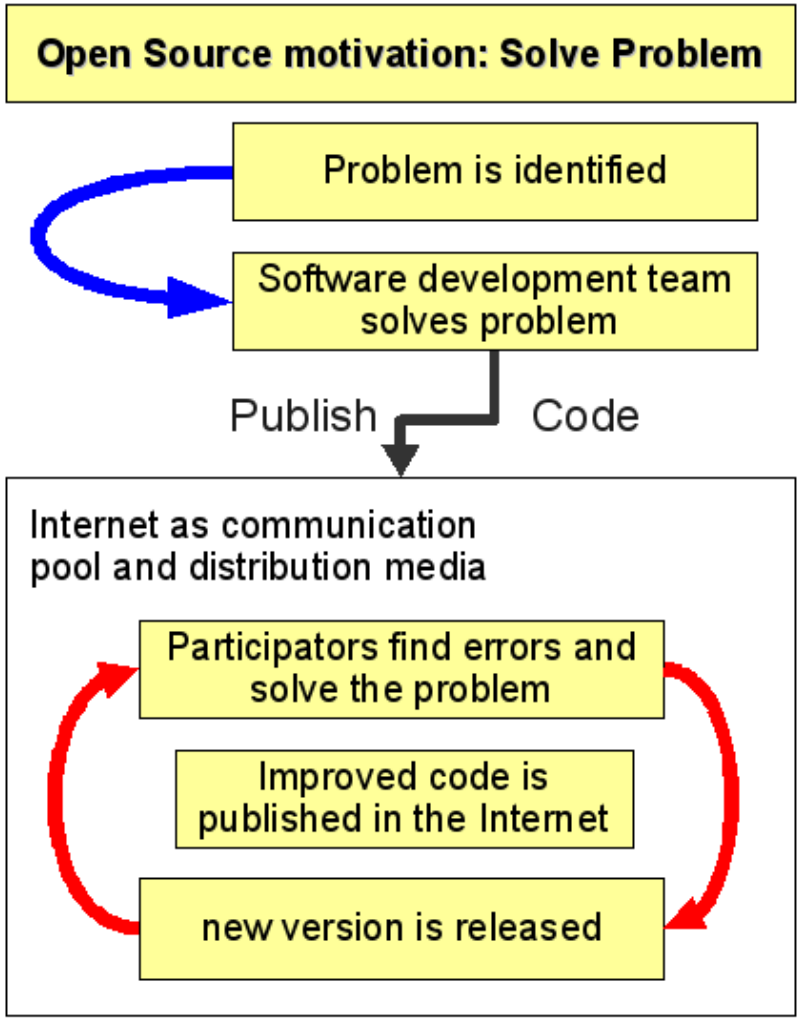


Figure 2: Open source development model

(SOURCE: BILL WEINBERG, OPEN SOURCE DEVELOPMENT LABS, 2006)

Proprietary vs OSS development





Who develop OSS

- Công ty
  - IBM: uses and develops Apache and Linux...
  - Apple: released core layers of Mac OS X Server
  - HP: uses and releases products running Linux
  - Sun: IDE for Java and the Mozilla web browser
- Nhóm nghiên cứu khoa học
  - sourceforge.net; code.google.com
  - Ví dụ: lĩnh vực Tin Sinh học (Bioinformatics)
- Cá nhân

OSSD Informalisms

Email lists	Discussion forums	News postings	Project digests
IM/Internet Relay Chat	Scenarios of usage	How-to guides	To-do lists
FAQ's and item lists	Project Wikis	System documentation	External publications
Copyright licenses	Architecture diagrams	Intra-app scripting	Plug-ins
Code from other projects	Project Web site	Multi-project Web sites	Project source code web
Project repositories	Software bug reports	Issue tracking databases	etc.

Binary Code and Source Code

Binary code

- Machine language (native code)
  - Example of instruction set directly executable by CPU
  - Represented by hexadecimal numbers
- Byte-code
  - Executed by virtual machine
  - Used for Java, etc.

Source code

- Programming language
  - Understandable to people, Modifiable
- Requires conversion to binary code
  - Conversion by compiler, byte-code compiler or interpreter

Biên dịch và thông dịch

- Biên dịch
  - Converts source code to binary code during compiling
  - Advantages of compiled languages
    - Low overhead during execution; high-speed execution
  - Drawback of compiled languages
    - Changes in source code require recompiling
- Thông dịch
  - Source code interpreted at each execution
  - Advantages of interpreted languages
    - No compiling required; easy to create codes
  - Drawbacks of interpreted languages
    - Inferior performance during execution; not suited for large-scale systems

Ngôn ngữ lập trình

- Major compiled languages
  - C language
    - Object-oriented extensions of C
      - C++, Objective-C
    - Conversion from C -> Assembler* -> Machine language
  - FORTRAN, Pascal
    - Programming languages that use byte-code interpreter (Virtual Machine type)
    - Java, C# (.Net)
- Major interpreted languages
  - Perl, PHP, Python, BASIC, LISP, Ruby and many others

Cấp độ phát triển phần mềm

- Network server
  - Web server, mail server, file server, portal site....
- Internet business and enterprise system
  - E-commercial, E-learning
- Hệ thống nhúng
  - PDA, mobile phone,...
- EWS (Engineering Work Station)
- Desktop

Server example

- Web server
  - Apache
- Mail servers
  - MTA
  - ML server
  - POP3/IMAP
- DB server
  - PostgreSQL/MySQL/Fire bird
- DNS
  - BIND
- File shareing
  - Samba/WebDAV
- LDAP
  - OpenLDAP
- Mining server
  - namazu
- CMS
  - XOOPS/Zope
- Business applications
  - E-Learning, e-commere

Developing Languages on Linux

GCC (GNU Compiler Collection)

- Collection of compilers for C, C++, Fortran, Java, etc.
- Standard compiler for development on Unix

Perl

- Strong text processing
- Flexibility to use various syntax for same process
- Frequently used for system management and CGIs

PHP

- HTML-embedded, server-side scripting language
- Main language for Java and Web system development
  - LAMP/LAPP

Python

- Features block designation using indentation

Ruby

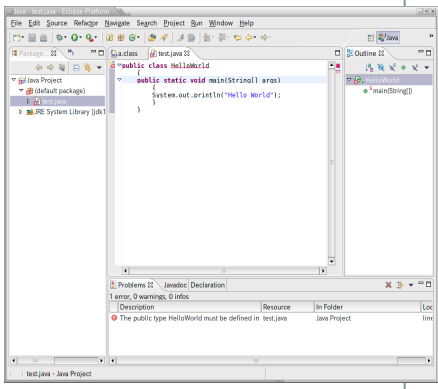
- Developed by Yukihiro Matsumoto

Web Development with PHP

Integrated Development Environments (IDE)

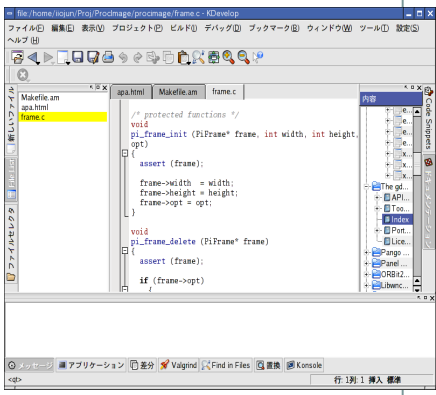
Eclipse

- Development environment implemented in Java
- Supports languages other than Java
  - Plug-ins for C/C++ development
  - C/C++ Development Toolkit (CDT)



IDE for various desktop environments

- Kdevelop for Qt/KDE
- Anjuta for GTK+/GNOME



Other IDE

- WideStudio: For creating GUI applications using C/C++

Development Tools

- Compiler
- Debugger
- Analyzer, profiling
- Source code management (CVS)
- Maintaining compatibility
- Localization
- Documentation
- IDE based on GUI
- Bug tracking tools

Compiler

- Process of building softwar
  - Compile
    - Source code -> object code
  - Link
    - Set of object code -> executable code
  - File describing the process of building software
    - Makefile
- gcc, make, ld
  - standard tools for OSS development

GCC

- gcc (the GNU Compiler Collection)
  - Development started in 1984 by Richard Stallman
- Originally stood for GNU C Compiler
  - Now stands for GNU Compiler Collection
  - Includes compilers and libraries for C, C++, Objective-C, Fortran, Java and Ada
  - C++ compiler g++
  - Fortran compiler g77
  - Java compiler gcj (gc-jdk)
- Features
  - Widely used for commercial and non-commercial operating systems
  - Can also be used as cross-compiler

Biên dịch với GCC

- Ví dụ đơn giản trong ngôn ngữ C
  - Soạn thảo hai file main.c và func.c

```
/******func.c*******/
#include <stdio.h>
void Hello(){
        printf("Hello World");
}
```

```
/******main.c*******/
int main(){
        Hello();
}
```

Biên dịch với GCC

- Biên dịch gcc [options] sources

  Các tùy chọn của options

  - -o: sinh ra tập tin output
  - -c: sinh ra tập tin đối tượng .o
  - -S: sinh ra tập tin assembly .s
  - -I: đặc tả thư mục chứa tập tin include
  - -l: đặc tả thư viện (.lib hoặc .a)
  - -L: đặc tả đường dẫn đến thư viện
- Chi tiếp xem thêm manual của GCC
  - gcc.gnu.org/onlinedocs/gcc-4.7.2/gcc/

Biên dịch với GCC

- Sử dụng gcc để dịch ra file đối tượng .o (obj)
  - Gõ gcc -c main.c (sinh ra file main.o)
  - Gõ gcc -c func.c (sinh ra file func.o)
- Sử dụng gcc đê sinh ra file thực thi từ files obj
  - Gõ gcc -o main main.o func.o (sinh ra file thực thi main)

Biên dịch make và Makefile

- Dịch với make và Makefile
  - Soạn thảo Makefile (nếu đặt tên khác thì khi sử dụng make phải dùng: make -f tenkhac)
  - Makefile là 1 file đặc biệt dùng để quản lý các tập tin trong dự án (project)
  - Chứa các quy tắc biên dịch
  - Sử dụng thuộc tính "modified time" để biên dịch lại
  - Tạo ra 1 đồ thị phụ thuộc
  - Cú pháp:
    - <target>: <danh sách các file phụ thuộc>
    - <TAB> lệnh

      Chú . phải dùng phím <TAB> chứ không phải khoảng trắng

Biên dịch make và Makefile

- Biên dịch với make và Makefile
  - Makefile có thể sử dụng biến, ví dụ:

```
OBJ = foo.o main.o
main: $(OBJ)
gcc -o main $(OBJ)
```

  - Một số biến đặt biệt:

$@: target

$+: danh sách các files phụ thuộc

$<: file đầu tiên trong danh sách

Biên dịch make và Makefile

- Biên dịch với make và Makefile để dịch tự động các bước trên
  - Soạn thảo Makefile như sau:

```
#Makefile
CC=gcc
main: main.o func.o
	$(CC)  -o main main.o func.o
main.o:  main.c
	$(CC)  -c main.c
func.o:
	$(CC)  -c func.c
```

  - Gõ make
  - Lệnh make sẽ đọc các bước dịch trong Makefile để dịch và sinh ra file thực thi main

Biên dịch: thư viện liên kết tĩnh

- Mặc định gcc có thể liên kết với các file đối tượng
  - Gõ: gcc -o main main.c func.o
- Thư viện tĩnh: chứa các file đối tượng .o được tạo bởi công cụ ar
  - Gõ: ar rcs libfunc.a func.o (tạo ra thư viện libfunc.a)
- Dịch main với: gcc -o main main.c libfunc.a

Biên dịch: thư viện liên kết động

- Mặc định gcc có thể liên kết với các file đối tượng
  - Gõ: gcc -o main main.c func.o
- Thư viện động: được load khi chạy chtrình, được tạo bởi gcc
- Gõ: gcc -c -fPIC func.c (tạo ra func.o)
- Gõ: gcc -shared -W1,soname,libfunc.so.1 -o libfunc.so.1.0 func.o

  (trên windows: gcc -shared -o libfunc.dll func.o)
- Tạo ra libfunc.so.1.0 (libfunc.dll)
- Dịch main với: gcc -o main main.c libfunc.so.1.0

  (trên windows: gcc -o main main.c libfunc.dll)

Biên dịch: thư viện liên kết động

- Chạy chtrình main: có lỗi không tìm thấy thư viện liên kết động
- Hoặc chỉ lại đường dẫn đến thư viện
- Gõ: export LD_LIBRARY_PATH=$(pwd)
- Hoặc sao chép libfunc.so.1.0 đến /usr/lib rồi
- Chạy ldconfig để cập nhật LD

Biên dịch với công cụ khác

- G++: biên dịch C++
- automake
- autoconf

Công cụ phân tích GNU GProf

- Utility for measuring and displaying operating status of program
  - Number of calls for each function, processing time, etc.
  - Shows bottlenecks to consider for acceleration
- Using gprof
  - Specify -pg option when compiling
  - Execute program normally
  - # gprof *executable-file* gmon.out
- Sample output
  - Func1 takes up zero time
  - Func2 has room for acceleration
-

```
 %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.00      0.40     0.40       80     5.00     5.00  func2
  0.00      0.40     0.00        3     0.00   133.33  func1
```

Công cụ phân tích memory: Valgrind

- Valgrind là công cụ phân tích/kiểm tra tìm lỗi trong việc cấp và giải phóng bộ nhớ

```
#include <stdlib.h>
void f(void)
{
  int* x = malloc(10 * sizeof(int));
  x[10] = 0;   // problem 1: heap block overrun
}              // problem 2: memory leak -- x not freed

int main(void)
{
  F();     return 0;
}
```

```
/*******compile ********/
gcc -o mypro mypro.c
/*******tesing********/
valgrind --leak-check=yes myprog arg1 arg2
```

Hệ thống theo dõi mã nguồn

- Lưu trữ trực tuyến mã nguồn các dự án
- Theo dõi những thay đổi trên mã nguồn
- Trộn các đụng độ trên cùng một tập tin
- Mô hình tập trung
  - CVS, Subversion, Perforce
- Mô hình phân tán
  - Git, Mercurial, Darcs

Xưởng phát triên phần mềm

- Gforge: Collaborative D.Environment
  - InriaForge
  - FusionForge (ex-GForge)
- SourceForge: sf.net
- Google code
- PicoForge
- NovaForge
- ...


AFUL
apereo
cenatic
democracyLab
Open Source Politics
debian
Drupal
eclipse
FreeBSD
Joomla!
THE LINUX FOUNDATION
LinuxFund.org
MariaDB FOUNDATION
mozilla FOUNDATION
NZOSS
OpenSource
Software Institute
Open Source Sweden
OUTERcurve FOUNDATION
OW2
Plone
prpig
python
SAHANA SOFTWARE FOUNDATION
tiki
wiki cms groupware
WIKIMEDIA FOUNDATION
wikiotics

Duy trì sự thương thích

- Differences in platforms
  - Differences in operating systems
    - Linux, BSD, Unix, Windows, etc.
  - Differences in libraries
    - OpenGL/Mesa, Xaw, Motif/lesstif, etc.
  - Differences in versions
    - Specifications can change due to version upgrades
  - Differences in paths
- Need arrangement for absorbing these differences
  - Labor-intensive to implement manually
  - Difficult to support platforms not possessed by developers