What the daemon / client supposed to do?

Client:

- Show the response from the daemon
- Give input from user to the daemon

Daemon:

- Process user input
- Evaluate user command
- Give response to the client

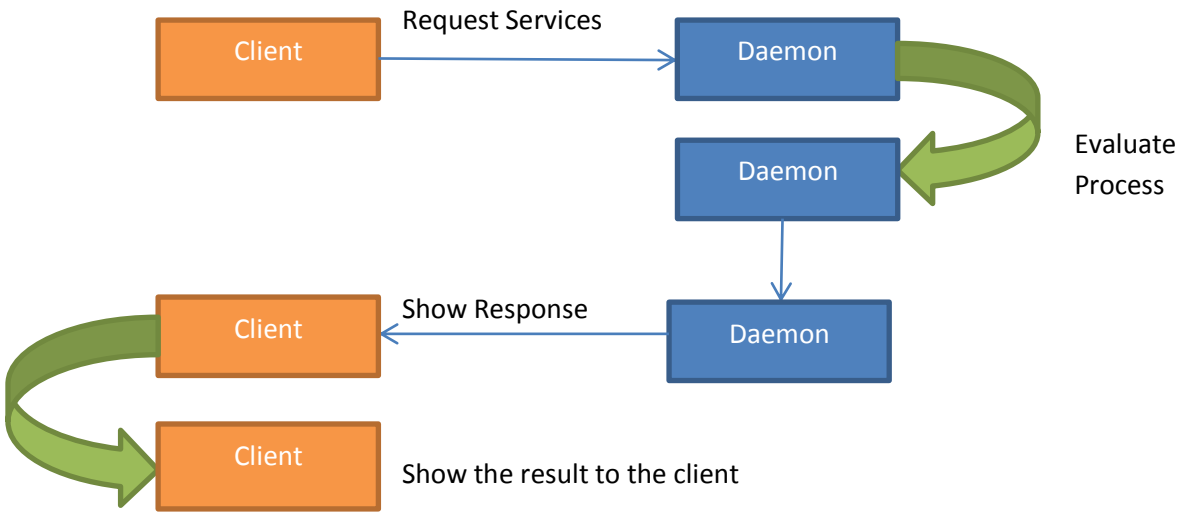
- What designs you could use (e.g. light client/heavy daemon, the other way around or a mixture)?

Light Client Heavy Daemon	Heavy Client Light Daemon
More secure because Daemon can filter what the client should see and what the client shouldn't see	Less secure because the daemon just give all of the information that the client request without filtering it
All of the process is done by the daemon	All of the process is done by the client so it makes the computer works slower
Commonly used	Rarely used

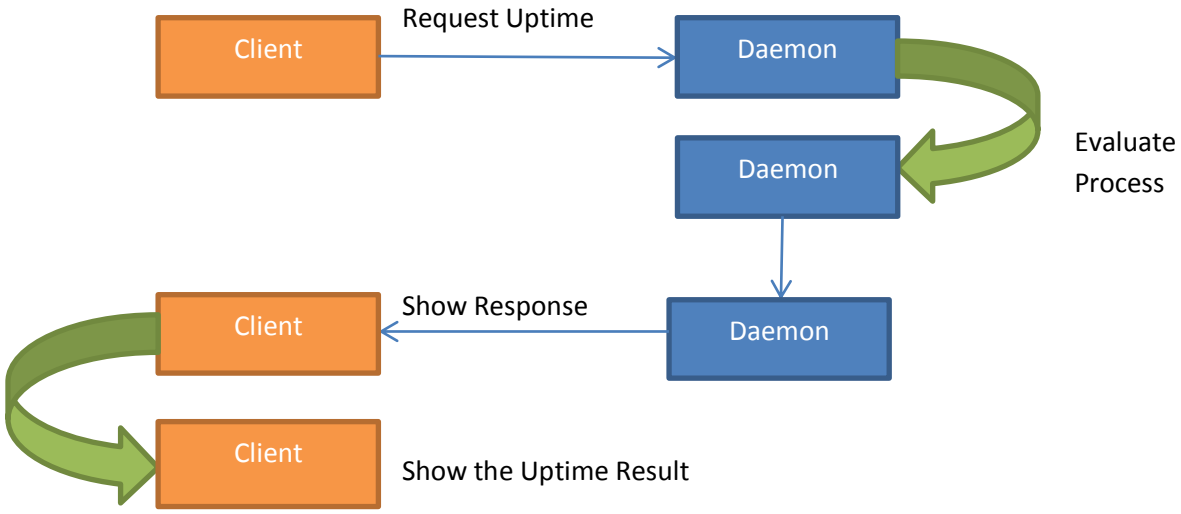
- How shoud the interaction between the two look like (e.g. a little chart)?

Request Services
Request Uptime Services
Request Space Disk
Request Process
Request User ID

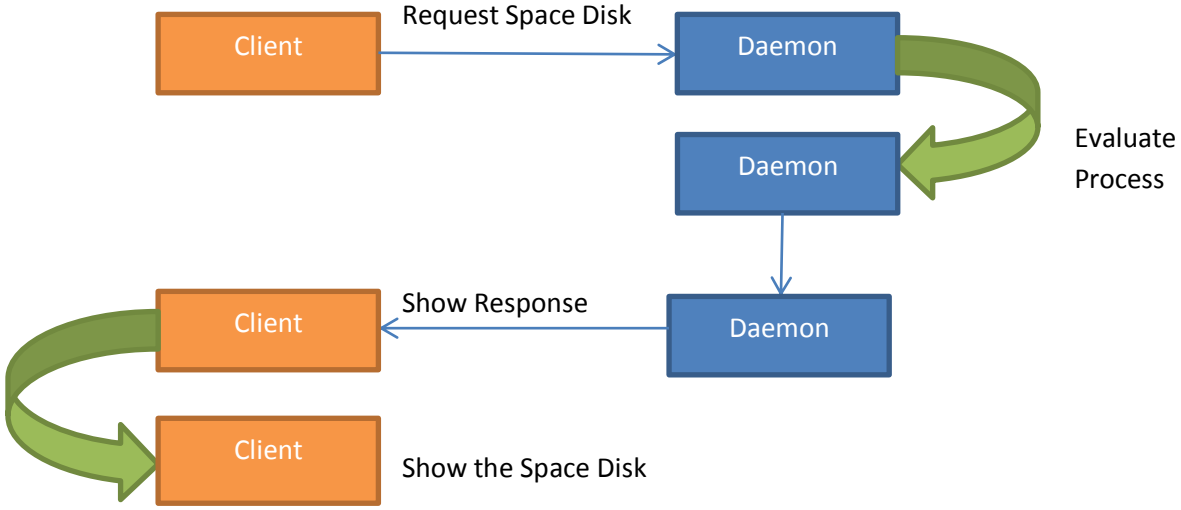
1. Request Services



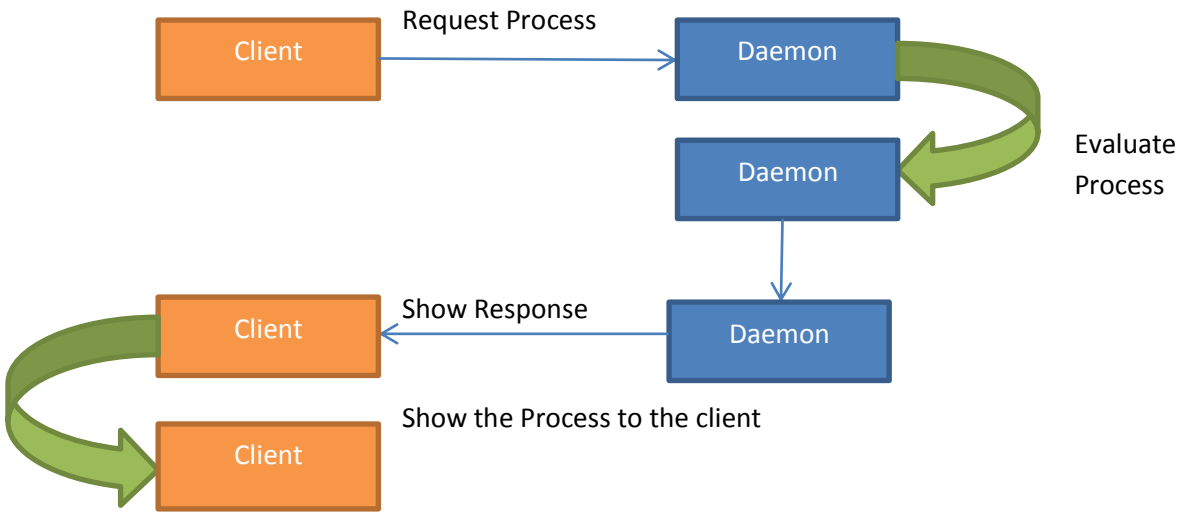
2. Request Uptime



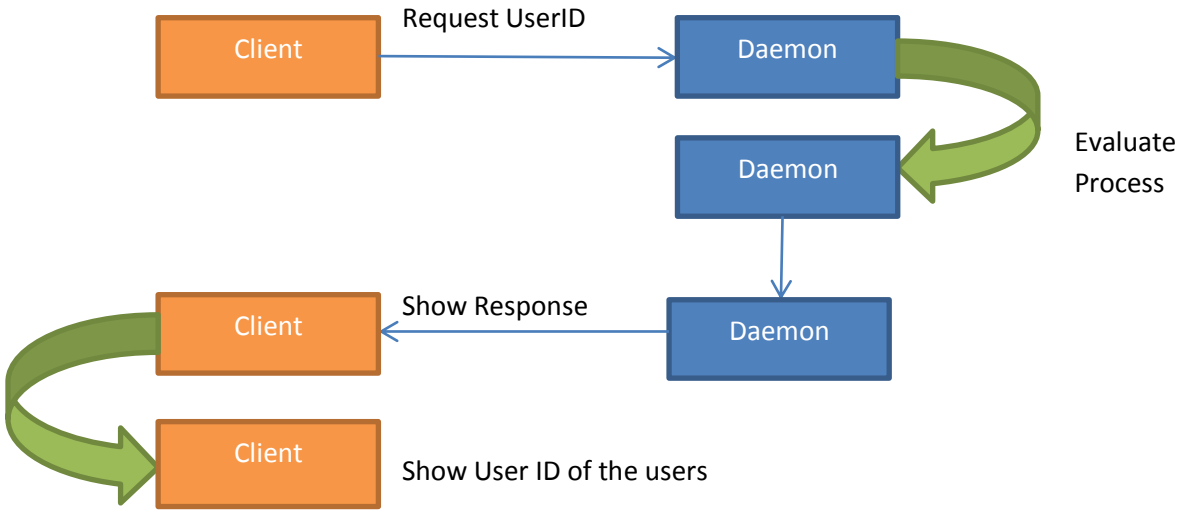
3. Request Space Disk



4. Request Services



5. Request User ID



- how to design the user interface that even the stupidest user could use it?

#Help
#Input
Please enter your choice:

If the user choice is help then this interface will come out

Command		Description
#Uptime	||	Show the uptime of the computer
#Space Disk	||	Show the disk usage and free space of the disk
#Process	||	Show the process manager
#User ID	||	Show the user who is log on the computer

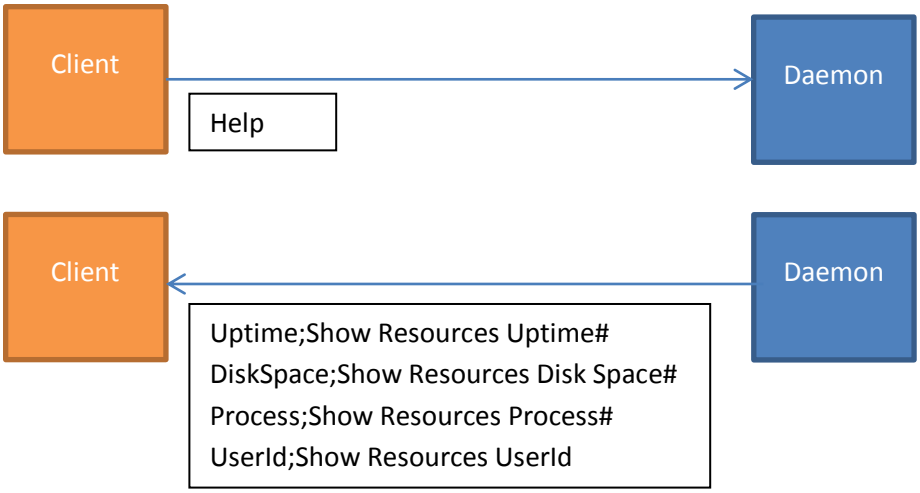
If the user choice is input, then he/she has to give the command based on the command that is available.

The result of uptime
03-09-2010 (date) 3 hours 24 minutes (Uptime)

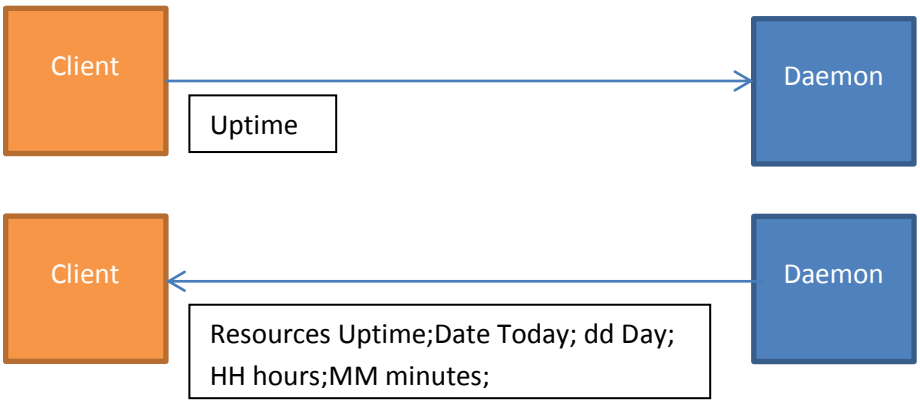
The result of Space Disk
C: 120 GB used, 80 GB free space
D: 24 GB used, 56 GB free space

Protocols

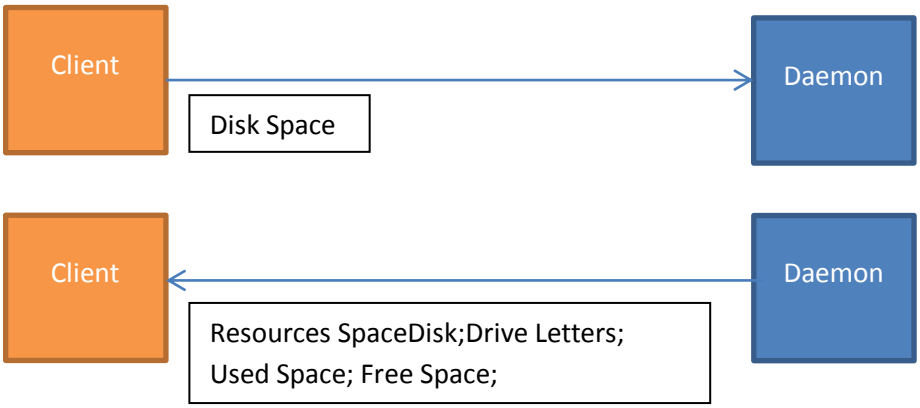
1. Services Protocols



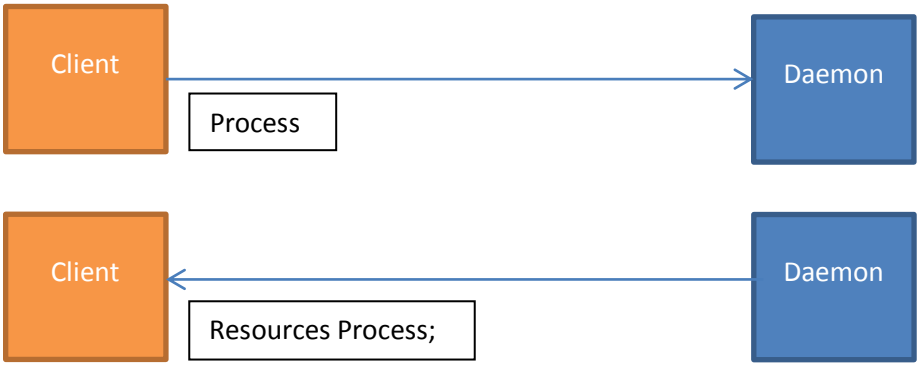
2. Uptime Protocols



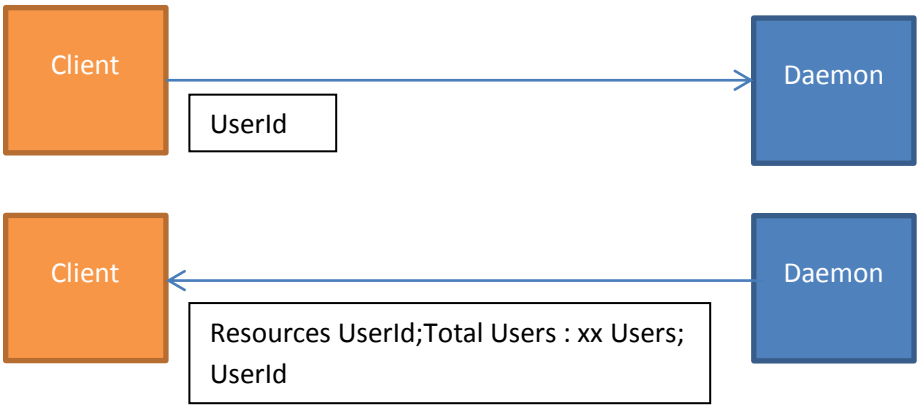
3. Disk Space Protocols



4. Request Process

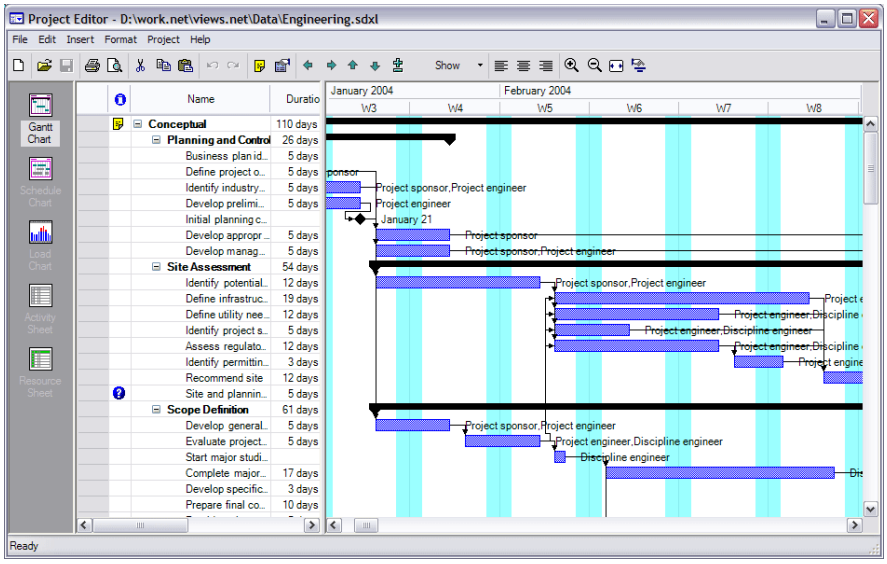


5. Request UserId

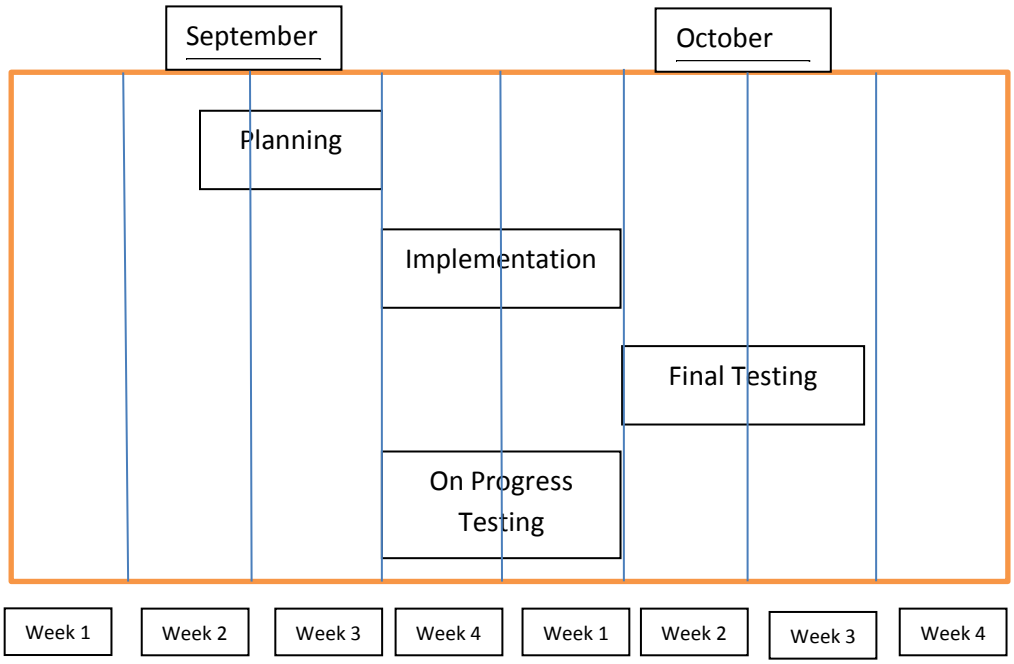


Appropriate Tools for Tracking Progress: Gantt Diagram

- Example of Gantt Diagram



My Gantt Diagram



Comparison between JAVA, Python and C/C++

1. Comparison between JAVA and C/C++

Java	C/C++
Created for supporting Network Computing	Mainly for System Programming
Don't have compatibilities with previous version of programming language	C++ has compatibility with C source code except for some cases
Write Once Run Anywhere / Everywhere (WORA/WORE)	Write Once Compile Anywhere (WOCA)
OOP Paradigm	OOP / Procedural Programming
Runs in Virtual Machines	Low-level system Facilities
No support for Unsigned Arithmetic	Supports Unsigned Arithmetic
Not always provide full access to the features and performance of the platform that the software runs on	Simple and robust providing containers and associative arrays
Portability	Speed of Execution
Ease of development	Speed of execution

2. Comparison between JAVA/C(C++) and Python

JAVA/C(C++)	Python
Run faster	Runs slower
Take more time to develop	Take less time to develop
Low Level implementation language	"Glue" Language

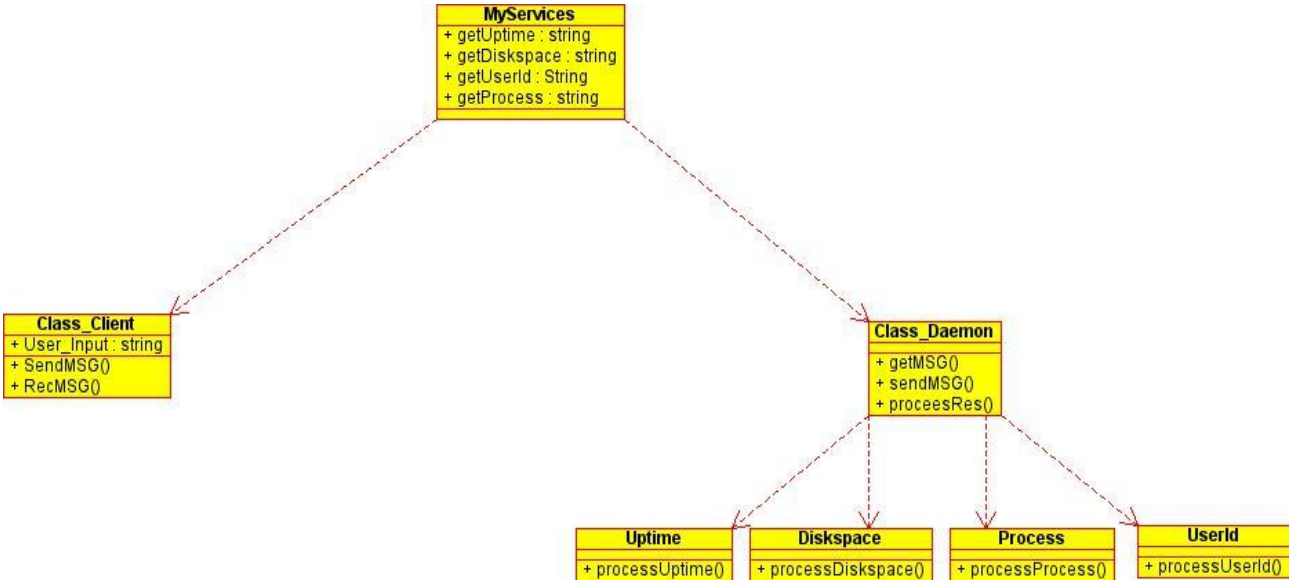
Static Typing	Dynamic Typing
Have to declare types of arguments or variables	Don't have to declare variables

3. Comparison between 3 programming language

	Java	C/C++	Python
Speed if Execution	Runs slower	Runs the fastest	Run the slowest
Typing type	Static	Static	Dynamic
Variables	Have to be declared	Have to be declared	Don't have to be declared
Developing Time	Need a long time	Need a long time	Need short time
Paradigm	OOP	OOP/Procedural	Multiple Paradigm
Language Level	Low Level Language	Low Level Language	High Level Language
Unsigned arithmetic	No support	Support	Support
Advantages	Portability	Speed of Execution	Ease of development

Creating Class Diagram

Client	Daemon
VarChoice = Variable	-
Public String getHelp = Method	-
Public String getUptime = Method	processUptime = Method
Public String getDiskSpace = Method	processDiskSpace = Method
Public String getProcess = Method	processProcess = Method
Public String getUserId = Method	processUserId = Method



Maintenance and security

For maintenance it is useful to just use 1 class service than 2 classes. If we make some modification then it will be applied into the entire class diagrams. So you won't have to change in the client or daemon services. Even it is a little bit lack in security but it is the common class that is used.

Pseudocode

Because we don't have the server, we simulate the reply with the usage of static string

class Client(object):

Initialization:

Init the socket

'Asking server for provided services... please wait....'

#Ask the server for services available on the server

services = UPTIME and DISKSPACE;

#Show the current uptime of the resource and Available diskspace

UPTIME and Diskspace in separate variables

We close the Services Variables

To show the interface to the user

Do as long as the user input is not exit:

Ask the user for the input for the services that he/she wants and sends it to the server

If we have the server, it will give responds based on the user input

But if we don't have it, we can simulate it with static string

Pseudocode for Client

```
#Iniitialization

Server.init #connection to server

Services = server.askservices() #getAvailableServices


#Display choice to user

Print services

Read choice

Server.send(choice)

Reply= server.receive()

Print reply

Goto askchoice()


Server pseudocode

Con = create sockert

Con.wait()

Request = con.receive()

Switch.req:

        UPTIME =getUPTIME

        DISKSPACE = getDISKSPACE

        PROCESS = getPROCESS

        USERID = getUSERID

server.send(choice)
reply = client.receive

print reply
```

http://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B

http://www.comp.lancs.ac.uk/~ss/java2c/diffs.html

http://www.python.org/doc/essays/comparisons.html

http://javanetbeans.net78.net/kb/60/uml/class-diagram.html

http://wiki.answers.com/Q/What_is_the_difference_between_Java_and_the_C_programming_language

http://en.wikipedia.org/wiki/Python_%28programming_language%29

http://www.dmh2000.com/cjpr/

http://en.wikipedia.org/wiki/Gantt_chart

http://www.google.com/imgres?imgurl=http://www.gantt-chart.biz/gcImages/Gantt_Chart.gif&imgrefurl=http://www.gantt-chart.biz/gantt-charting-made-easy/&h=596&w=945&sz=55&tbnid=DXzIRQ6WZqasCM:&tbnh=93&tbnw=148&prev=/images%3Fq%3DGantt%2Bdiagram&zoom=1&q=Gantt+diagram&hl=en&usg=__HWoQFB26iR_GybD2Dd3PtWBTSsQ=&sa=X&ei=ryicTIjyEoGsvgO_mtC6DQ&ved=0CBsQ9QEwAA

http://en.wikipedia.org/wiki/Unified_Modeling_Language