Macro-level software evolution: a case study of large software compilation

This paper studied the software evolution of compilation software that composed of large quantity of independent development product,engineered to work together. The packages that they studied was libre software that is collection of software packages engineered to work in coordination providing the use with large operating system, may be thousands of applications, they studied the stable releases of this software.

They find that total size of the application growth in super-linear in stable versions, and they find that the size of large packages getting large while produce new version of the software also there is many new small packages are added over time. while the maintenance packages there is some packages removed in new version some other did not get any modification ,other the get change. Also the programming languages that used was mostly C programming languages, then C++, also there is some use of Shell increased over time, because adding new packages need to more shell command to install and integrate.

Bad Points:

1- In introduction when they talked about evaluation of software system that comprised large set of applications and libraries they said that "study the source of whole system at certain point of time is not easy" they did not said why isn't easy they cannot say just isn't easy they have to give evidence for that.

2-In description of Total size they said that "if we considered only released, the growth would be super linear, the main reason is the time interval between subsequent" they have to say why time interval between release effect on software growing, they cannot say just if effect they have to say why, and which factors leads for that like developer have enough time or what.

3- In maintenance of packages, they said there is some packages did not changes, over releases , but they didn't said why they still stable in all version and why.

4- Growing size of packages is they never mentions the effect of using new programming languages of project size , because different programming languages has different size “ hello word in C++” different in size with “hello word” in Python because the dependent files that python need is more than C languages.

Good points

1- They test system have many applications that dependent on each other rather than another researches that used single application, that make then have a lot of samples to prove their work instead of one sample.

2- The description of Libre software and it packages was very clears to anyone to understand how it is work and why they select this software , because it is open source and have many packages.

3- The prove that the numbers of dependencies between packages are growing quickly over the versions, because more small packages are added to support the large size packages work without change it, because they didn't want to do many changes in some large packages or it is hard to do that, so they tried to use new small packages to do that so the dependency is increased.