Open source framework

Introduction

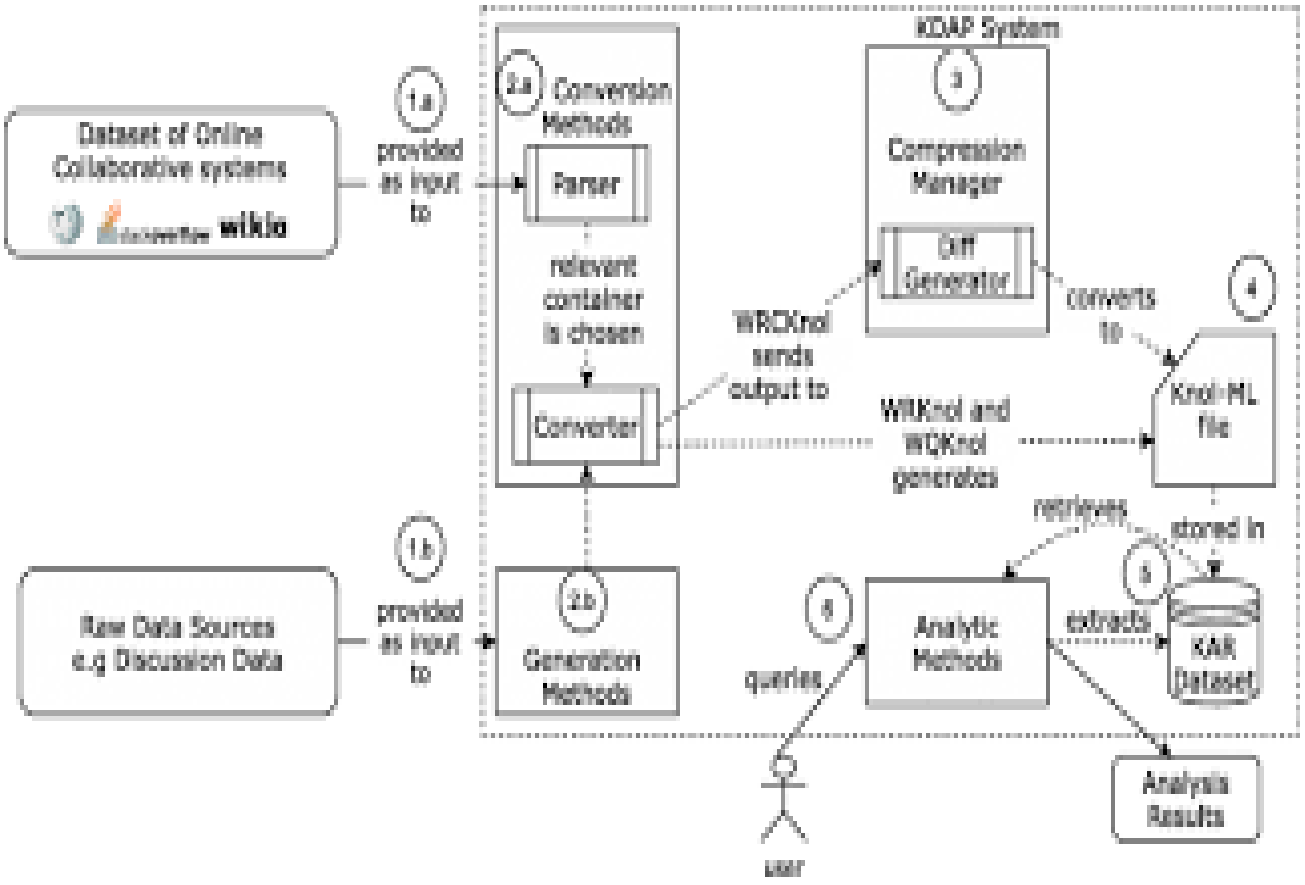
It is frequent to make a distinction between the terms ‘free software’ and ‘open source software’. Free software refers not to price but to liberty to modify and redistribute source code. The Free Software Foundation [4], founded by Richard Stallman, advocates the use of its GNU General Public License (GPL) as a copyright license which creates and promotes freedom. He writes “to understand the concept, you should think of free speech, not free beer” [5]. The term ‘open source’ was coined by a group of people concerned that the term ‘free software’ was anathema to businesses, this was resulted in the creation of the Open Source Initiative (OSI) [6]. We use the acronym FOSS for both movements for the sake of simplicity and because both movements share most of their practical goals and Free Software Foundation (FSF)

In January 1984 one of the original MIT AI Labs hackers, Richard M. Stallman, quit his job at MIT and founded Free Software Foundation (FSF). He objected to the increasing commercialization of university software research [4]. Stallman feared that despite the fact that popular Unix standards like Sun's were broadly distributed, they still remained under private ownership and would be used for proprietary advantage, which is what happened by the early 1990s.

Open Source Initiative (OSI)

The Open Source Initiative (OSI) is a California public benefit corporation, it is an organization founded in 1998 by Eric S. Raymond to promote open source software and development strategies. The OSI are the stewards of the Open Source Definition (OSD) and the community-recognized body for reviewing and approving licenses as OSD-conformant [4].

Framework model diagram:



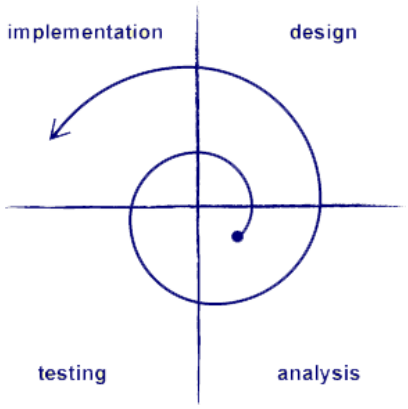
Software Development Life Cycle (SDLC)

A software life cycle model depicts the significant phases or activities of a software project from conception until the product is retired. It specifies the relationships between project phases, including transition criteria, feedback mechanisms, milestones, baselines, reviews, and deliverables. Typically, a life cycle model addresses the phases of a software project: requirements phase, design phase, implementation, integration, testing, operations and maintenance

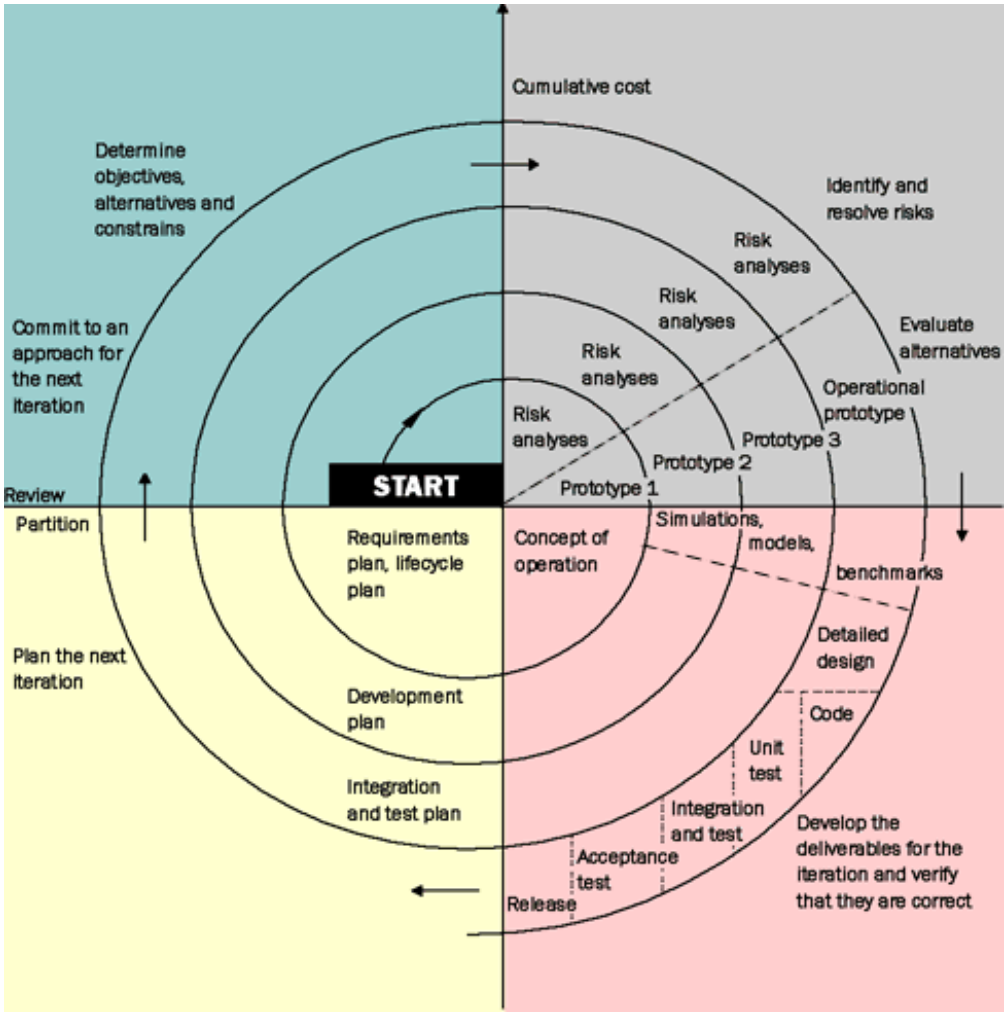
Software life cycle models describe the interrelationships between software development phases. The common life cycle models are:

A better model, the "spiral model" was suggested by Boehm in 1985; the spiral model provides useful insights into the life cycle of the system. It could be considered as a generalization of the prototyping model [30]. That why it is usually implemented as a variant of prototyping model with the first iteration being a prototype. But it also supposes unlimited resources for the project, No organization can perform more then a couple iterations during the initial development of the system, the first iteration is

usually called prototype as shown in Figure 4.1 Spiral model phases.



Spiral model phases



Advantages of Spiral Model

1. Avoidance of Risk is enhanced.
2. Supports for dynamically changing requirements.
3. Strong approval and documentation control.
4. Implementation has priority over functionality.
5. Additional Functionality can be added at a later date.

Disadvantages of Spiral Model

1. Highly customized limiting re-usability.
2. Applied differently for each application.
3. Risk of not meeting budget or schedule.
4. Requires expertise in risk evaluation and reduction Complex, relatively difficult to follow strictly. [29]

Recent Software Process Challenges

There are many challenges that facing software development process, however we find that OSS approach cover some of these areas in its development practices such as how to maintain global software development, easily process improvement because OSSD processes often iterate daily versus infrequent singular software life cycle engineering events which could make it easier to improve, but still yet there are no solid solutions to face these challenges such as process improvement,here we will summarize some of these challenges that facing both: Distributed, decentralized, and/or global software development.