Survey of software architecture V0.0

See book: http://www.aosabook.org/en/index.html

Name of system: CMake
Reviewer: Alan Kinzie
Date: Nov 10th, 2011
Author of software: Kitware (lead architect: Bill Hoffman)
Author of book chapter: Bill Hoffman, Ken Martin
Five star rating of book chapter: ****
Purpose of system:

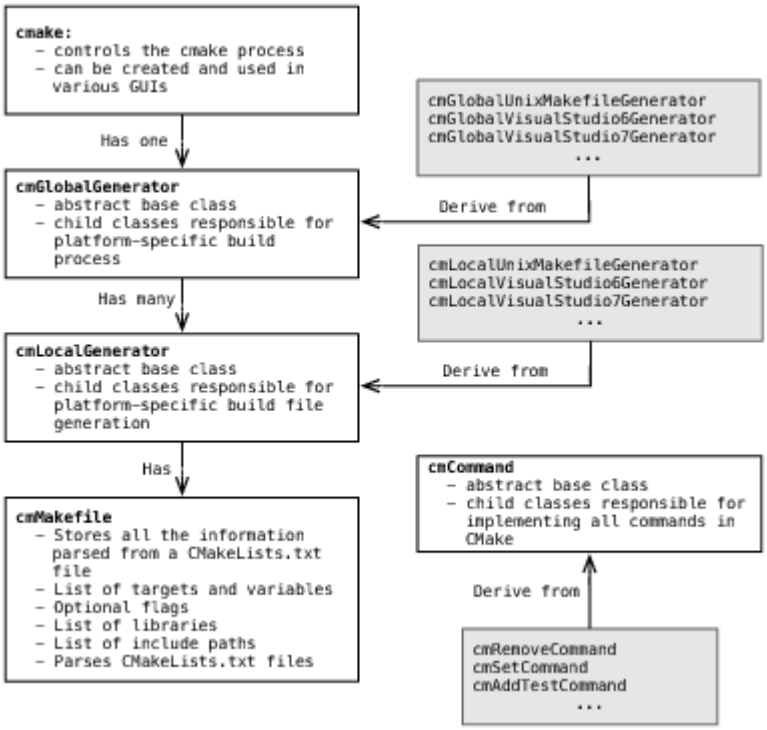
The purpose of CMake is to make it easy to build complex software for different platforms. It is meant to increase productivity by allowing developers to focus on more important things.

Basic metrics

KLOC: 1000
Project start-up: 1999
Number of major releases: 2
Number of developers: ~100 contributors
Size of user community or number of installations: 110 listed projects (http://www.cmake.org/Wiki/CMake_Projects) use CMake, plus probably many unlisted projects.
Major stakeholders: Developers of large and complex software projects, especially if those projects are cross platform.
Use of concurrency: There does not seem to be any use of concurrency
Implementation language: C++
Supporting software: Only needs a C++ compiler installed. Dependencies were purposefully kept to a minimum.

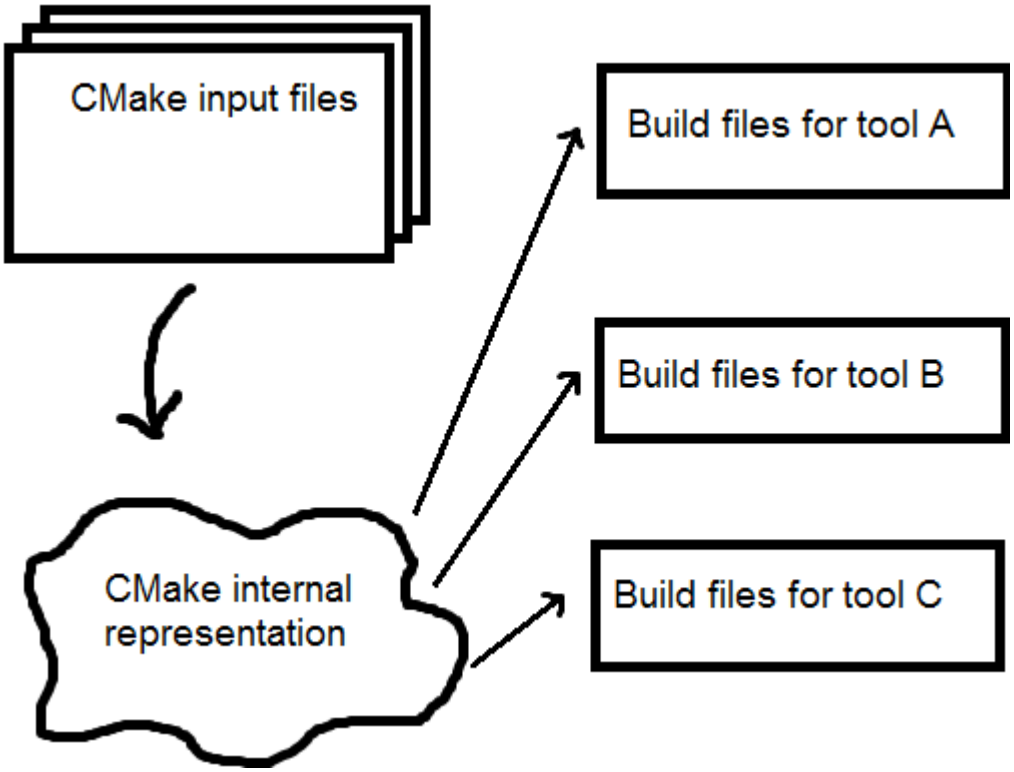
High level architecture

Diagram of software architecture



High level scenarios

Project needs to be build in three different environments (A, B, and C).



Data structures or algorithms

CMake has support for automatic dependency analysis. The dependency data is stored in four files (storing dependency information of object files, compilation flags, source file statuses, and building dependencies).

Control flow and/or data key to the architecture if any

Architectural style: The major architectural style is object oriented. Each command in the Cmake language has a corresponding C++ object which has the implementation and documentation for that command. Also, each specific environment has a global generator object that is derived from a common abstract class (and similarly for local generator objects).
Major evolutionary changes: I do not know of any major architecture changes.
Performance bottlenecks: Performance doesn't seem to be a big deal for Cmake. Building a large project would usually be automated and behind the scenes (or at least out of sight), so one usually wouldn't care exactly how long the build process takes.
Real time: Nothing about Cmake needs to be real time (other than the GUI, but that's not interesting).
Notation for architecture: An object diagram was used to describe the architecture in the chapter.
Methodology: Testing is done in an automatic continuous integration manner using entirely CMake, CTest, and CDash. Other than that, the methodology is not very clear. The development has clear objectives, however, having a set of requirements to full-fill from the very beginning (which is what guided the development for the first while).

Appendix:

Kruchten's eight context attributes applied to Brown/Wilson systems

Kruchten attributes described in his slides 17-24 of
https://files.me.com/philippe.kruchten/sbz0ma
Also in: https://files.me.com/philippe.kruchten/1q00nw

1. **Size:** L
2. **Criticality:** Med
3. **Age of system:** L
4. **Rate of change:** Lo
5. **Business model:** open source
6. **Stable architecture:** Lo
7. **Team distribution:** med
8. **Governance:** med