Everything on configuration management

A large program may have different released versions/configurations for different hardware, operating systems or client needs.

There may be parallel versions of components (e.g. same function but for different OS)

There may be sequentially related (e.g. earlier and later versions)

Some clients may choose to remain with older, known and stable versions of a piece of software for financial or stability reasons.

Configuration Database

Used to:

Assess the impact of changes

Generate reports

Control system builds for releases

Monitor progress on requested changes

CASE (Clearly Aided Software Engineering) tools are valuable here.

Build Control Tools

Make	Utility for automatically building and deploying executable programs Controlled by files and called makefiles. These contain rules specifying how to derive the target program from source files (declarative) Dates from the 1970s and is still widely used
Apache Ant (Another Neat Tool)	Can be used stand-alone or as IDE plugin Oriented to Java Uses XML build file to describe build process It is procedural (not declarative) build.xml contains targets (a.k.a rules); targets are named goals (not files); each mentions other targets that it depends on so they are executed first. Is low level
Apache Maven	Originally for Java but plugins available for other languages Can be used stand-alone or as IDE plugin Maven is higher level than Ant

Projects with standard structure:

Detailed configuration not required

A default project object model assumed

Maven knows how to compile, test and build

Developer shielded from details

Projects with non-standard structure:

The POM must be customized and described

Maven projects must be understood in great detail

pom.xml describes the project configuration; this is auto-generated for standard projects and customized for non-standard