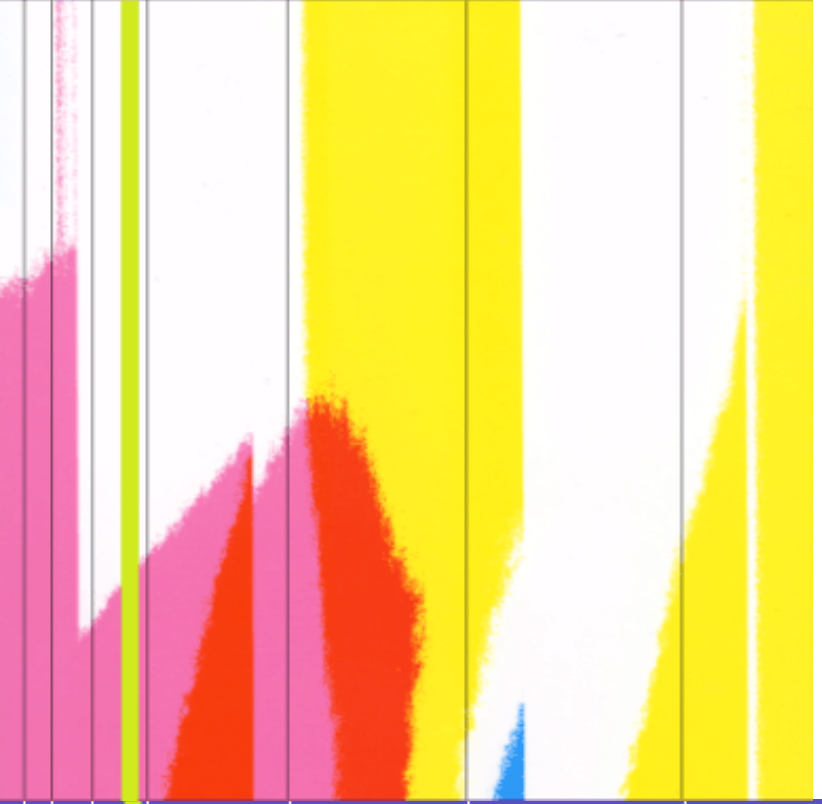
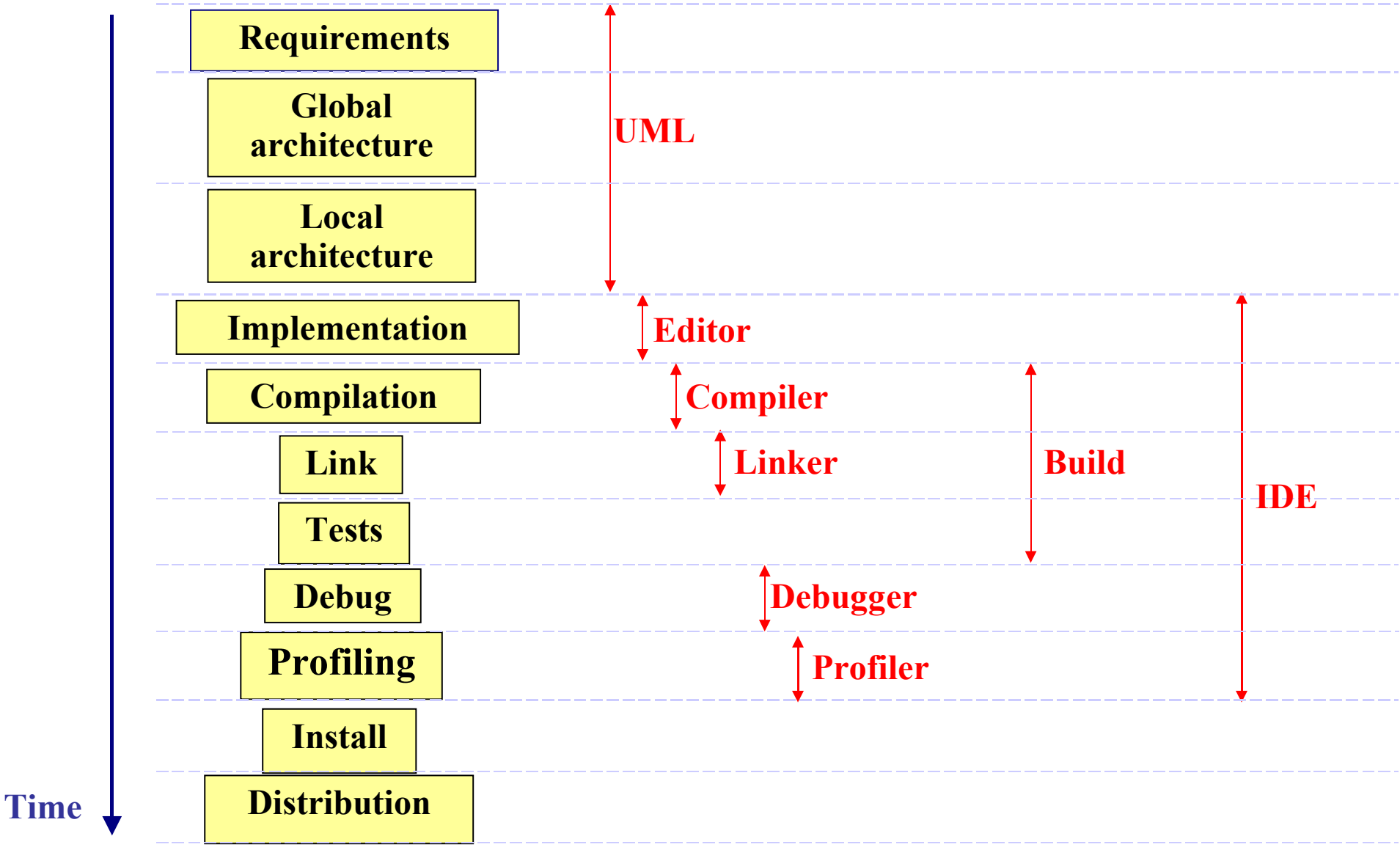


Closing Remarks

DREAM

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA


Requirements
Global architecture
Local architecture
Implementation
Compilation
Link
Tests
Debug
Profiling
Install
Distribution
Time
UML
Editor
Compiler
Linker
Debugger
Profiler
Build
IDE

Closing Remarks

Software Development best-practices

- Known as efficient
- Shared and agreed by all members of a development team

Two classes of best-practices

- For pure development activities (day 1)
- For transverse activities (day 2)

Development best-practices - 1

Specifications and Design

- Not wasted time
- Build on well-known and widely accepted elements : design patterns, language idioms
- All documents are under version control

Build

- Learn how libraries work on the target platform
- Use GNU make for small projects
- Automate all operations

Debug

- Ask other people when no progress is made
- Use the right tool(s): static analysis, debuggers, memory checkers

Development best-practices - 2

Tests

- Do the easy work first:
  - Simple input combinations
  - Boundary conditions
- Begin with white-box testing
- Automate test execution and reporting: start with a simple and easy mechanism

Performance and profiling

- No optimization without first profiling
- No premature optimization
- No over-optimization

Best-practices in transverse activities

Progressive improvement

- Practice
- Accept that learning takes time
- 2 levels of best-practices to start with

First level

Simple planning of development work

- Define and schedule the work over short periods (eg. a week)
- Use specifications to define work
- Re-schedule frequently and regularly

Use versioning systems

- Useful also for a single person (we are **always** part of a team)
- Create a project on the Inria Forge

Second level

Advanced Planning

- Schedule over long periods of time
- Use task/subtask decomposition
- Track progress

Documentation

- Design and implementation
  - Simple and short documents: revise them frequently
  - Build over standard knowledge: design pattern, language idioms, etc.
- API documentation
  - Use documentation extraction tools
- Develop examples and ready to use code from the use-cases (specifications)

Resources

Document : “Un processus de développement logiciel pour l'INRIA”

http://www-sop.inria.fr/dream/rapports/devprocess/index.html

Resources

Document : “Un processus de développement logiciel pour l'INRIA”

http://www-sop.inria.fr/dream/rapports/devprocess/index.html

DREAM: dream.permanents@sophia.inria.fr