



# Enhanceable tooling for every day programmers

Oliver Schneider, Hubert B. Keller

Institut für Angewandte Informatik IAI

# Tooling

- Ubiquitous tooling got traction in scripting languages
  - Javascript 
  - Ruby 
- Recently also in compiled languages
- Easy-to-use tools are used by more developers
- Higher usage leads to more attention paid to improving the tools
- Thus: Easy-to-use tools are better

## Usability influences Usage

- Number of installation steps required (compiler, IDE, ...)
- Useful error/warning diagnostics
  - Visually close to actual mistake
  - Suggest fix(es)
  - Comprehensible for beginners
- Effort required to update compiler, IDE, tools
- Forward Compatibility (Code still works after compiler update)
- Effort required to cross compile
- Effort required to integrate other libraries in project
- Effort required to add custom features to tools

# Number of installation steps

## ■ C/C++

- Linux: through package manager: one command
- Windows: Together with IDE (Find download, download, install)

## ■ Rust

- [rust-lang.org](http://rust-lang.org), downloads, download, install

## ■ Go

- Linux: install through package manager
- [golang.org](http://golang.org), download, install

## ■ Ada (GNAT)

- Linux: through package manager possible, but not recommended
- [libre.adacore.com](http://libre.adacore.com) → downloads → free software → continue → select platform → select tool → download → install

# Effort required to update compiler/IDE

## ■ C/C++

- Linux: automatically updated through package manager
- Windows: Repeat installation steps

## ■ Ada

- Repeat installation steps for each installed target

## ■ Go

- Linux: Automatically updated through package manager
- Or repeat installation steps

## ■ Rust

- „rustup update“

# Effort required to integrate libraries

## ■ C/C++, Ada

- Method depends on compiler, no language standard
- requires manual download and integration of libraries
- Repeat process for updating libraries

## ■ Rust

- Package manager integrated into build tool
- Semantic versioning of libraries
- `regex = „1.1.0“`

## ■ Go

- Several package managers available
- Copies all libraries into project for reproducibility

# Tooling

- C/C++, Ada

- No standardised method of tool distribution, usage, ...

- Rust

- „cargo install tool-name“

- Go

- „go get tool-name“

## Examples for tools

- Static analyses
- Code formatting
- Symbolic execution
- Code Coverage
- Language Server
  - generic interface between IDE and Compiler by Microsoft
- Code statistics
- Debugging aids
- Profiling



## Every tool is imperfect

- Extend tools with project specific features
- Fix bugs that occur when running tool on private project
- Tool might be opinionated
  - Make it configurable so it has your opinion if you tell it to
- Making changes yourself reduces waiting time to days instead of months or years

## Conclusion

- Simplify downloading, installing, updating, modifying and using of
  - The compiler
  - Tools
  - IDEs
  - Libraries
    - Not a compiler issue, but a language issue
- Simplify contributions or improve extensibility
  - Closed source tools can offer plugin APIs
  - Open source tools need to be easy to build, quick to comprehend, ...



## Open Access

- Many Ada Tools are purely commercial
- Commercial tools are not used in open source projects
- Low visibility of unused tools
- Unknown tools are not used in any project

## Error detection

- Early error detection is better/cheaper than late

Coding → Compiling → Testing → Running → Deploying → Using

no significant move from exceptions to compile-time diagnostics (Ada)