Programming Languages and Development Tools: State of the Art and (Hopefully) the Future

Dr. Bambang Purnomosidi D. P.
@bpdp

STMIK AKAKOM
17 Juli 2017

Agenda

1. Programming Languages and Development Tools
2. Programming Paradigms
3. Programming Environment
4. Domain Problems
5. The Future (?)

Programming Languages and Development Tools

- Programming Languages (PL): **formal language** that specifies a set of instructions for computer to do something.
- PL should be seen from 2 points: specification, reference implementation, and vendor implementation.
- Development Tools (DT): a set of tools to help software developer in doing their tasks to create software product:
  - IDE (Integrated Development Environment)
  - Compiler / Interpreter
  - Profler
  - Debugger
  - Editor
  - Pavkage Management
  - Programming Environemnt (discussed later)

Programming Paradigms

- Programming Paradigm: related with the way programmer solve programming problem.
- PP are implemented as programming language features.
- Some PP:
  - Imperative: allow side effects, sequential, there are constructs to control order. Ex: Perl, Python, JavaScript, PHP
  - Functional Programming: disallow side effects, treats computation as the evaluation of mathematical functions, avoids changing state and mutable data. Ex: Haskell, OCaml
  - Declarative: order is not important, declare something and have computer do the work. Ex: SQL
  - Object Oriented Programming: treas class as an encapsulation of executrable code and let the objects from the class to collaborate. Ex: Java, C++
  - And many others

Programming Environment

- PL is not the only thing to consider when we want to build software product
- PL environemnt are other things related to programming and DT, important for successful delivery of software products although may not be technically related.
- It is the answer of: “I can program, I can do programming tasks, now how should I begin and how do I manage it so that I can deliver software product?”
- PL env:
  - Software development methodology: waterfall, prototyping, agile.
  - Cloud-enabled DT (platform as a service): Docker, Kubernetes, rkt, unikernel
  - Distributed Computing: Data serialization: XML (OBSOLETE), JSON, msgpack, protocol buffer, etc.
  - Testing and Continous Integration
  - Online tools: workflow management (ex: trello), communication (Slack)
  - Project Management: Github, Gitlab
  - Product Development: Aha (aha.io)

Domain Problems

- Frontend:
  - GUI and TUI
  - Web
  - Mobile
- Backend - Distributed System: low latency PL
  - PL - Comiler and libraries: microservices. Go is still the best.
  - Database: SQL, NOSQL, NewSQL
- Big Data
- Artificial Intelligence: Machiine Learning and Data Mining (ex: TensorFlow), Deep Learning: see Julia and R.
- Embedded System: Raspberry Pi etc: need specific OS and specific development tools: Python, Node.js, Rust, Go, C

The Future (?)

- What PL most used in industry? From TIOBE, still C, C++, Java. Pay attention to serious contender: Go and Rust.
- JavaScript is and still be the lingua franca: People love it and hate it at the same time. See JS engine (V8 from Google, etc). To track its development, look at ECMAScript website: ES5, ES6, ES7, ES Next
- There is a trend that every compiler now has package management and the package management is iused to install the compiler and its supporting tools: rustup in Rust, opam in OCaml, Stack in Haskell, etc.
- PL are getting into multiparadigms
- Distributed Systems need special attention, there will be no software intended for single computer only.
- Big data, NewSQL, Artificial Intelligence will be big
- One framework - multiplatform deployment

Discussion Time