MATLAB in a Parallel Environment

Jos Martin
Principal Architect for Parallel Computing Tools

jos.martin@mathworks.co.uk

Overview

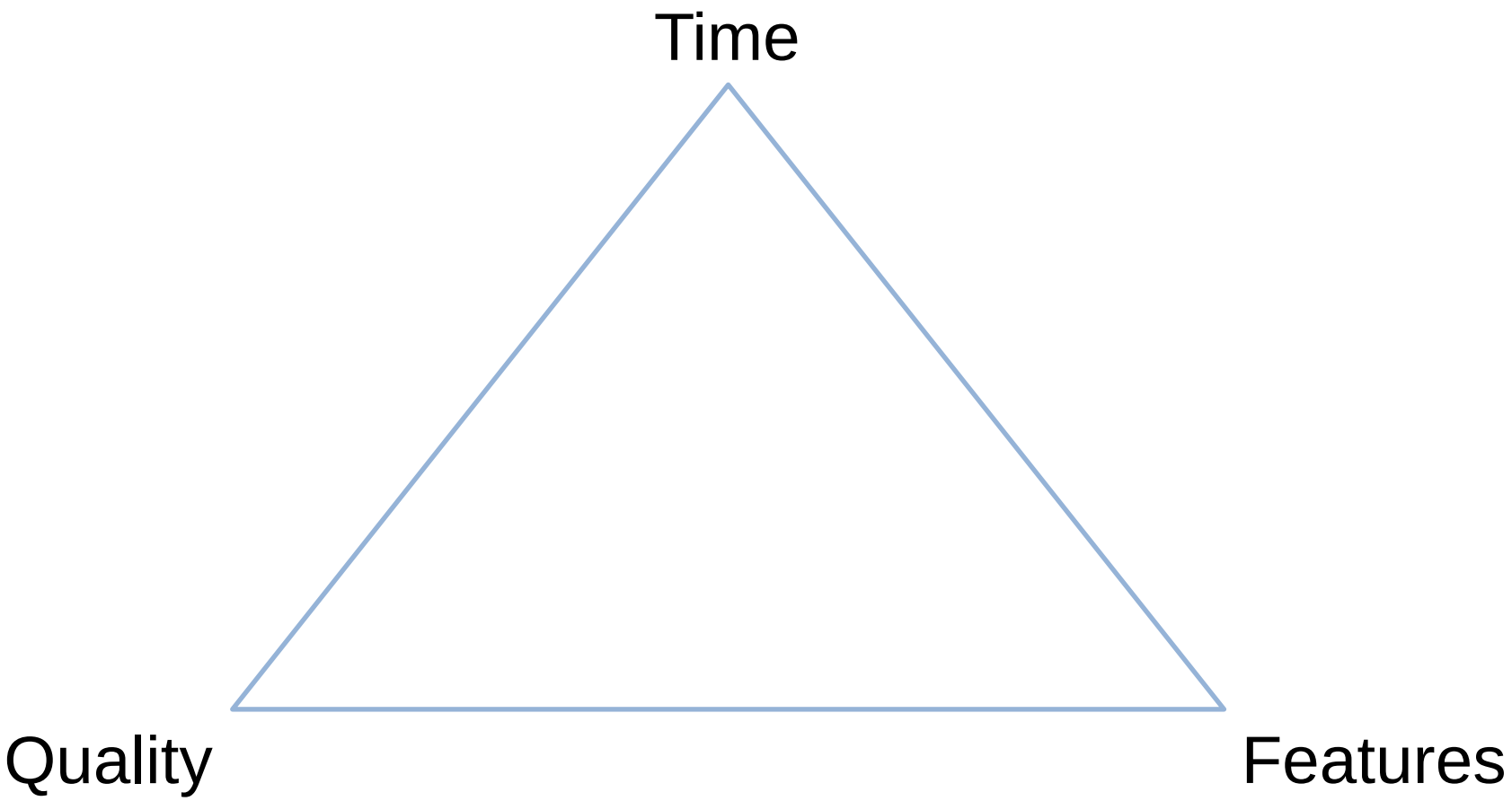
- Setting up the problem – what are we trying to build?
- What have we got to work with?
- How are we going to do
  - Design
  - Implementation
  - Testing

Attributes of good software

- Bug-free, Fast, Accurate, Usable

- Accurate
- Usable
- Bug-free
- Fast

Development trade-off



The Martin Software Engineering Principle

Frequently Jos is surprisingly stupid

A software engineering methodology

- What and Why?
- How? (for the user)
- How? (for the software system)
- How? (no seriously, I do actually need to write some code!)
- Did I make a mistake? (in my own code)
- Did I make a mistake? (in my team)
- Did I make a mistake? (in any code from MathWorks)
- …

A software engineering methodology

- Requirements
- Functional Design
  - Function prototypes, object interface, API, etc.
  - Graphical User Interfaces
- Architecture
  - Modularity
  - Components
  - Reuse

Real situation: Make MATLAB use GPU's

- Requirements
  - Things the GPU is good at – maths, what else?
  - All GPU's? Some subset?
- Functional Design
  - The Nike Approach – Just Do It …
  - Explicit opt-in

Real situation: Parallel NLA

- Requirements
  - PGAS
- Functional Design
  - Only collective operations?
  - Need parallel jobs? MPI?
  - Interactive?

“Simple to use” vs. “Lots of control”

- Solve **`A*x = b`** for unknown **`x`**
- Simple

```
x = A\b
```

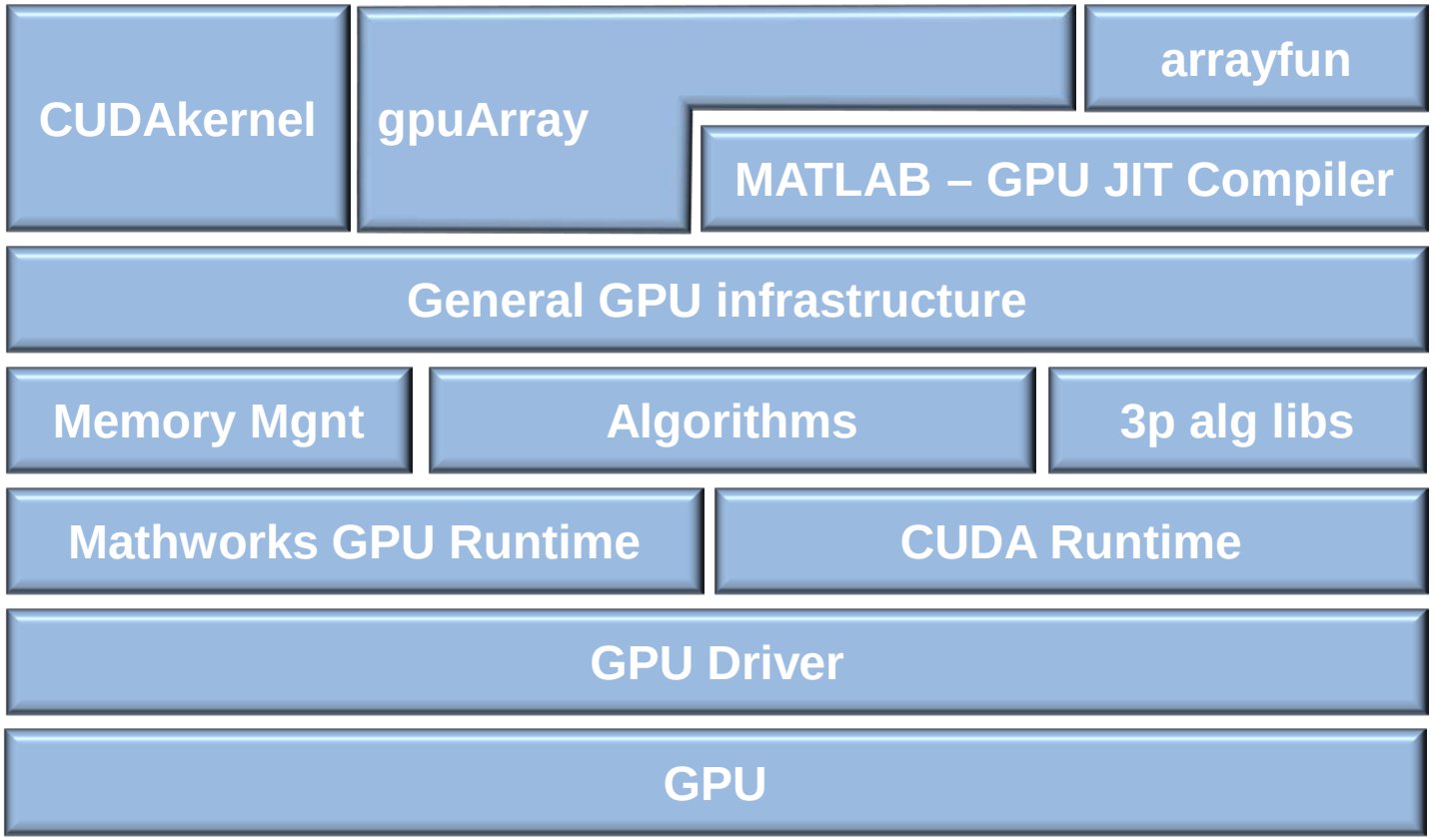
- Lots of control

```
DGETRS
DGBTRS
DGTTRS
DPOTRS
DPPTRS
DPBTRS
DPTTRS
DSYTRS
DSPTRS
```

“Simple to use” vs. “Lots of control”

Level of Control	Feature	
Simple	`gpuArray`, maths, same syntax as MATLAB	`distributed`
Intermediate	`arrayfun(@fun,...)`	`codistributed`, **`spmd`**, `composite`
Detailed	Direct integration with CUDA kernels	`codistributor`, MPI-like programming

Architecture (GPU)



Architecture (parallel NLA)