The Timing Definition Language (TDL)

Department of Computer Science
University of Salzburg

Josef Templ
updates by Claudiu Farcas

UNIVERSITÄT SALZBURG

Overview

- What is TDL?
- TDL Component Model
- Simple TDL Example
- Tool Chain
- Current State

What is TDL?

- A high-level textual notation for defining the timing behavior of a real-time application.
- Conceptually based on Giotto (University of California, Berkeley).
- **TDL = Giotto + syntax + component architecture + cleanups.**

Analogy: IDL (CORBA, MIDL) vs. TDL

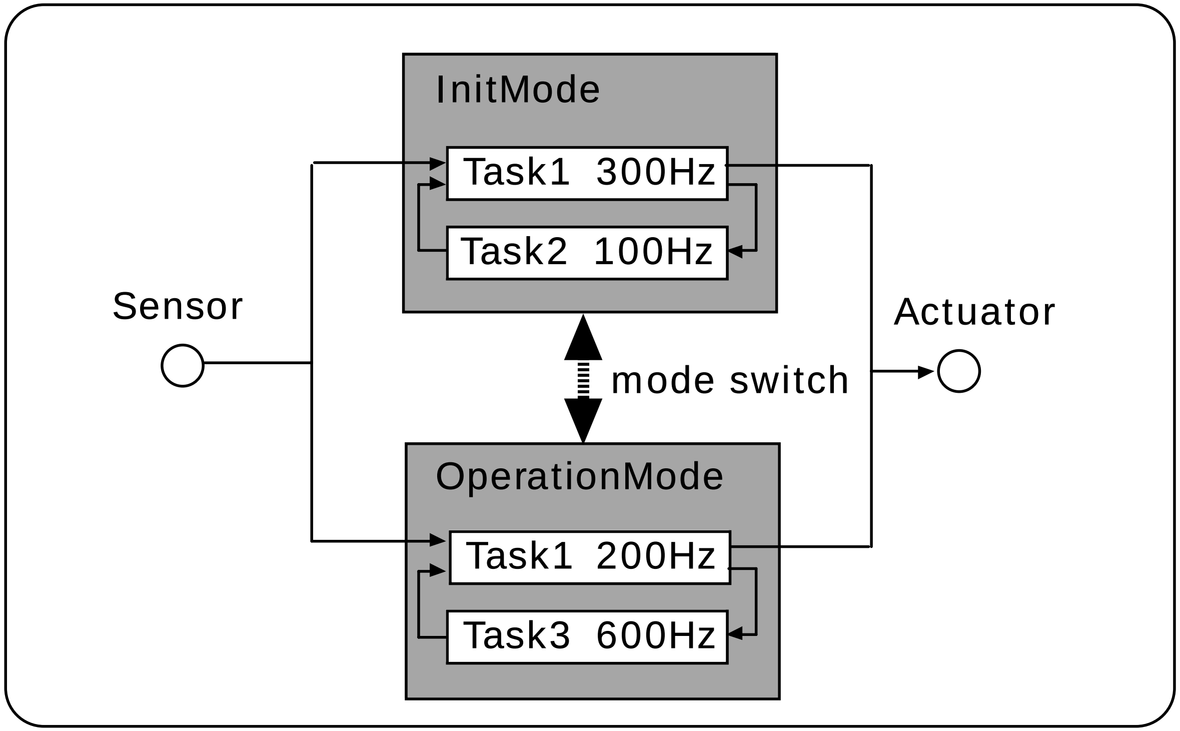
IDL defines an interface for a distributed application

=> Separates interface from implementation

TDL defines the timing for a real-time application

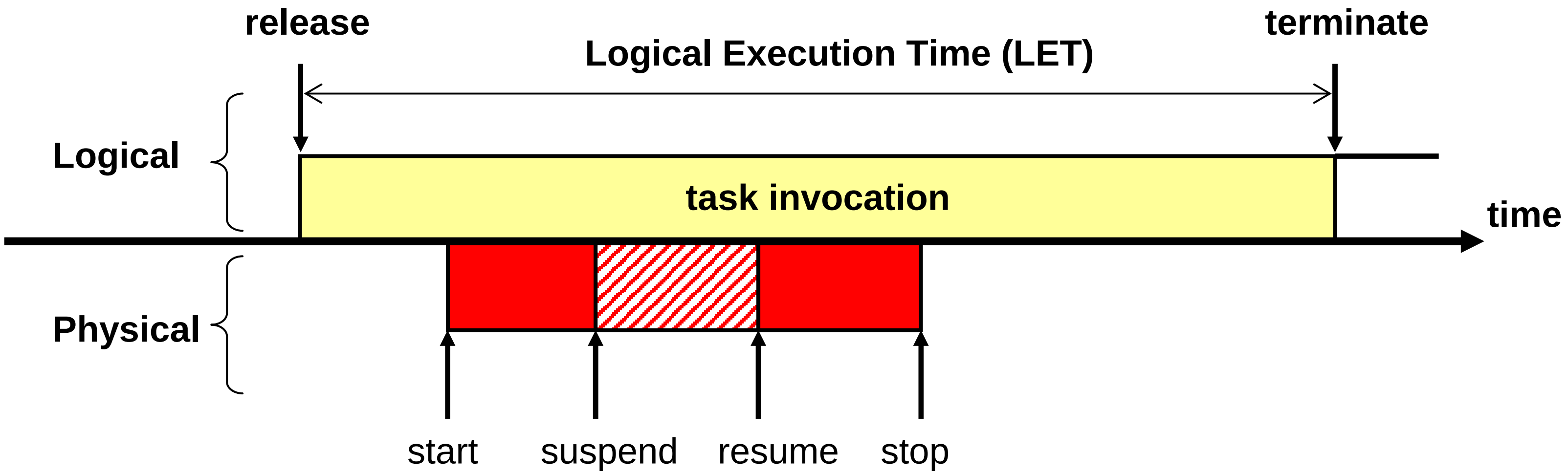
=> Separates timing from implementation

Schematic overview of Giotto/TDL concepts



Giotto programs are multi mode & multi rate systems for long running tasks.

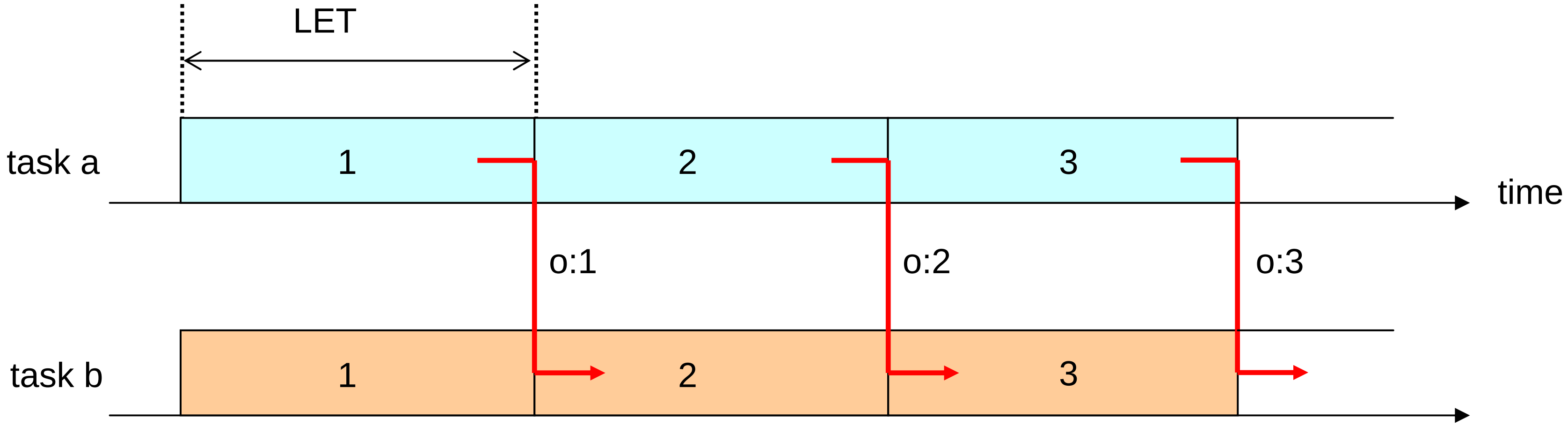
The Giotto/TDL Programming Model (LET)



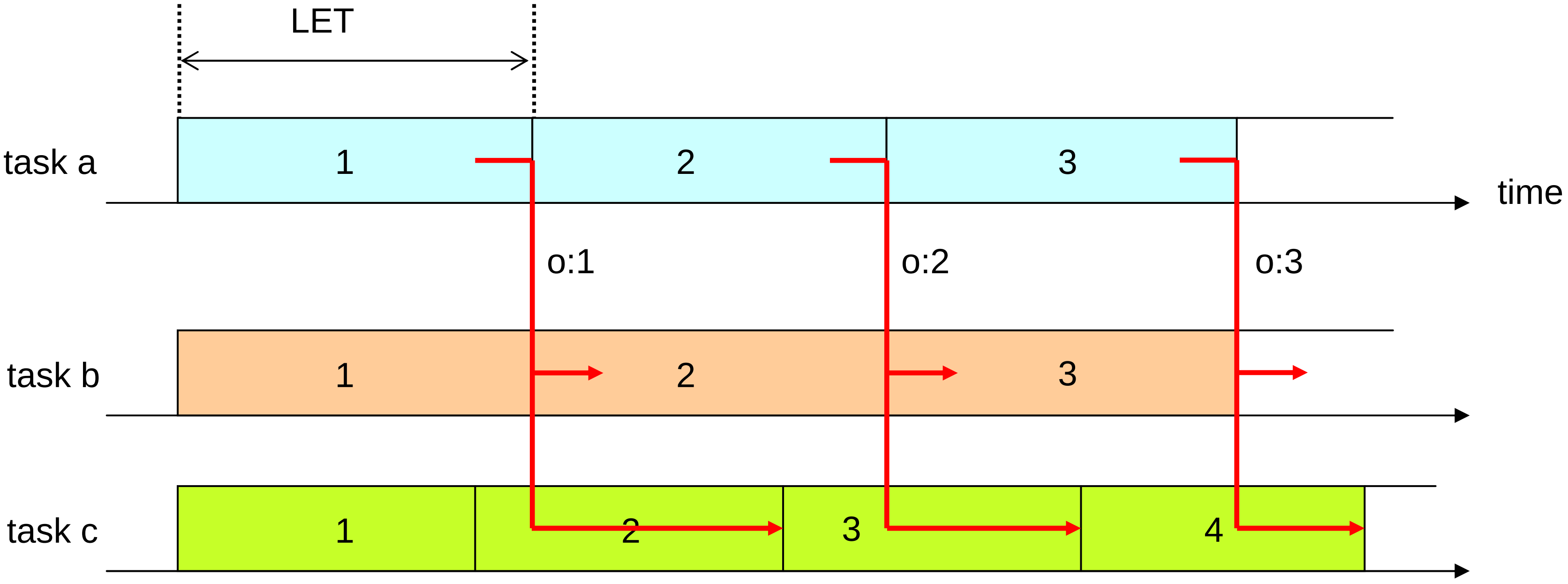
ET <= WCET <= LET

results are available at 'terminate'

Unit Delay



Unit Delay



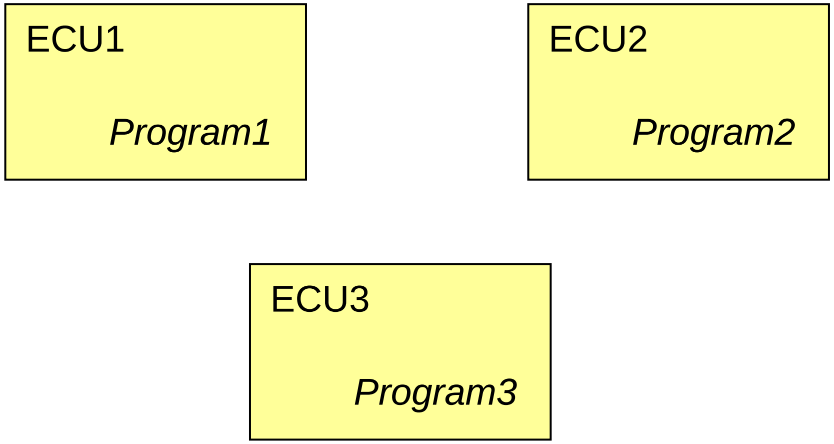
... but isn't it a waste of time?

=> determinism, composition, transparent distribution

Summary of Giotto Heritage

- Sensor and actuator ports are used to interact with the environment.
- A program is in one of potentially multiple modes.
- Every mode consists of periodic activities:
  - task invocations
  - actuator updates
  - mode switches
- A mode has a fixed period.
- Activities are carried out conditionally.
- Activities have their individual execution rate.
- Timing and interaction of activities follows LET semantics.

TDL Component Model: Motivation



ECU1

Program1

ECU2

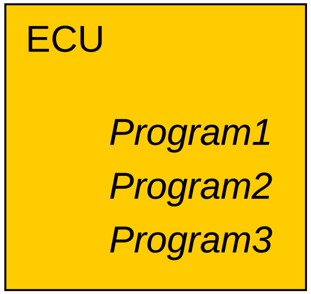
Program2

ECU3

Program3

- e.g. modern cars have up to 80 control units (ECUs)
- ECU consolidation is a topic
- run multiple programs on one ECU
- leads to TDL component model

TDL Component Model



ECU

Program1

Program2

Program3

- ProgramX is called a *module*
- modules may be independent
- modules may also refer to each other (DAG)
- modules can be used for multiple purposes

Usage of Modules

- decomposition of large programs
- grouping of unrelated modules
- parallel automatons
- ECU consolidation
- client/service relationship
  - provide common definitions for constants, types, etc.
  - data flow from service to client module
- distributed execution

TDL Syntax by Example

```
module M1 {

  sensor boolean s1 uses getS1;
  actuator int a1 uses setA1;

  public task inc [wcet=4ms] {
    output int o := 10;
    uses incImpl(o);
  }

  start mode main [period=10ms] {
    task
      [freq=1] inc();
    actuator
      [freq=2] a1 := inc.o;
    mode
      [freq=1] if exitMain(s1) then freeze;
  }

  mode freeze [period=1000ms] {}
}
```

Legend:

External functionality

Types

TDL Keywords

Annotations

Module Import

```
module M2{

  import M1;
  …
  task clientTask [wcet=10ms] {
    input int i1;
    …
  }
  mode main [period=100ms] {
    task [freq=1] clientTask(M1.inc.o);
    …
  }
}
```

- Import relationship forms a DAG.
- TDL supports structured module names (e.g. `com.avl.p1.M1`)
- import with rename: (e.g. `import com.avl.p1.M1 as A1;`)
- group import: (e.g. `import com.avl.p1 {M1, M2, M3};`)

More Language Constructs

- Constants

```
const c1 = 100;
const p = 100ms;
```

- Types

  Basic types: like Java

  `byte`, `short`, `int`, ...

  User defined opaque types: defined externally

```
type T;
```

Module Summary

- provides a named program component
- provides a name space
- allows for exporting sensors, constants, types, task outputs
- may be imported by other module(s)
- acts as unit of composition
- acts as the unit of loading
- acts as the unit of execution
- partitions the set of actuators
- acts as the unit of distribution

TDL supports multi mode & multi rate & multi program systems.

Differences to Giotto

- TDL provides a component model (module).
- TDL defines a concrete syntax and .ecode file format.
- TDL does not need explicit task invocation drivers, mode switch drivers and actuator update drivers as Giotto does.

  Drivers are defined *implicitly* by the TDL syntax and semantics.

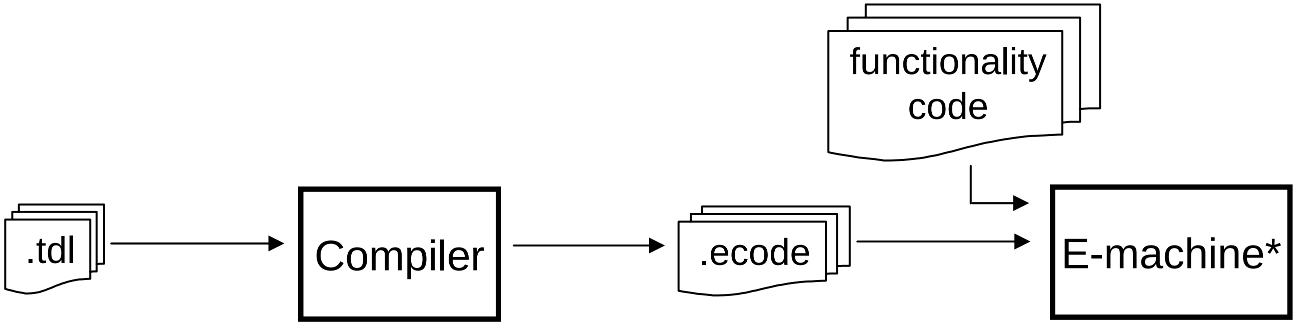
  The user needs to implement only guards, sensor getters, actuator setters, port initializers, and, of course, task functions.

Differences to Giotto

- TDL defines program start as mode switch to start mode.
- TDL disallows non-harmonic mode switches.
  - improved timeline logic -> determinism
  - easier compile time scheduling analysis
  - enables distributed mode-switches

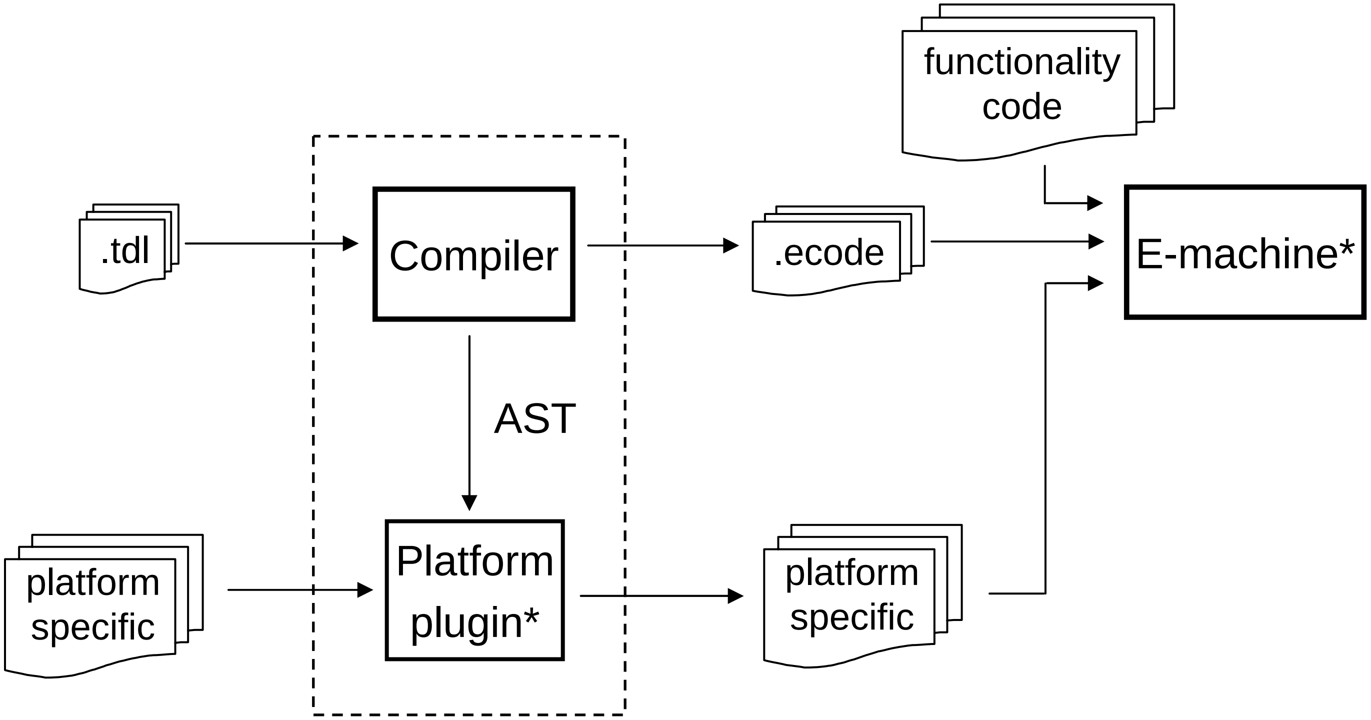
- Mode port assignments differ.
- Higher resolution timing: us.

Tool Chain Overview



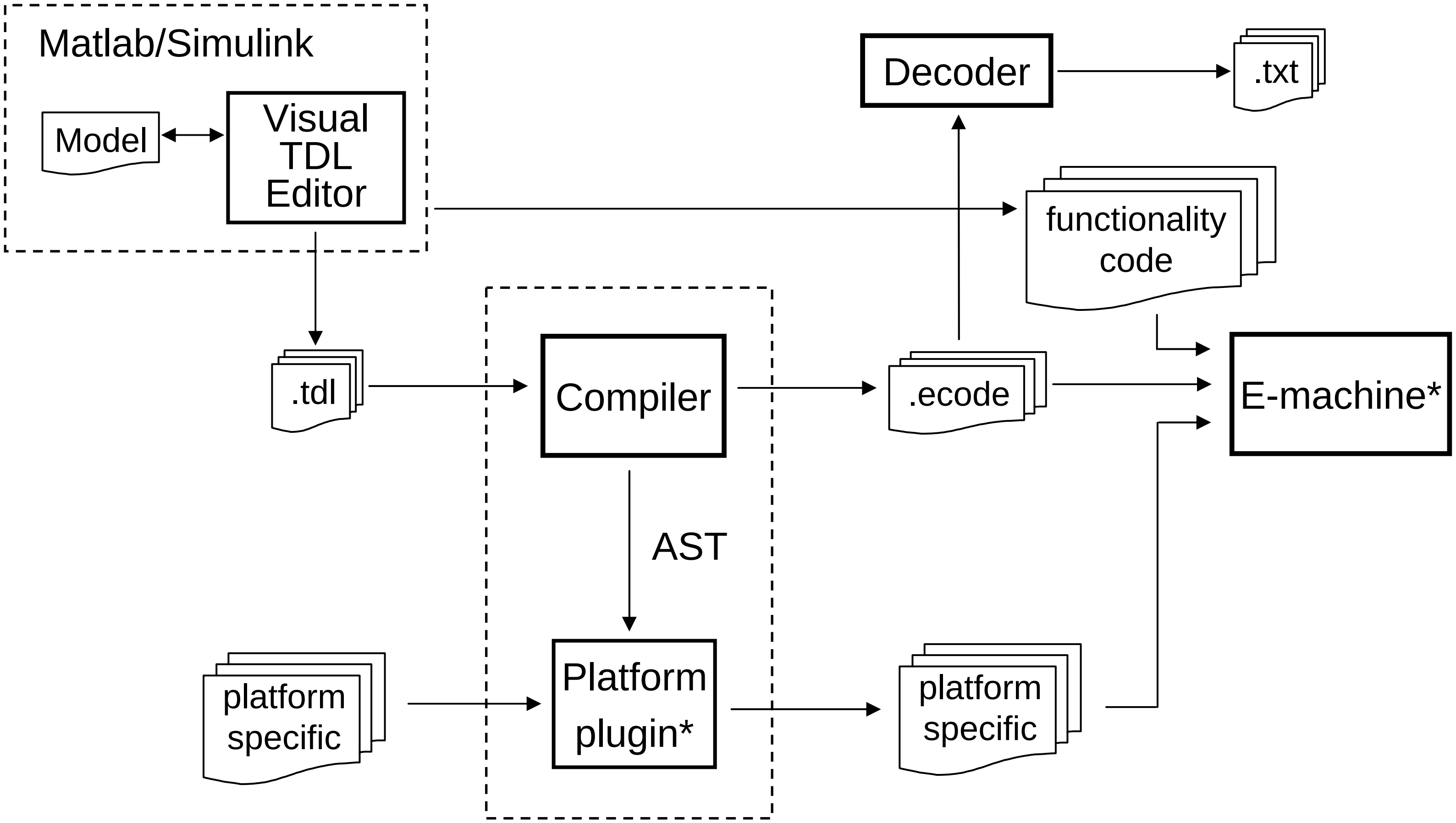
*Java, OSEK, InTIME, RTLinux, ...

Tool Chain Overview



*Java, OSEK, InTIME, RTLinux, ...

Tool Chain Overview



*Java, OSEK, InTIME, RTLinux, ...

Source Code Organization

`emcore`	(37.775 loc)
`ast`	abstract syntax tree (1.180)
`ecode`	ecode instructions and reader (613)
`scheduler`	node schedulers (1.039)
`tools`	(34.829)
`decode`	`.`ecode decoder (222)
`emachine`	E-machine (3.323)
`tdlc`	TDL compiler (5.248)
`platform`	standard platform plugins (2.261)
`vtdl`	visual TDL editor (24.198)
`busch`	bus scheduler (1.824)
`util`	various utility classes (114)

TDL Compiler

- implemented with compiler generator Coco/R for Java.

  (Mössenböck, JKU Linz)

  production quality recursive descent compiler in Java.

  2 phases:

  1. parse source text and build AST
  2. generate .ecode file from AST

- plugin interface defined by base class *Platform*
- plugin life cycle: `open {emitCode} close`
- additionally: `setErrorHandler`, `setDestDir`

Java-based E-machine

- used as proof of concept
- experimentation platform
- not hard-real time
- consists of
  - .ecode loader
  - task scheduler
  - E-code interpreter
  - dispatcher
  - bus controller (for distribution)
- Interacts with functionality code via drivers

State (as of 2004)

- Ready
  - TDL Compiler for complete TDL.
  - Decoder
  - Java-based E-machine for multiple modules.
  - Visual TDL Editor
  - InTIME, OSEK, TTA
  - TDK (from modecs.cc)
- Work in Progress
  - ANSI C back ends for POSIX, RTLinux, OSEK, InTIME…
  - E-machines for distribution
  - Bus Scheduler