TTool/DIPLODOCUS: a UML Model-Driven Environment for Embedded Systems-on-Chip Design

Institut Mines-Telecom
Telecom ParisTech
CNRS/LTCI
Sophia Antipolis, France

Andrea Enrici
Ludovic Apvrille
Renaud Pacalet

MODELS 2014 Demonstration session

Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users

Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users
- **System on Chip** = software and hardware components working together to perform a predefined set of functions

Research context: embedded Systems on Chip (SoCs)

- **Embedded system** = information processing systems embedded into a larger product, normally not directly perceivable by users
- **System on Chip** = software and hardware components working together to perform a predefined set of functions
- **We focus on:**
  - **data-dominated embedded systems** (e.g., signal, video processing)

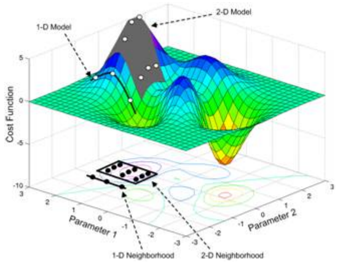
Our Design Challenge

Hardware/software partitioning: decide if a functionality should be implemented in hardware or in software or both

- **Design Space Exploration is the solution!**

Design Space Exploration (DSE)

Design Space Exploration = analyzing different implementations, that are functionally equivalent, in order to find an optimal solution according to given performance criteria (e.g., costs, area, power consumption)



Motivation

- **Embedded hardware has undergone a tremendous change:**
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores

Motivation

- **Embedded hardware has undergone a tremendous change:**
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores

- **so has done embedded software:**
  - from simple routines within control loops ...
  - ... to applications with video conferencing and voice recognition

Motivation

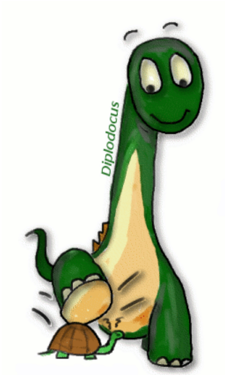
- **Embedded hardware has undergone a tremendous change:**
  - from single processor architectures ...
  - ... to on-chip cloud computers with tens or hundreds of cores
- **so has done embedded software:**
  - from simple routines within control loops ...
  - ... to applications with video conferencing and voice recognition

- **It is now impossible to design new products from scratch!**

A toolkit named TTool (say "tea-tool")

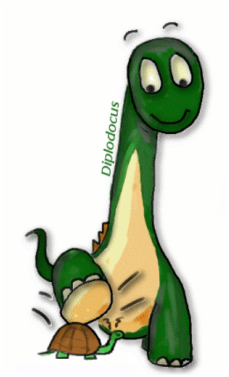
TTool/DIPLODOCUS in a nutshell

- **UML/SysML diagrams to model an embedded system**'s functionality, communication services and architecture



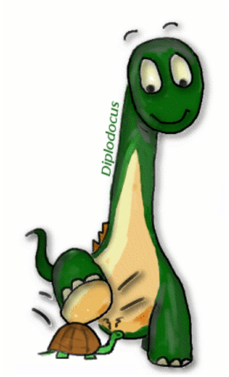
TTool/DIPLODOCUS in a nutshell

- **UML/SysML diagrams to model an embedded system**'s functionality, communication services and architecture
- **Simulate and verify formally** the system's models **at the push of a button**



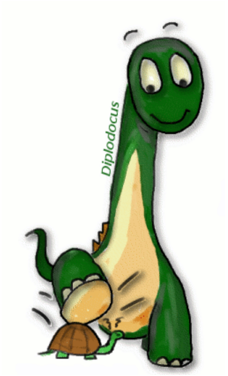
TTool/DIPLODOCUS in a nutshell

- **UML/SysML diagrams to model an embedded system**'s functionality, communication services and architecture
- **Simulate and verify formally** the system's models **at the push of a button**
- **No need to be an expert** in modeling, simulation or formal verification



TTool/DIPLODOCUS in a nutshell

- **UML/SysML diagrams to model an embedded system**'s functionality, communication services and architecture
- **Simulate and verify formally** the system's models **at the push of a button**
- **No need to be an expert** in modeling, simulation or formal verification
- **Let's model** a signal processing algorithm running on an embedded System-on-Chip!



High-level models ... what for?

- **Increase simulation speed**
  - the more models contain details, the longer are the simulation and verification

High-level models ... what for?

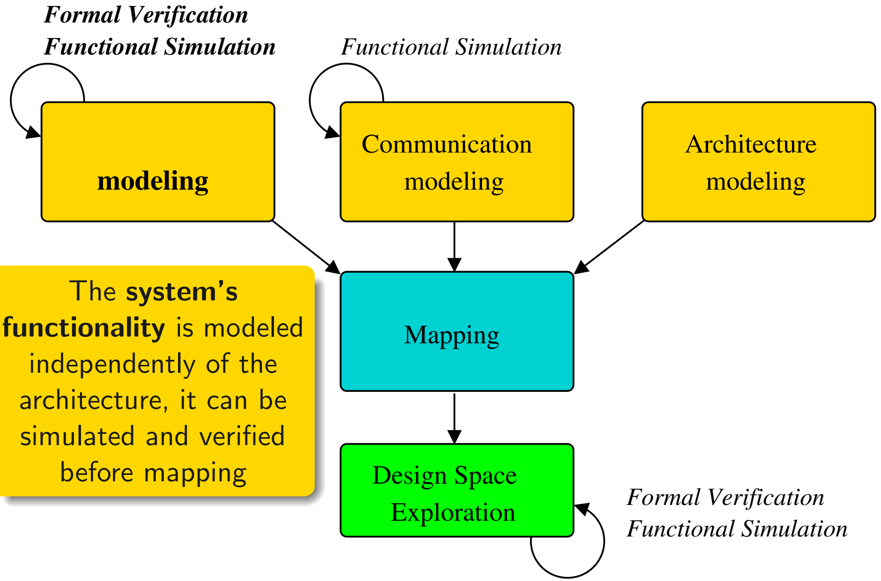
- **Increase simulation speed**
  - the more models contain details, the longer are the simulation and verification
- **Reduce design and testing effort**
  - the earlier the bugs are found, the less it costs to fix them

High-level models ... what for?

- **Increase simulation speed**
  - the more models contain details, the longer are the simulation and verification
- **Reduce design and testing effort**
  - the earlier the bugs are found, the less it costs to fix them
- **Increase portability**
  - save time and money by re-adapting models for next projects

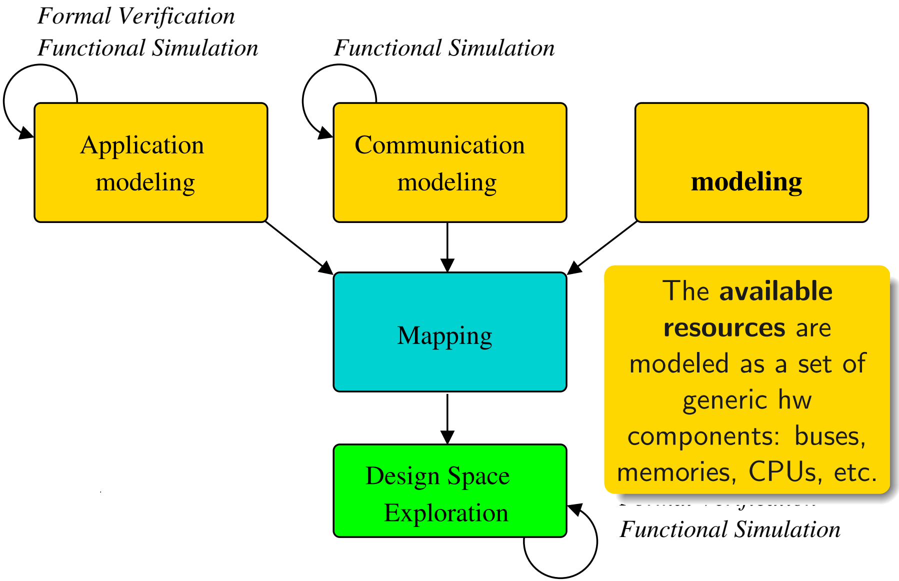
The Ψ-chart design methodology

The Ψ-chart design methodology



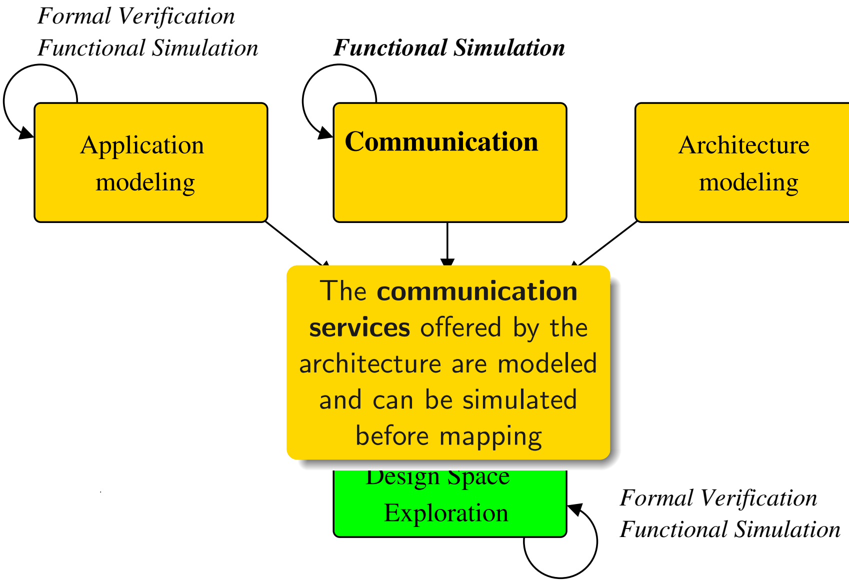
The **system's functionality** is modeled independently of the architecture, it can be simulated and verified before mapping

The Ψ-chart design methodology



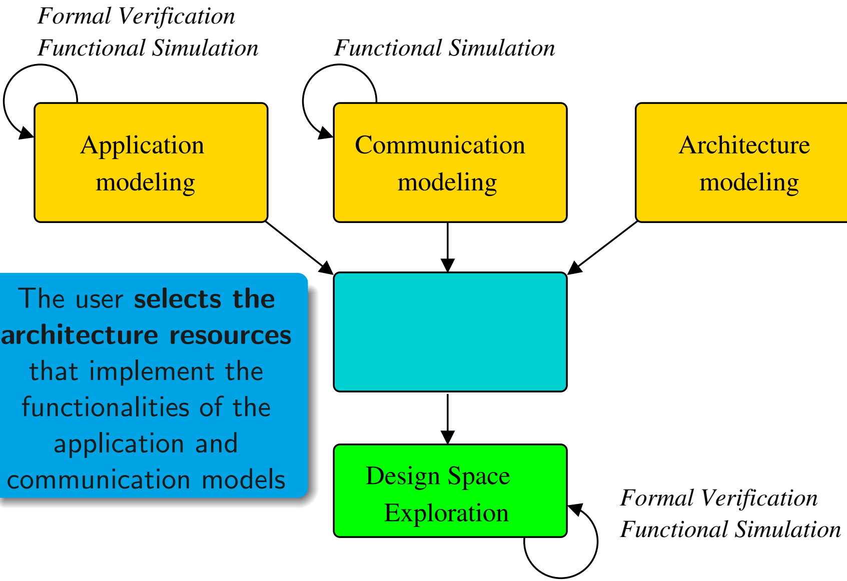
The **available resources** are modeled as a set of generic hw components: buses, memories, CPUs, etc.

The Ψ-chart design methodology



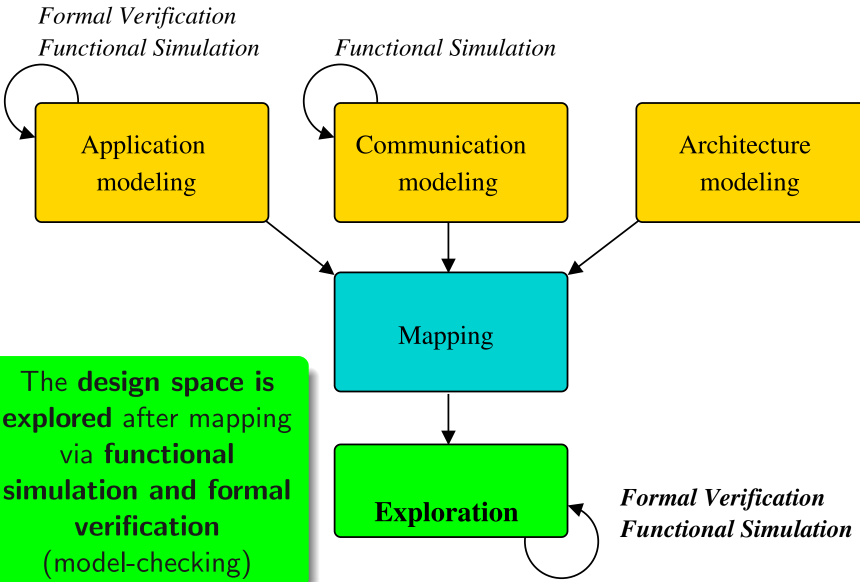
The **communication services** offered by the architecture are modeled and can be simulated before mapping

The Ψ-chart design methodology



The user **selects the architecture resources** that implement the functionalities of the application and communication models

The Ψ-chart design methodology

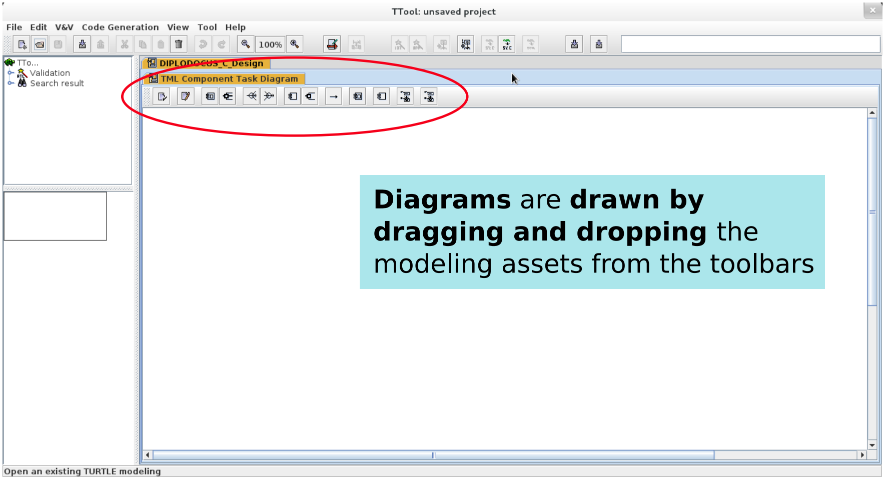


The **design space is explored** after mapping via **functional simulation and formal verification** (model-checking)

Let’s model with TTool/DIPLODOCUS!

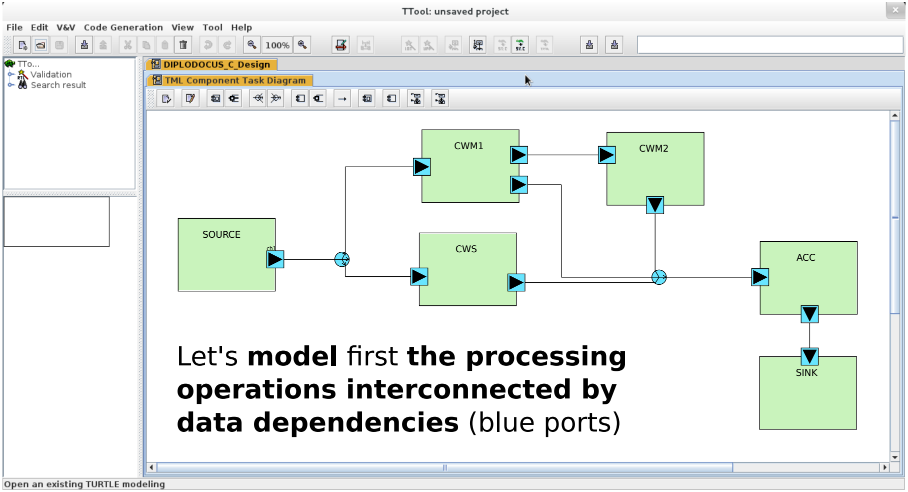
Let’s model a signal processing algorithm that runs on an embedded System-on-Chip!

Let’s model with TTool/DIPLODOCUS!


TTool: unsaved project
File Edit V&V Code Generation View Tool Help
100%
TTo...
Validation
Search result
DIPLODOCUS_C_Design
TML Component Task Diagram
Open an existing TURTLE modeling

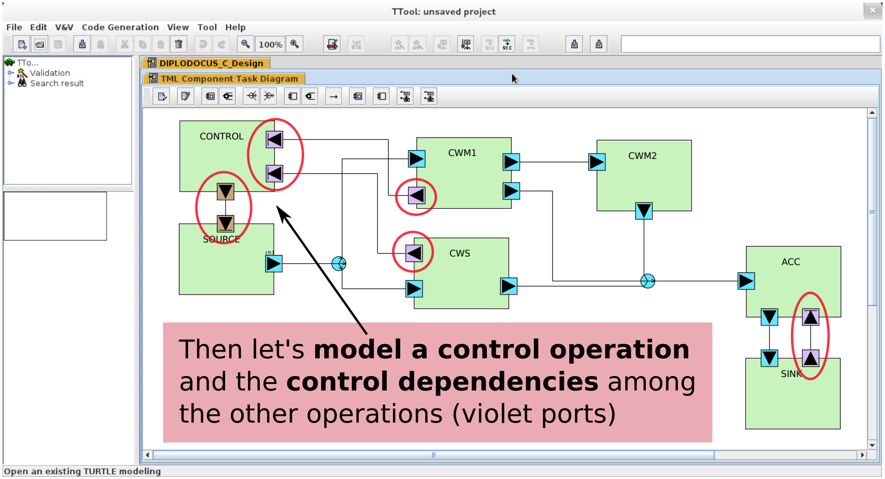
Diagrams are **drawn by dragging and dropping** the modeling assets from the toolbars

Application modeling


TTool: unsaved project
File Edit V&V Code Generation View Tool Help
100%
TTo...
Validation
Search result
DIPLODOCUS_C_Design
TML Component Task Diagram
CWM1
CWM2
SOURCE
ch1
CWS
ACC
SINK
Open an existing TURTLE modeling

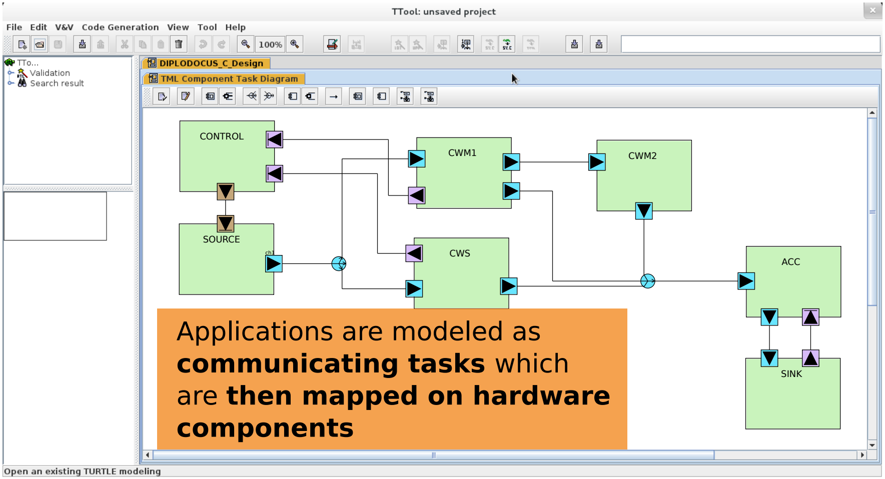
Let's **model** first **the processing operations interconnected by data dependencies** (blue ports)

Application modeling



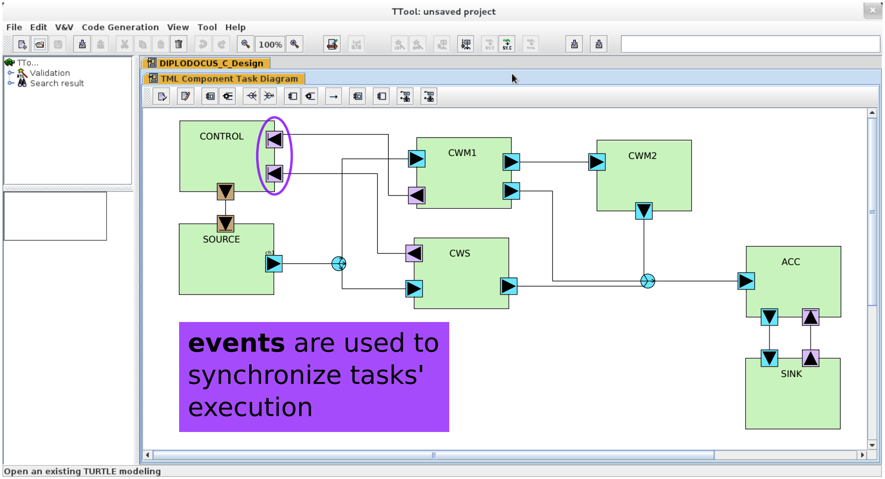
Then let's **model a control operation** and the **control dependencies** among the other operations (violet ports)

Application modeling



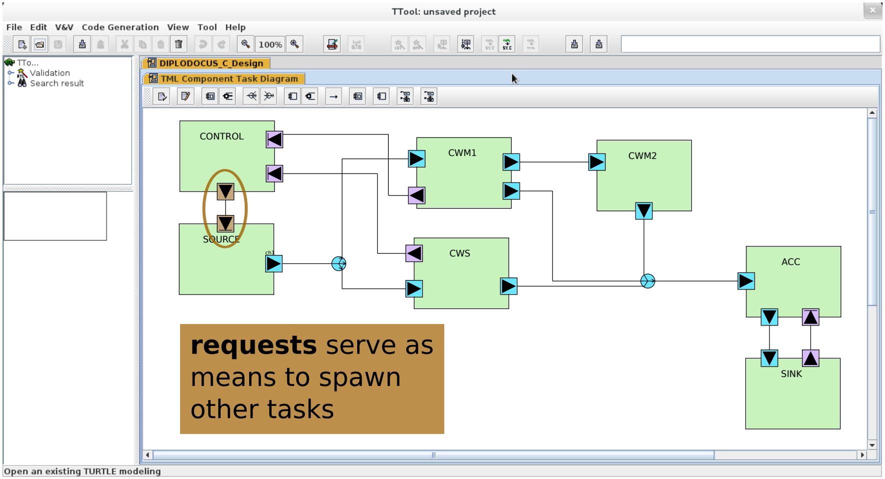
Applications are modeled as **communicating tasks** which are **then mapped on hardware components**

Application modeling



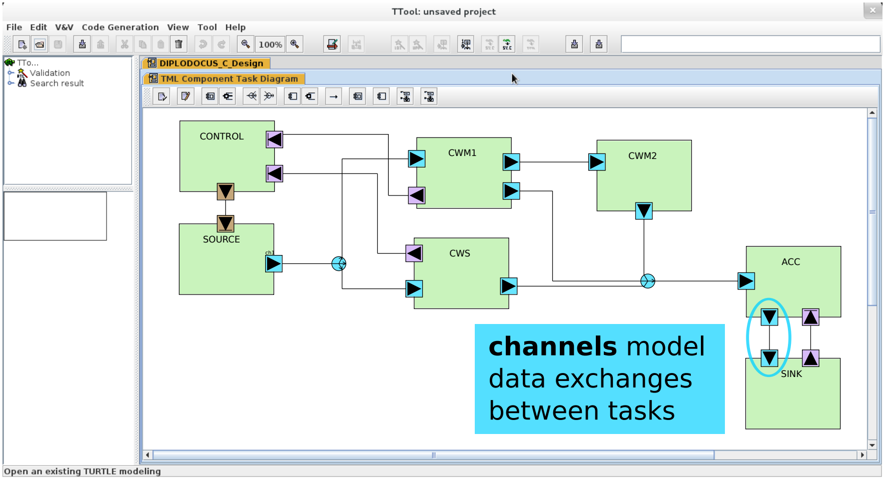
events are used to synchronize tasks' execution

Application modeling



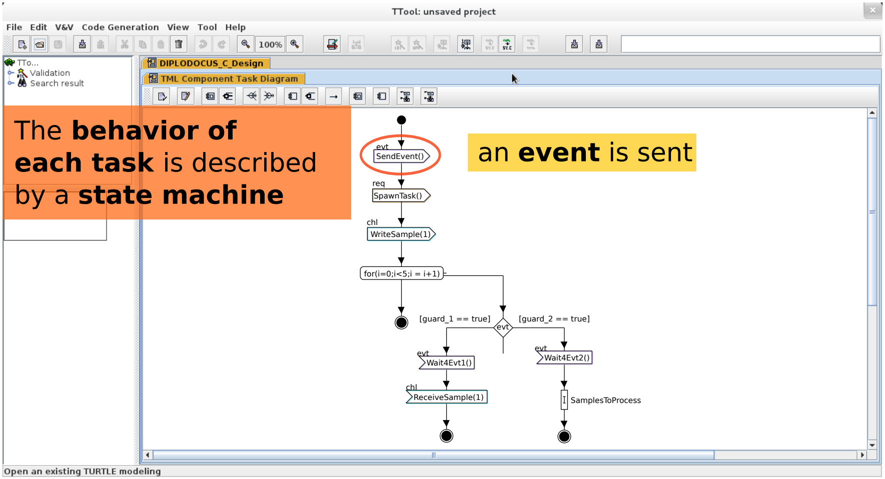
requests serve as means to spawn other tasks

Application modeling



channels model data exchanges between tasks

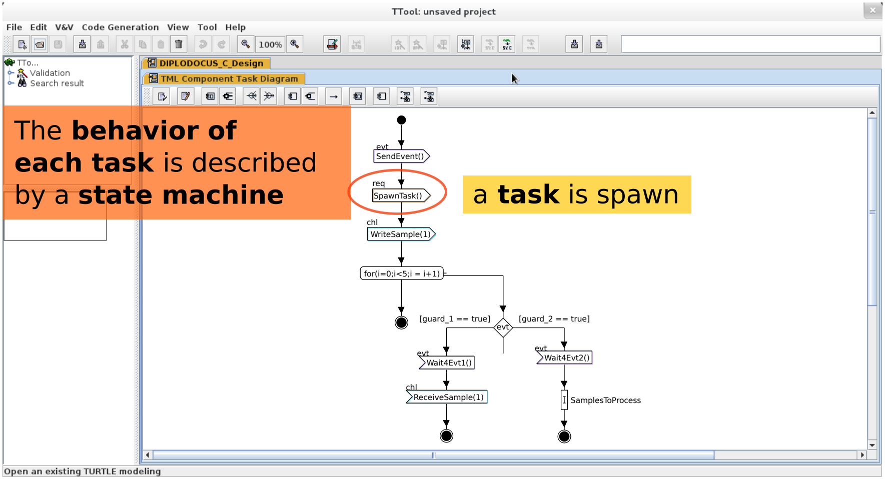
Application modeling



The **behavior of each task** is described by a **state machine**

an **event** is sent

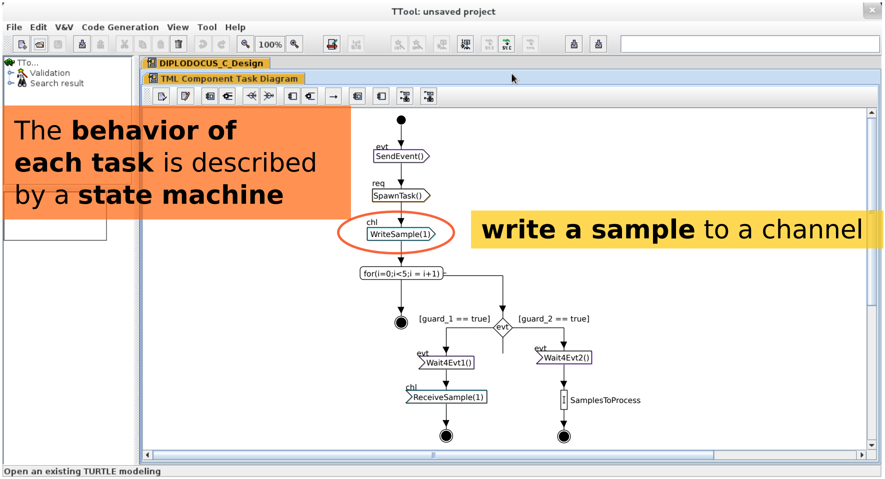
Application modeling



The **behavior of each task** is described by a **state machine**

a **task** is spawn

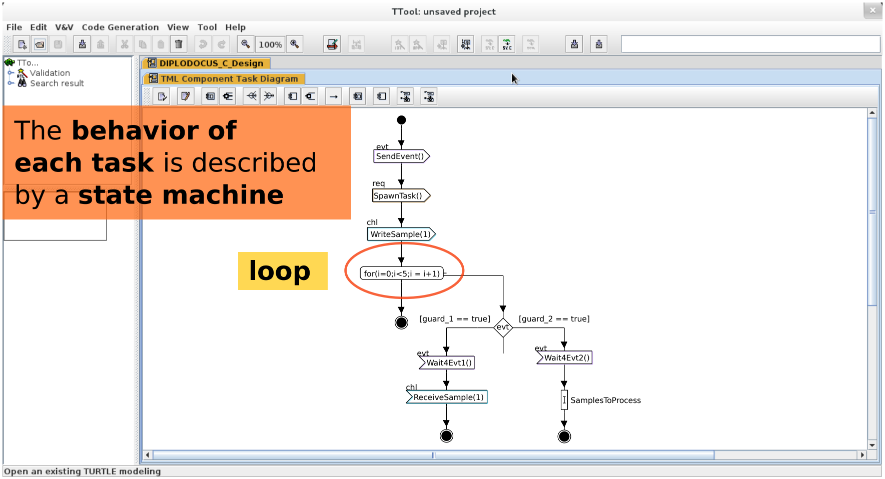
Application modeling



The **behavior of each task** is described by a **state machine**

write a sample to a channel

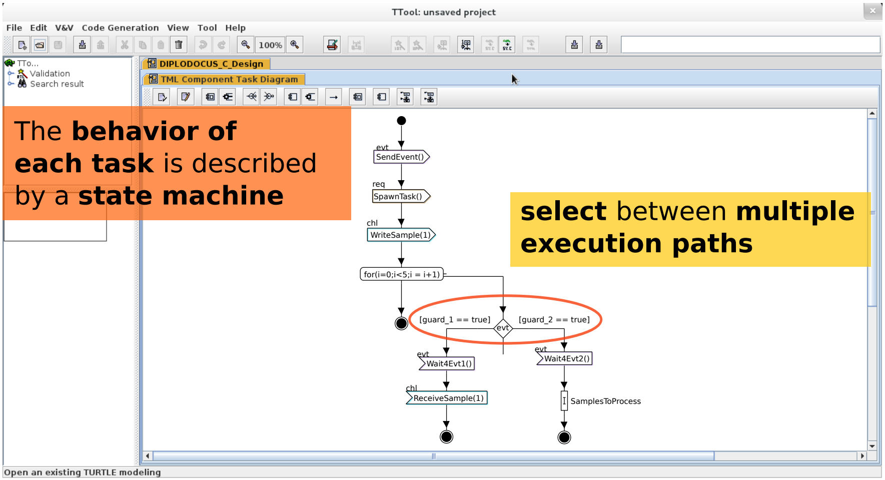
Application modeling



The **behavior of each task** is described by a **state machine**

loop

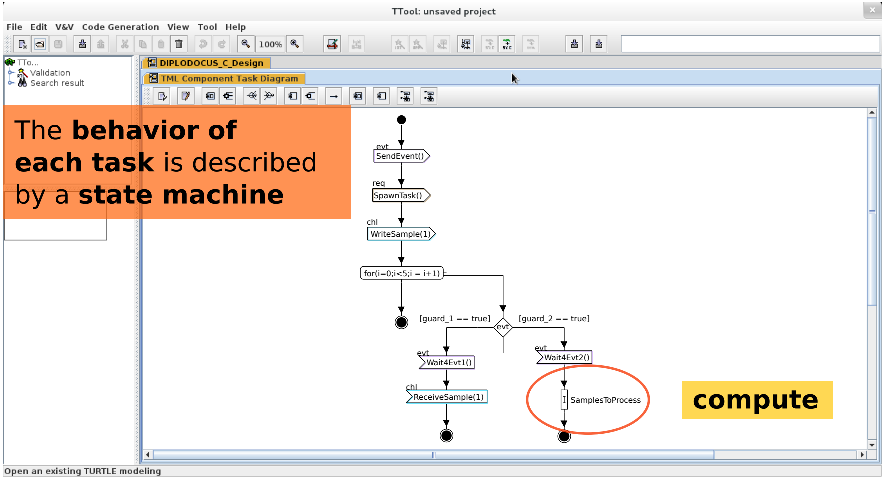
Application modeling



The **behavior of each task** is described by a **state machine**

select between **multiple execution paths**

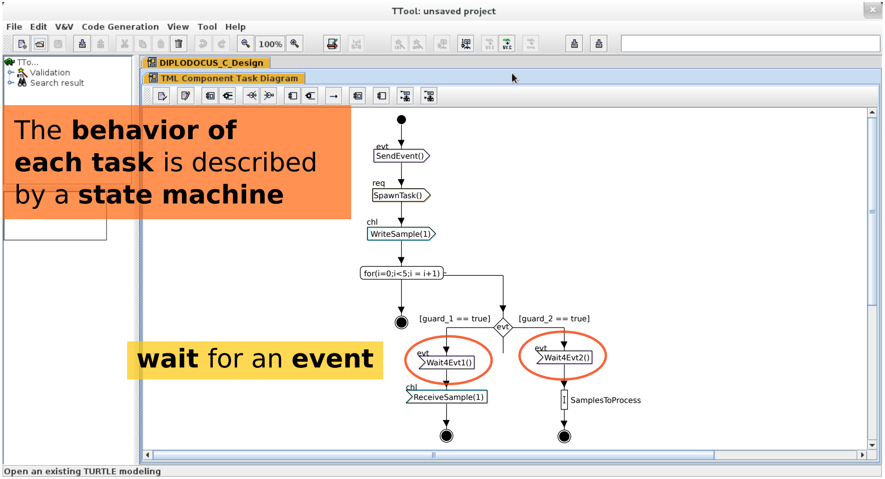
Application modeling



The **behavior of each task** is described by a **state machine**

compute

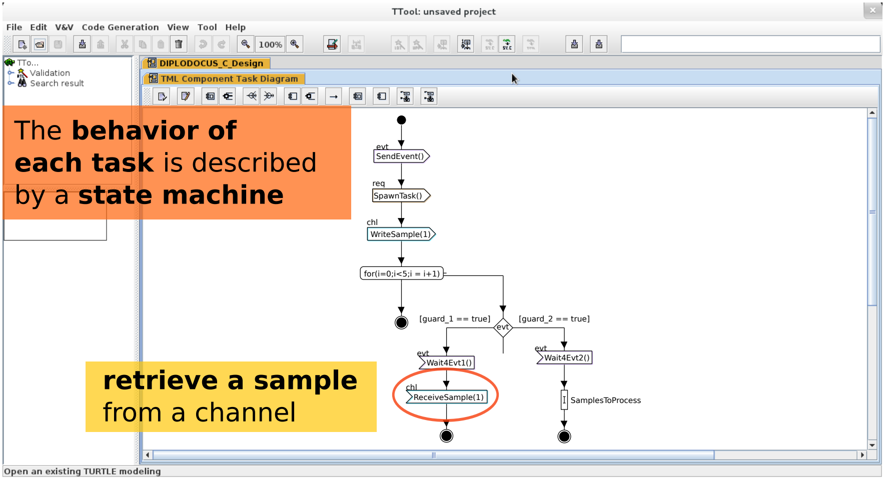
Application modeling



The **behavior of each task** is described by a **state machine**

wait for an **event**

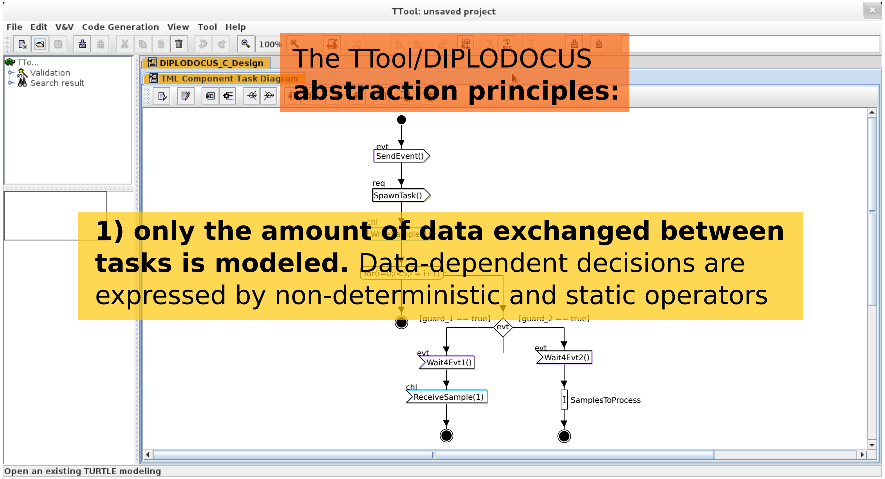
Application modeling



The **behavior of each task** is described by a **state machine**

retrieve a sample from a channel

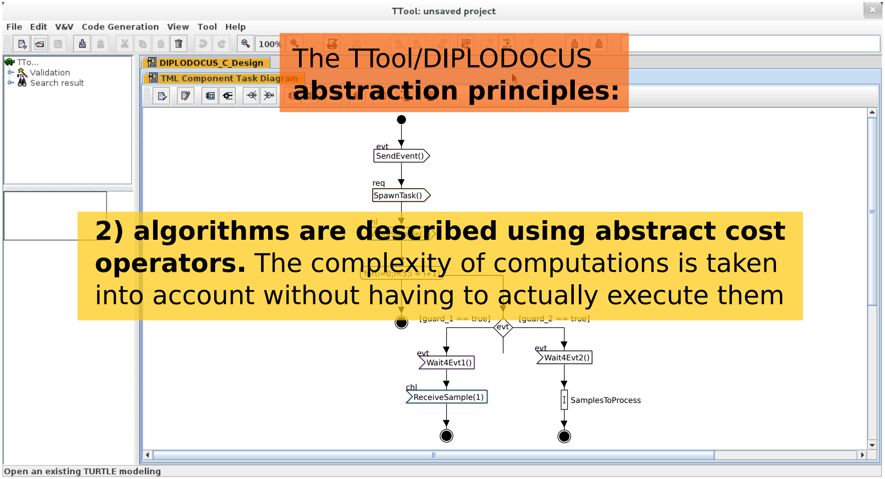
Application modeling



The TTool/DIPLODOCUS **abstraction principles:**

1) only the amount of data exchanged between tasks is modeled. Data-dependent decisions are expressed by non-deterministic and static operators

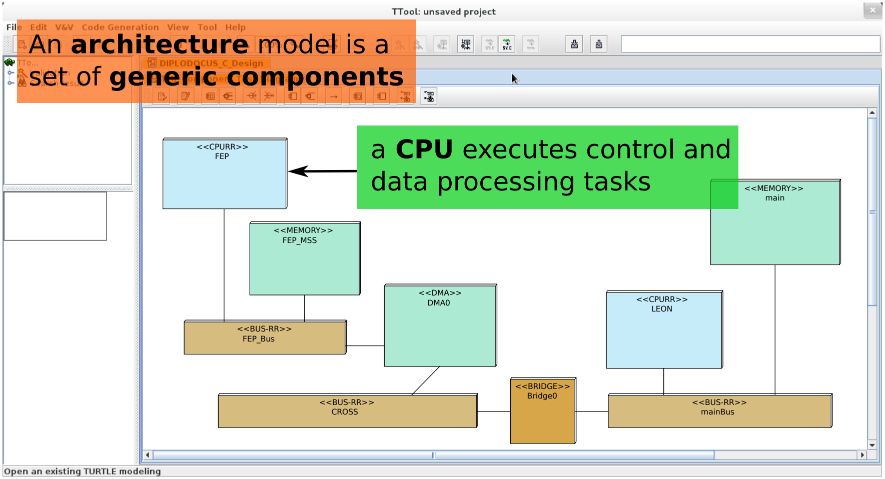
Application modeling



The TTool/DIPLODOCUS **abstraction principles:**

2) algorithms are described using abstract cost operators. The complexity of computations is taken into account without having to actually execute them

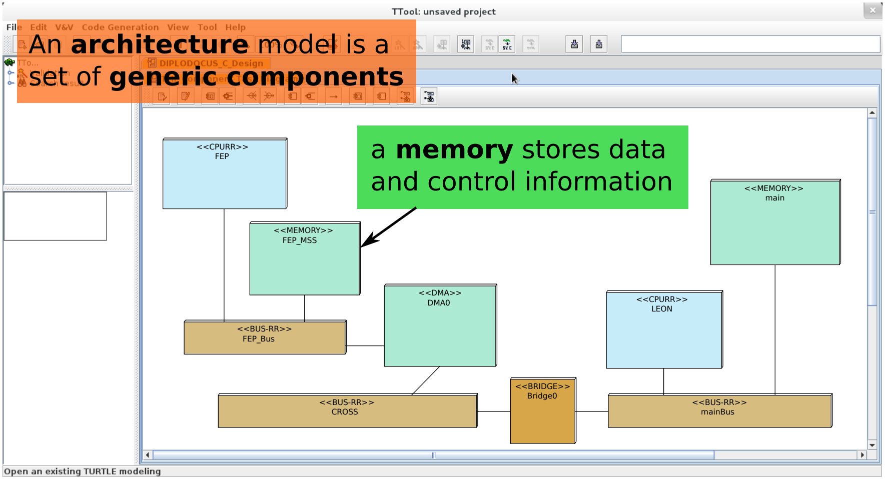
Architecture modeling



An **architecture** model is a set of **generic components**

a **CPU** executes control and data processing tasks

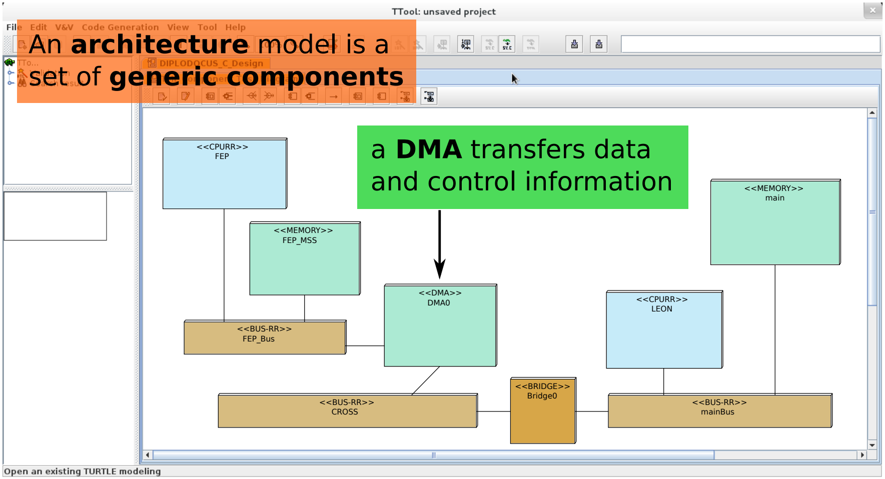
Architecture modeling



An **architecture** model is a set of **generic components**

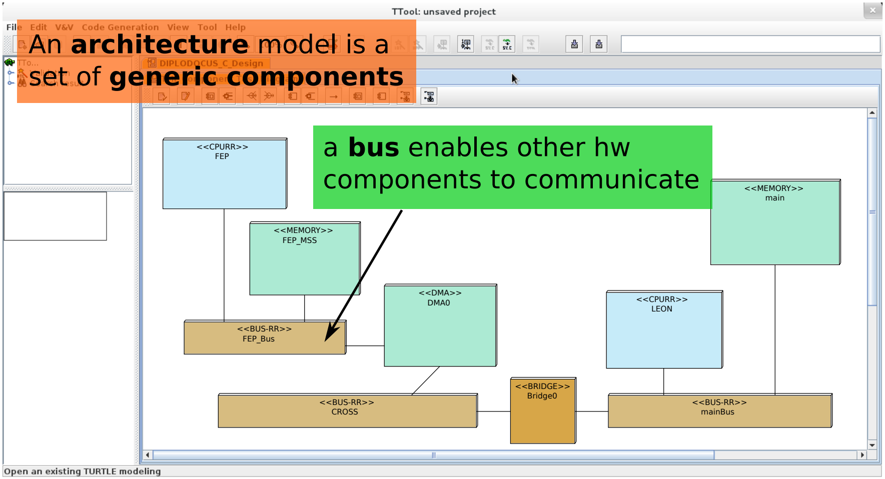
a **memory** stores data and control information

Architecture modeling


An **architecture** model is a set of **generic components**

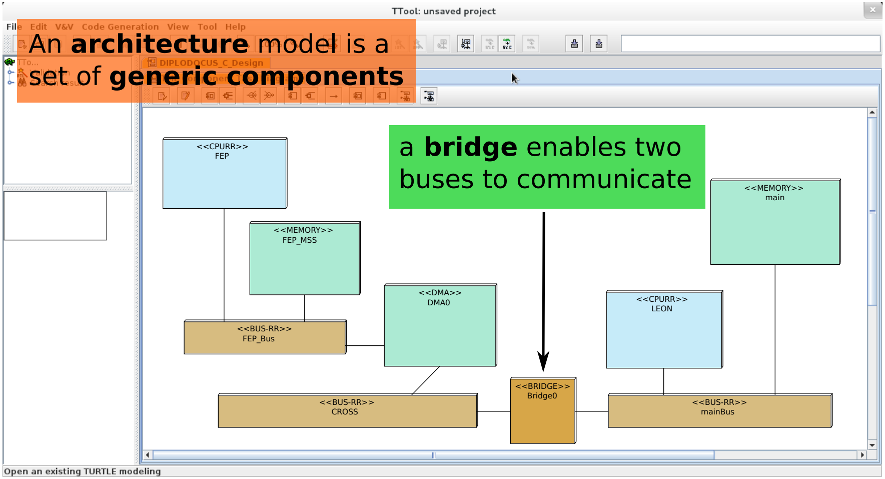
a **DMA** transfers data and control information

Architecture modeling


An **architecture** model is a set of **generic components**

a **bus** enables other hw components to communicate

Architecture modeling

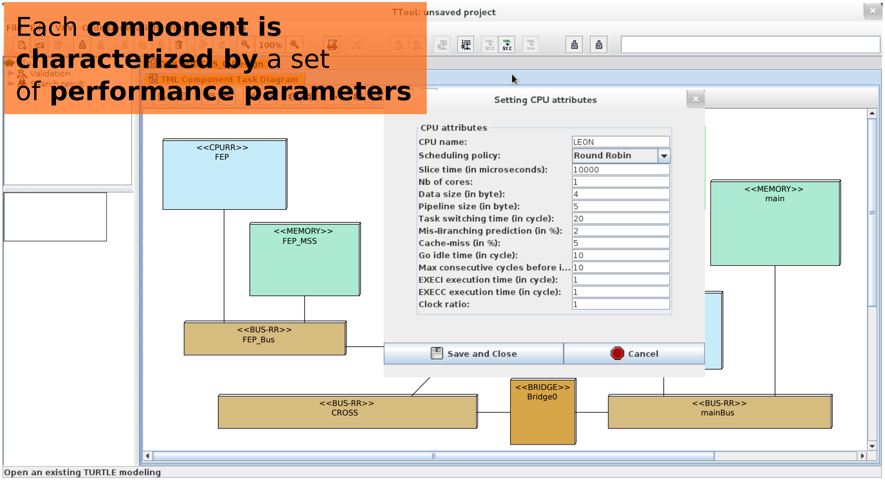


An **architecture** model is a set of **generic components**

a **bridge** enables two buses to communicate

Architecture modeling

Each **component is characterized by** a set of **performance parameters**

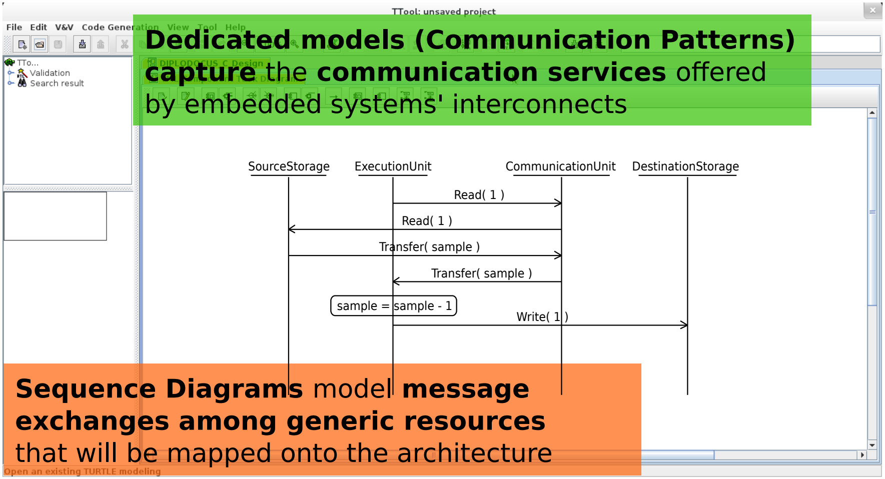


Setting CPU attributes

CPU attributes

CPU name:	LEON
Scheduling policy:	Round Robin
Slice time (in microseconds):	10000
Nb of cores:	1
Data size (in byte):	4
Pipeline size (in byte):	5
Task switching time (in cycle):	20
Mis-Branching prediction (in %):	2
Cache-miss (in %):	5
Go idle time (in cycle):	10
Max consecutive cycles before i...	10
EXECI execution time (in cycle):	1
EXECC execution time (in cycle):	1
Clock ratio:	1

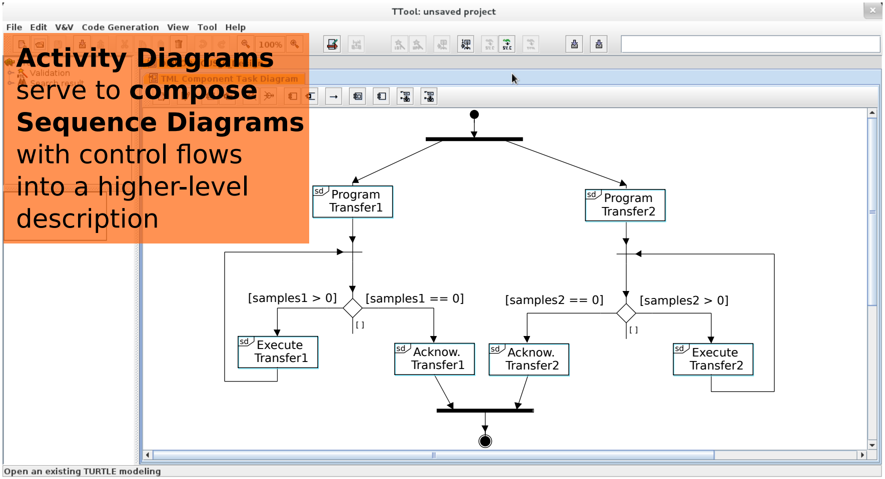
Communication modeling



Dedicated models (Communication Patterns) capture the **communication services** offered by embedded systems' interconnects

Sequence Diagrams model **message exchanges among generic resources** that will be mapped onto the architecture

Communication modeling

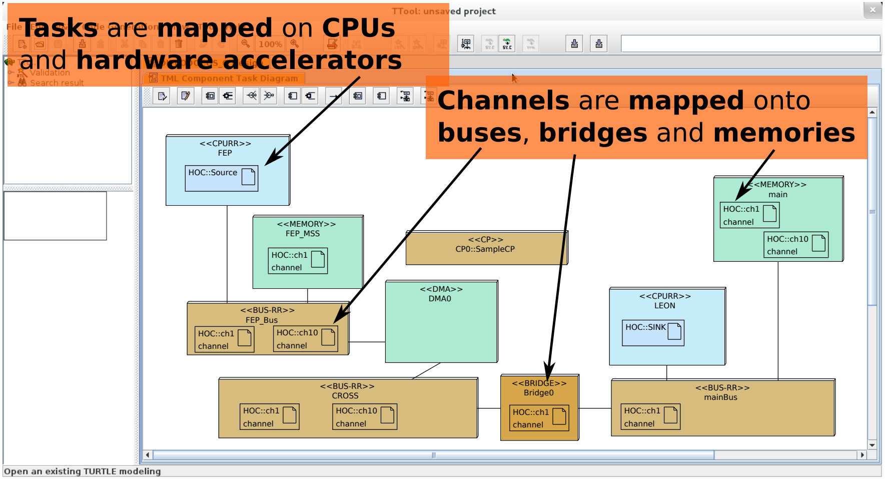


Activity Diagrams serve to **compose Sequence Diagrams** with control flows into a higher-level description

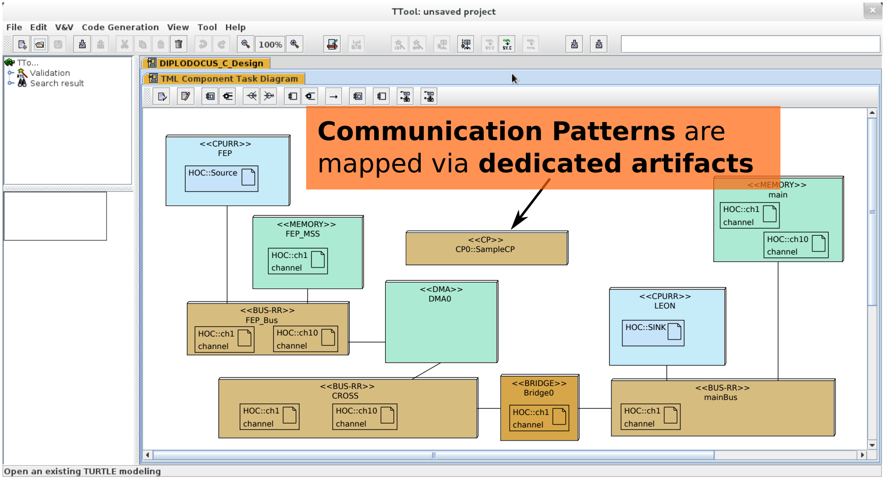
Mapping

Tasks are **mapped** on **CPUs** and **hardware accelerators**

Channels are **mapped** onto **buses**, **bridges** and **memories**

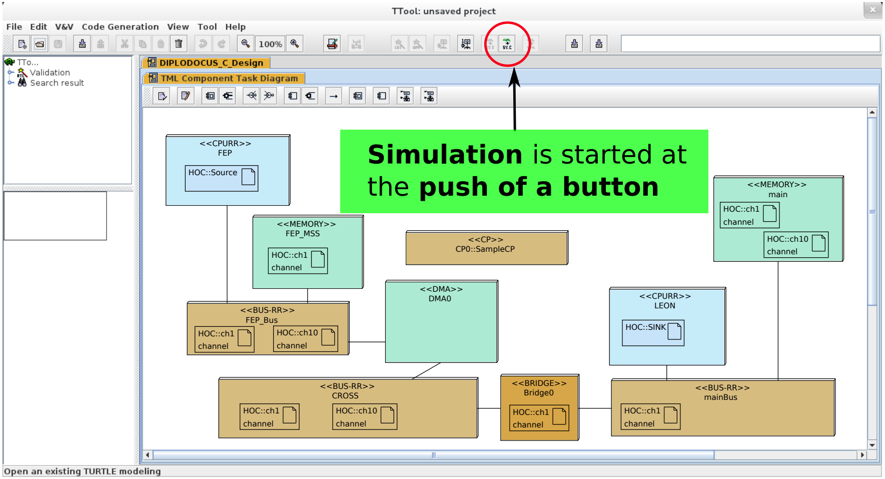


Mapping



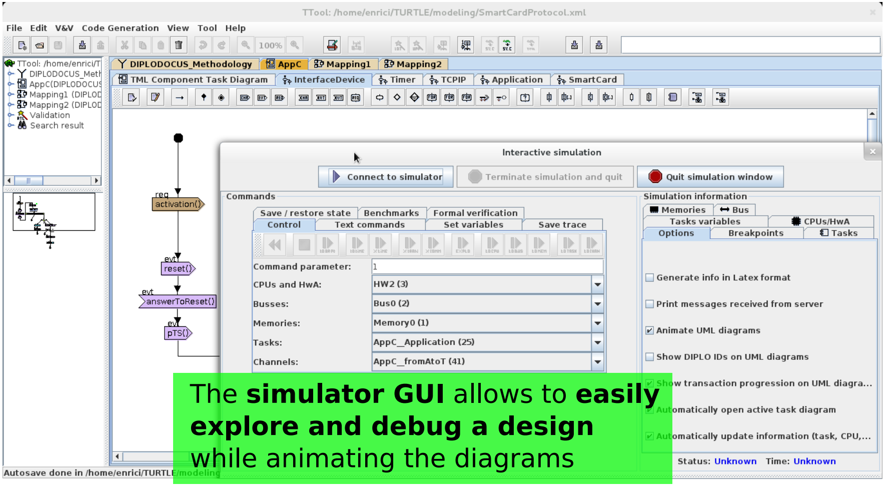
Communication Patterns are mapped via **dedicated artifacts**

Functional Simulation



Simulation is started at the **push of a button**

Functional Simulation



Command parameter: 1
CPUs and HwA: HW2 (3)
Busses: Bus0 (2)
Memories: Memory0 (1)
Tasks: AppC_Application (25)
Channels: AppC_fromAtoT (41)

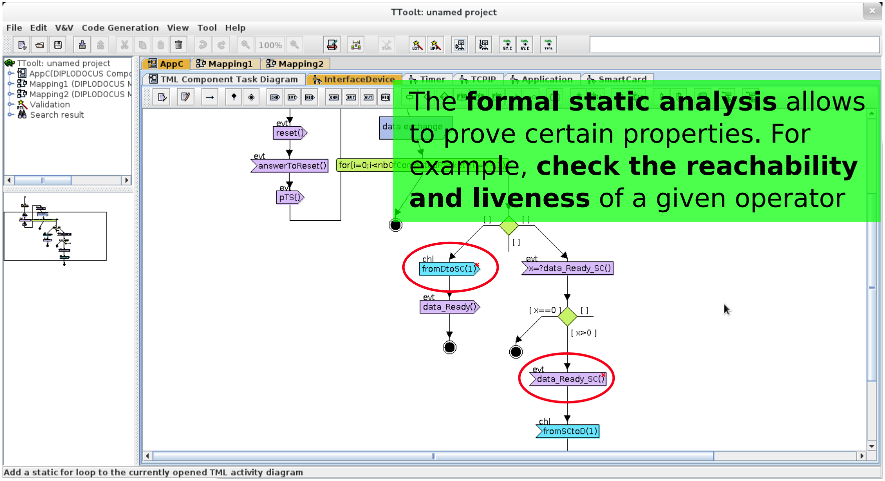
- [ ] Generate info in Latex format
- [ ] Print messages received from server
- [x] Animate UML diagrams
- [ ] Show DIPLO IDs on UML diagrams
- [x] Show transaction progression on UML diagra...
- [x] Automatically open active task diagram
- [x] Automatically update information (task, CPU,...

Status: Unknown Time: Unknown

Autosave done in /home/enrici/TURTLE/modeling

The **simulator GUI** allows to **easily explore and debug a design** while animating the diagrams

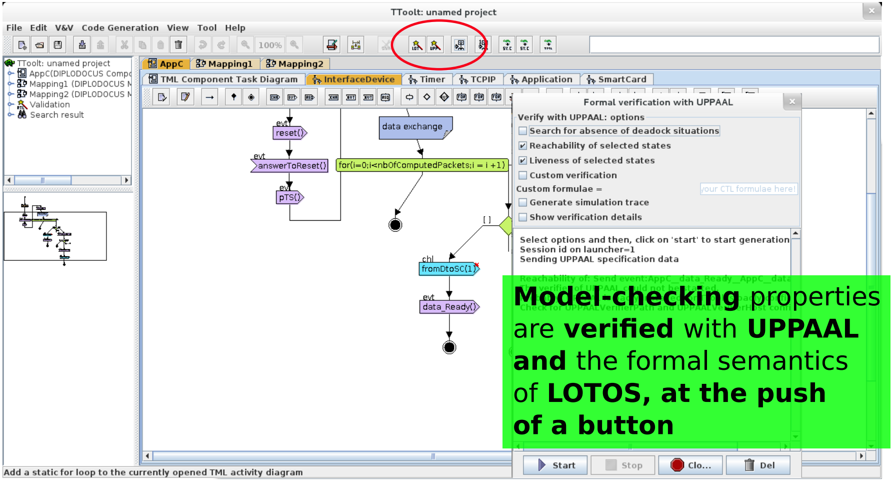
Formal Verification



The **formal static analysis** allows to prove certain properties. For example, **check the reachability and liveness** of a given operator

Add a static for loop to the currently opened TML activity diagram

Formal Verification



Formal verification with UPPAAL

Verify with UPPAAL: options
- [ ] Search for absence of deadock situations
- [x] Reachability of selected states
- [x] Liveness of selected states
- [ ] Custom verification

Custom formulae = your CTL formulae here!
- [ ] Generate simulation trace
- [ ] Show verification details

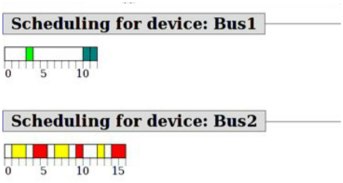
```
Select options and then, click on 'start' to start generation
Session id on launcher=1
Sending UPPAAL specification data
```

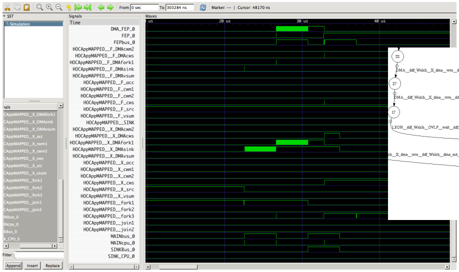
Add a static for loop to the currently opened TML activity diagram

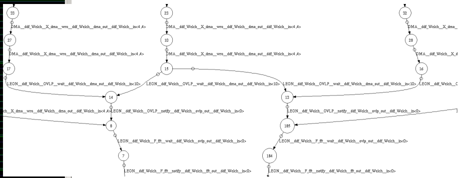
Model-checking properties are **verified** with **UPPAAL and** the formal semantics of **LOTOS, at the push of a button**

Debugging facilities

- **Debugging and design exploration facilities** are available from simulation and verification:
  - **Gantt charts**
  - **execution waveforms**
  - **reachability graphs**







Summary

An environment for the design of Systems-on-Chip:

- easy and rapid to deploy, open source

Summary

An environment for the design of Systems-on-Chip:

- easy and rapid to deploy, open source
- used by industrial and academic partners ...
  - Texas Instruments
  - Freescale
  - ISAE (Toulouse, France)

Summary

An environment for the design of Systems-on-Chip:

- easy and rapid to deploy, open source
- used by industrial and academic partners ...
  - Texas Instruments
  - Freescale
  - ISAE (Toulouse, France)
- ... and projects
  - EVITA (automotive embedded systems)
  - Embb (Software-Defined Radio)

Future works and research directions

Integrate **model transformations:**

- **into a code generation environment** to produce the application control code

Future works and research directions

Integrate **model transformations:**

- **into a code generation environment** to produce the application control code
- **into a run-time environment** for application scheduling and memory management

Do you want to now more?

Visit TTool's website:

- http://ttool.telecom-paristech.fr/

Contact us:

- andrea.enrici@telecom-paristech.fr
- ludovic.apvrille@telecom-paristech.fr
- renaud.pacalet@telecom-paristech.fr