SmartSoft MDSD Toolchain

Leuven 2009-07-09

Andreas Steck

Computer Science Department
University of Applied Sciences Ulm

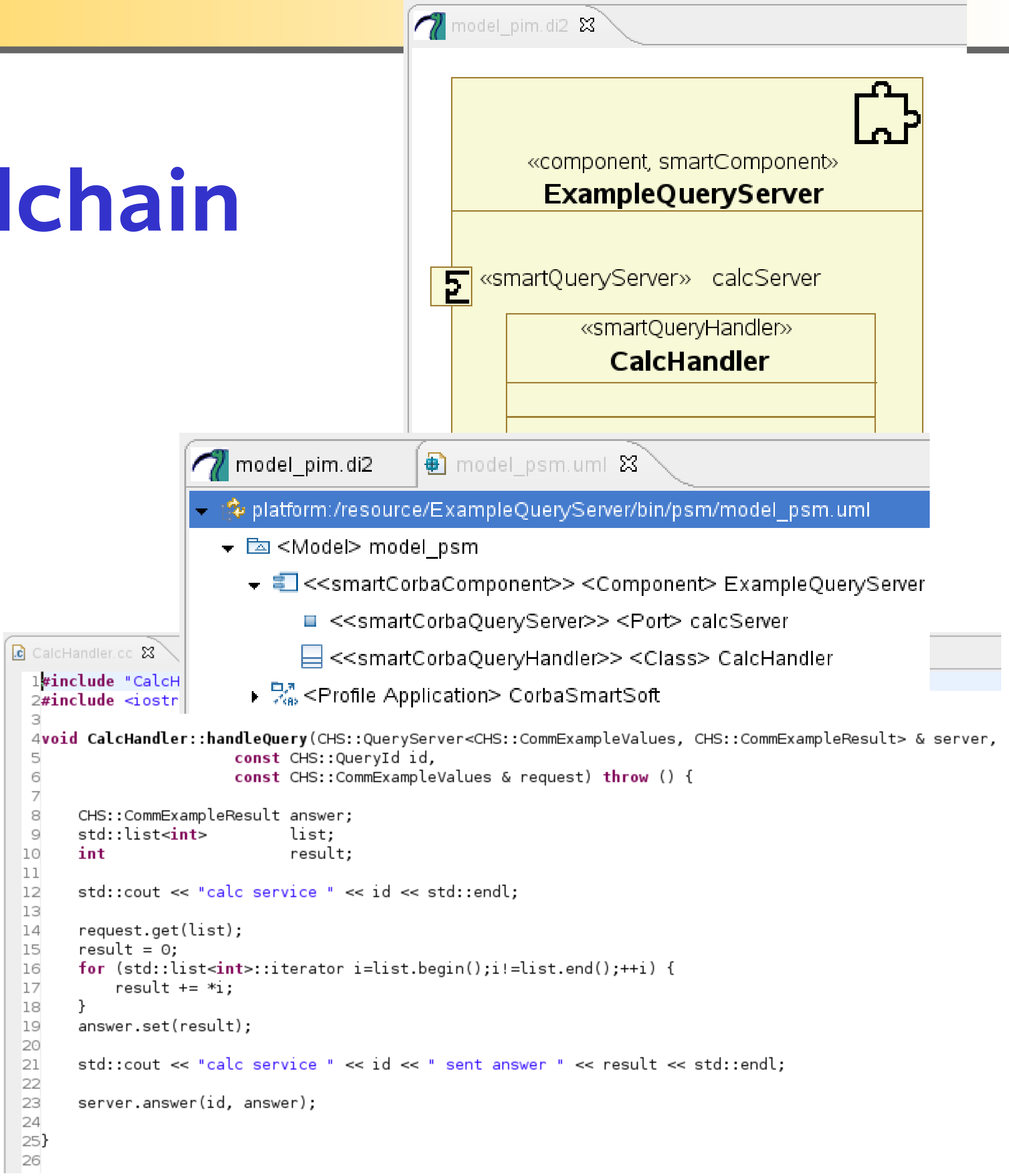
http://smart-robotics.sourceforge.net/
http://www.zafh-servicerobotik.de/ULM/index.php

Hochschule Ulm

SmartSoft MDSD Toolchain

Leuven 2009-07-09

- Presentation Toolchain
- Live Demo
- Behavior Modeling



platform:/resource/ExampleQueryServer/bin/psm/model_psm.uml
<Model> model_psm
<<smartCorbaComponent>> <Component> ExampleQueryServer
<<smartCorbaQueryServer>> <Port> calcServer
<<smartCorbaQueryHandler>> <Class> CalcHandler
<Profile Application> CorbaSmartSoft

```
#include "CalcH
#include <iostr

void CalcHandler::handleQuery(CHS::QueryServer<CHS::CommExampleValues, CHS::CommExampleResult> & server,
                              const CHS::QueryId id,
                              const CHS::CommExampleValues & request) throw () {

    CHS::CommExampleResult answer;
    std::list<int>         list;
    int                    result;

    std::cout << "calc service " << id << std::endl;

    request.get(list);
    result = 0;
    for (std::list<int>::iterator i=list.begin();i!=list.end();++i) {
        result += *i;
    }
    answer.set(result);

    std::cout << "calc service " << id << " sent answer " << result << std::endl;

    server.answer(id, answer);

}
```

Model Driven Software Development
Idea and Approach

SmartSoft can be seen as:

- the idea
  - how robotics systems should be composed out of components
  - how the components hull looks like
  - how the components interact with each other

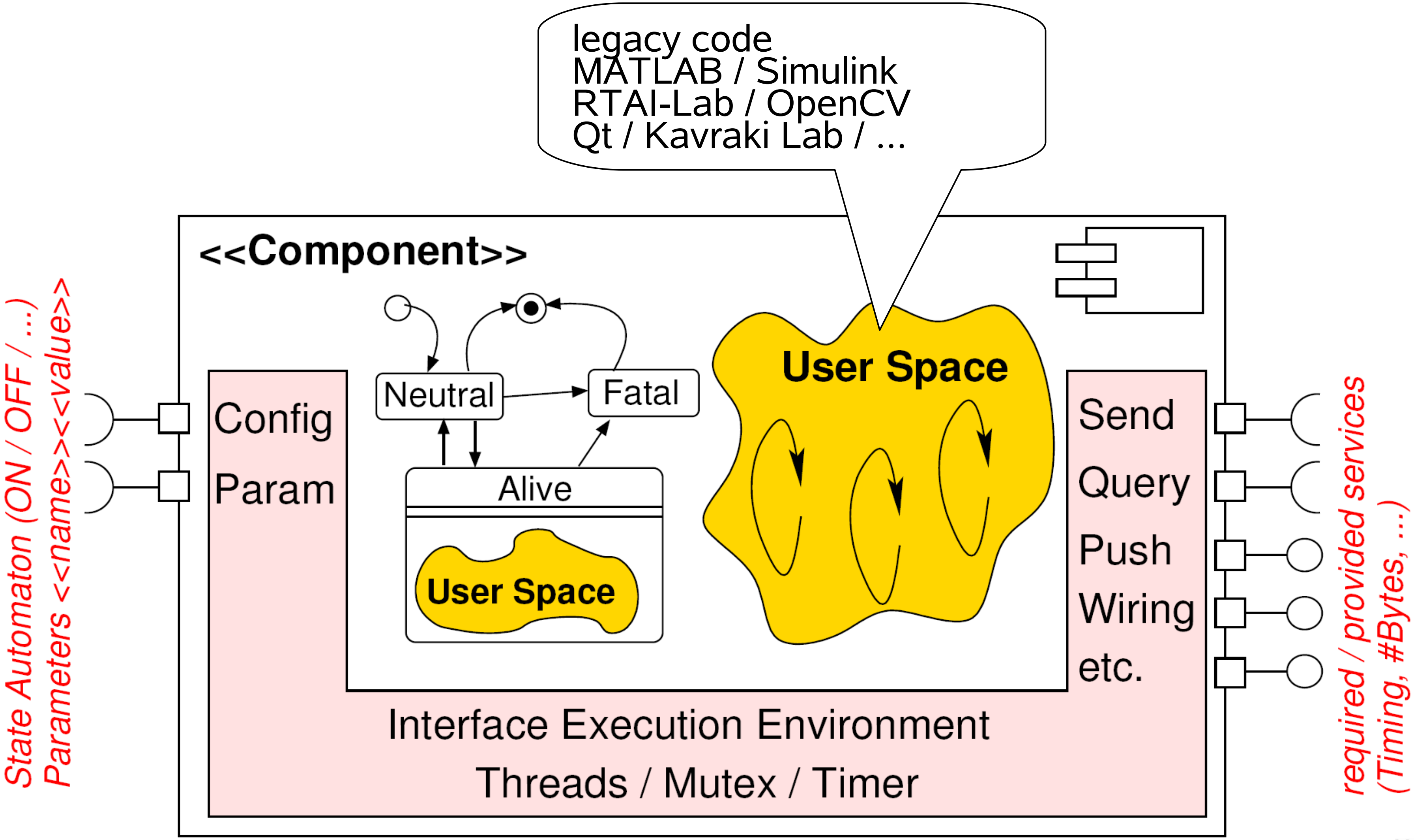
- the concrete implementations based on
  - CORBA => CorbaSmartSoft
  - ACE only
  - ...

These patterns are sufficient since they offer request/response interaction as well as asynchronous notifications and push services.

The SmartSoft Interaction Patterns

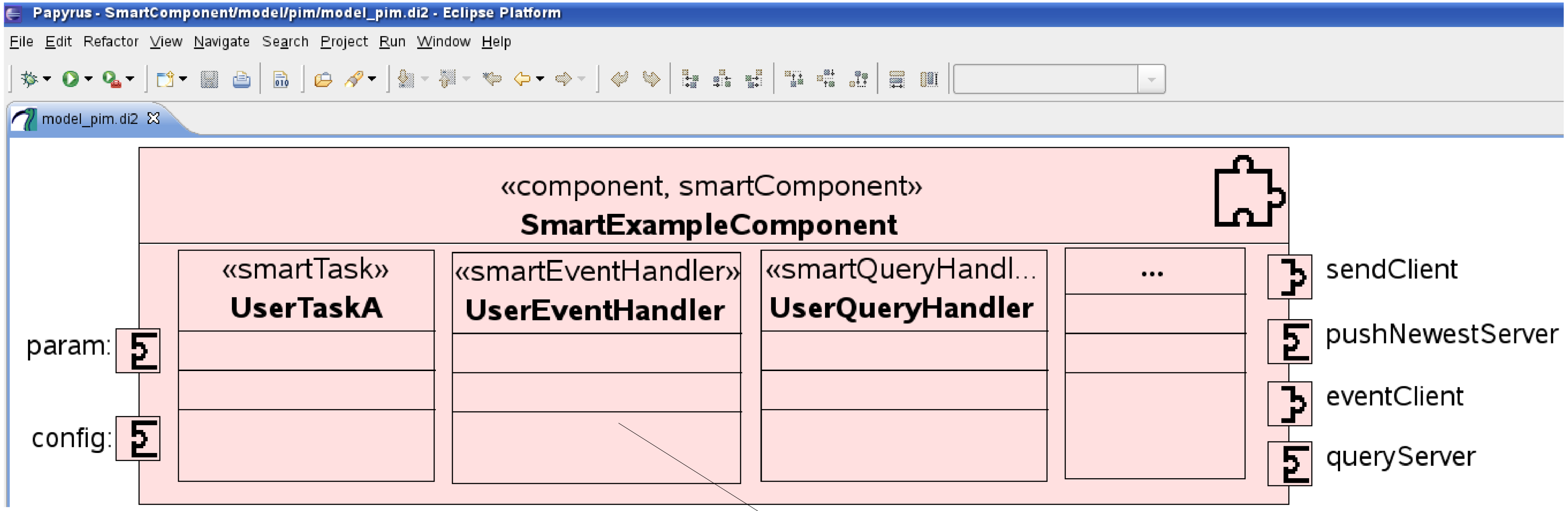
send	one-way communication
query	two-way request/response
push newest	1-to-n distribution
push timed	1-to-n distribution
event	asynchronous conditioned notification
wiring	dynamic component wiring

Model Driven Software Development
Idea and Approach



Model Driven Software Development Workflow Example (User View)

PIM



Papyrus - SmartComponent/model/pim/model_pim.di2 - Eclipse Platform

File Edit Refactor View Navigate Search Project Run Window Help
model_pim.di2

«component, smartComponent»
SmartExampleComponent

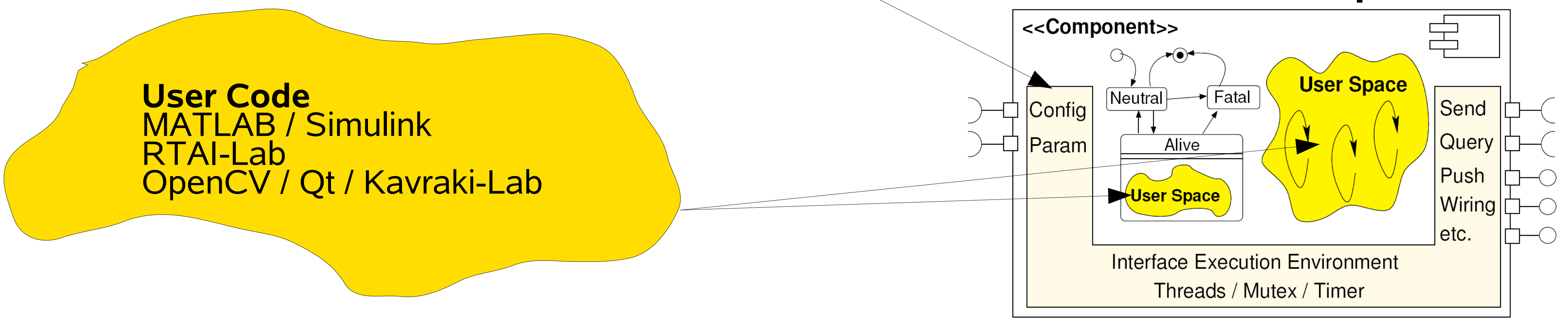
«smartTask»	«smartEventHandler»	«smartQueryHandl...	...
UserTaskA	**UserEventHandler**	**UserQueryHandler**	

param:
config:

sendClient
pushNewestServer
eventClient
queryServer

verification (e.g. QoS)+
transformation

executable component



User Code
MATLAB / Simulink
RTAI-Lab
OpenCV / Qt / Kavraki-Lab

Model Driven Software Development
The Workflow

PIM

SmartMARS – Metamodel

(**M**odeling and **A**nalysis of **R**obotics **S**ystems)

- UML2-Profile
- platform independent stereotypes
  - SmartComponent
  - SmartTask
  - SmartMutex
  - SmartQueryServer
  - SmartEventClient
  - ...

M2M
oAW
xTend

PSM

CorbaSmartSoft
CORBA based implementation of SmartSoft

AceSmartSoft
ACE based implementation of SmartSoft

Microsoft Robotic Studio
MSRS based implementation

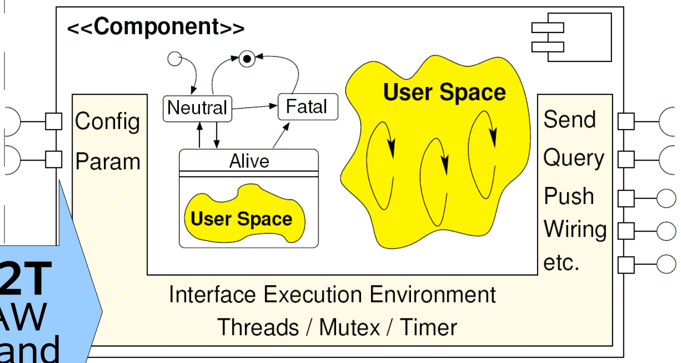
...
any other middleware

- UML2-Profile
- platform specific stereotypes

has to be created by a middleware expert

M2T
oAW
xPand

PSI



The User Space can contain arbitrary code and libraries

The User Space stays the same independent of the different platform specific models

Just the component hull will be created

Model Driven Software Development Workflow Example

PIM

SmartMARS – Metamodel

(**M**odeling and **A**nalysis of **R**obotics **S**ystems)

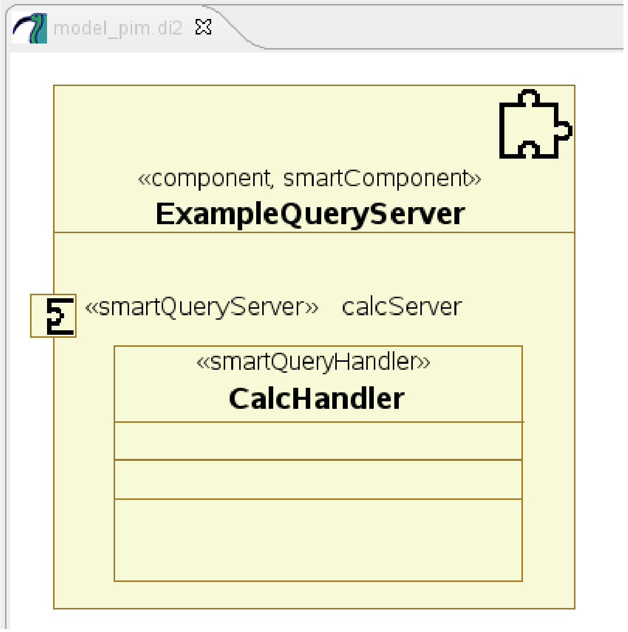
M2M
oAW
xTend

PSM

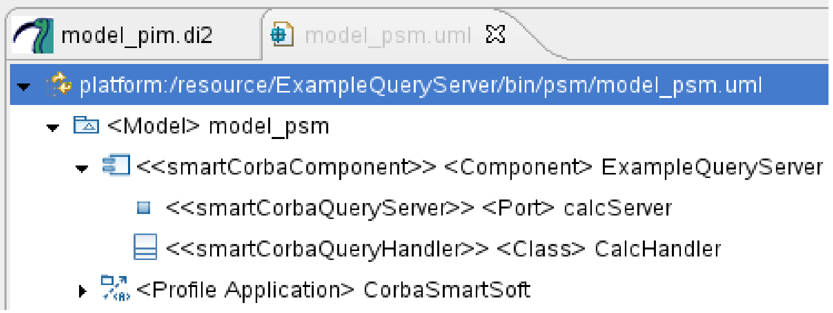
CorbaSmartSoft
CORBA based implementation of SmartSoft

M2T
oAW
xPand

PSI



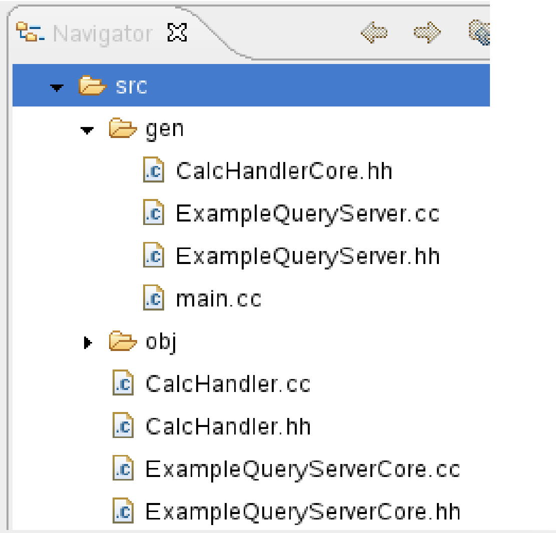
user has to create a PIM



model_pim.di2 | model_psm.uml

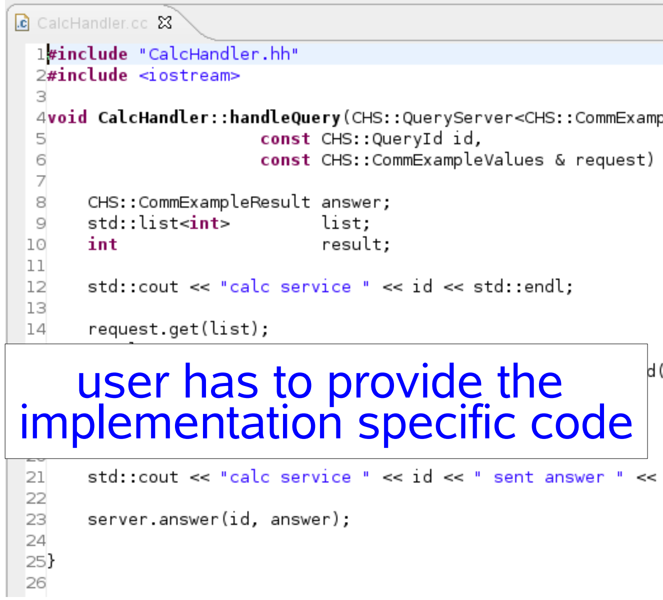
- platform:/resource/ExampleQueryServer/bin/psm/model_psm.uml
  - <Model> model_psm
    - <<smartCorbaComponent>> <Component> ExampleQueryServer
      - <<smartCorbaQueryServer>> <Port> calcServer
      - <<smartCorbaQueryHandler>> <Class> CalcHandler
    - <Profile Application> CorbaSmartSoft

no need to change anything in the PSM



Navigator

- src
  - gen
    - CalcHandlerCore.hh
    - ExampleQueryServer.cc
    - ExampleQueryServer.hh
    - main.cc
  - obj
  - CalcHandler.cc
  - CalcHandler.hh
  - ExampleQueryServerCore.cc
  - ExampleQueryServerCore.hh



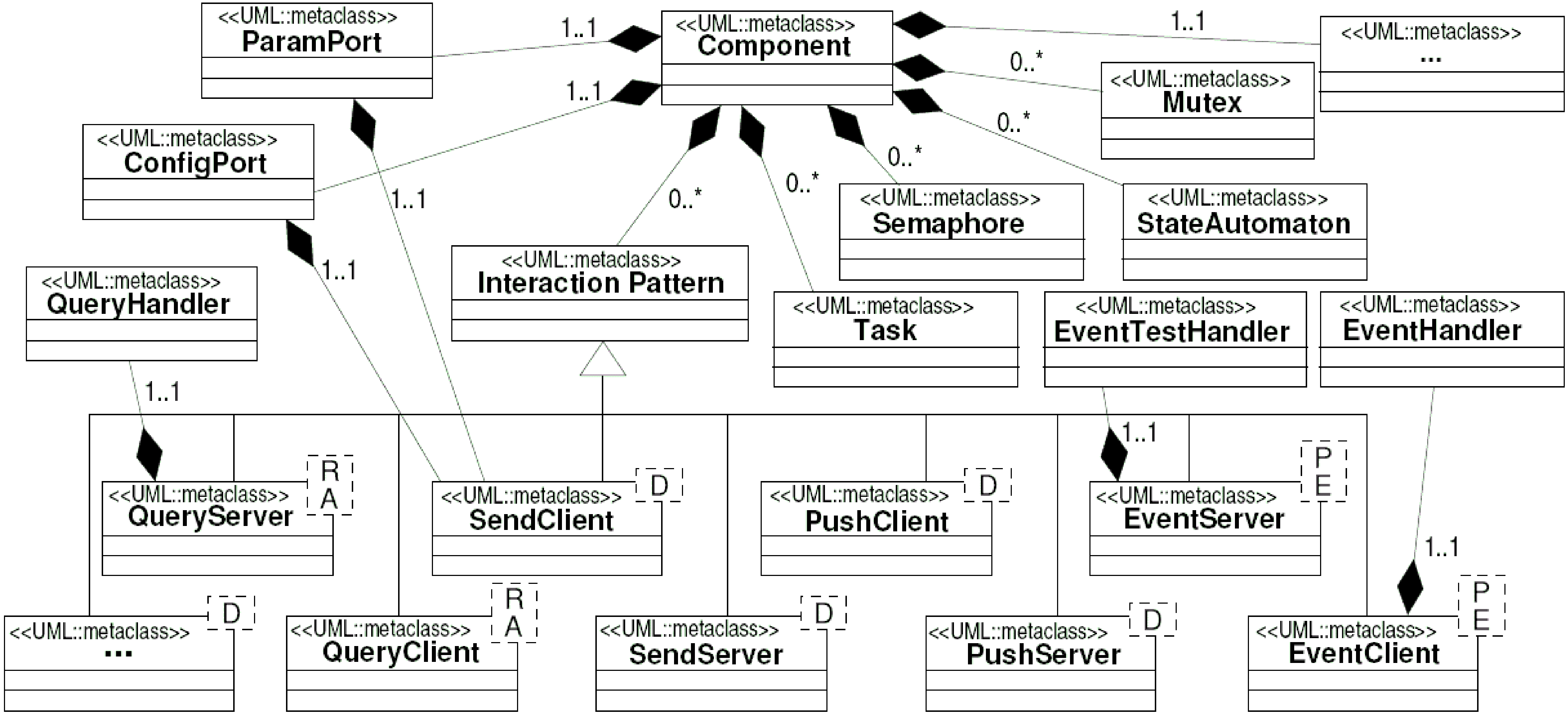
CalcHandler.cc

```
#include "CalcHandler.hh"
#include <iostream>

void CalcHandler::handleQuery(CHS::QueryServer<CHS::CommExamp
                              const CHS::QueryId id,
                              const CHS::CommExampleValues & request)

    CHS::CommExampleResult answer;
    std::list<int>         list;
    int                    result;

    std::cout << "calc service " << id << std::endl;

    request.get(list);
```

```
    std::cout << "calc service " << id << " sent answer " <<

    server.answer(id, answer);

}

```

user has to provide the implementation specific code

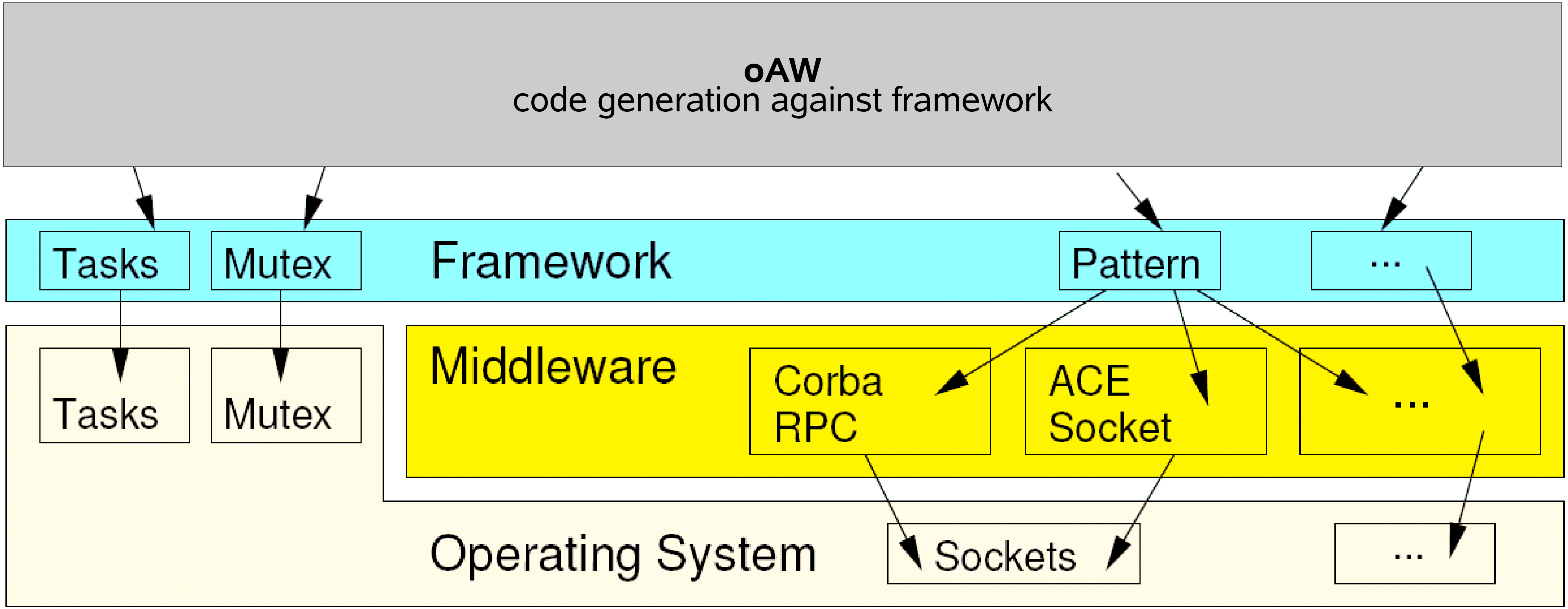
Model Driven Software Development
Metamodel



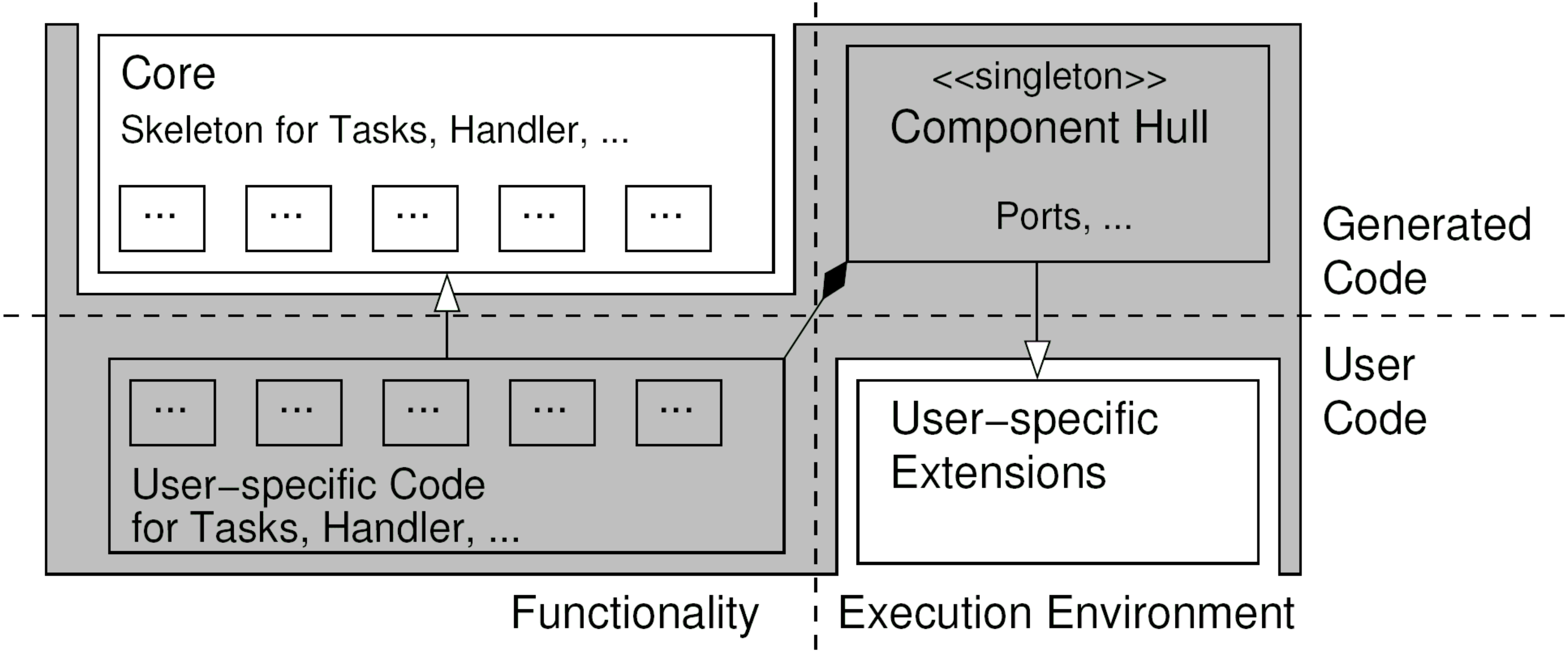
<<UML::metaclass>> ParamPort
<<UML::metaclass>> Component
<<UML::metaclass>> ...
<<UML::metaclass>> Mutex
<<UML::metaclass>> ConfigPort
<<UML::metaclass>> Semaphore
<<UML::metaclass>> StateAutomaton
<<UML::metaclass>> Interaction Pattern
<<UML::metaclass>> QueryHandler
<<UML::metaclass>> Task
<<UML::metaclass>> EventTestHandler
<<UML::metaclass>> EventHandler
<<UML::metaclass>> QueryServer
<<UML::metaclass>> SendClient
<<UML::metaclass>> PushClient
<<UML::metaclass>> EventServer
<<UML::metaclass>> ...
<<UML::metaclass>> QueryClient
<<UML::metaclass>> SendServer
<<UML::metaclass>> PushServer
<<UML::metaclass>> EventClient
1..1
1..1
1..1
0..*
0..*
1..1
0..*
0..*
0..*
1..1
1..1
1..1
1..1
R A
D
D
P E
D
R A
D
D
P E

Hochschule Ulm

Model Driven Software Development Framework



Model Driven Software Development
Structure source code



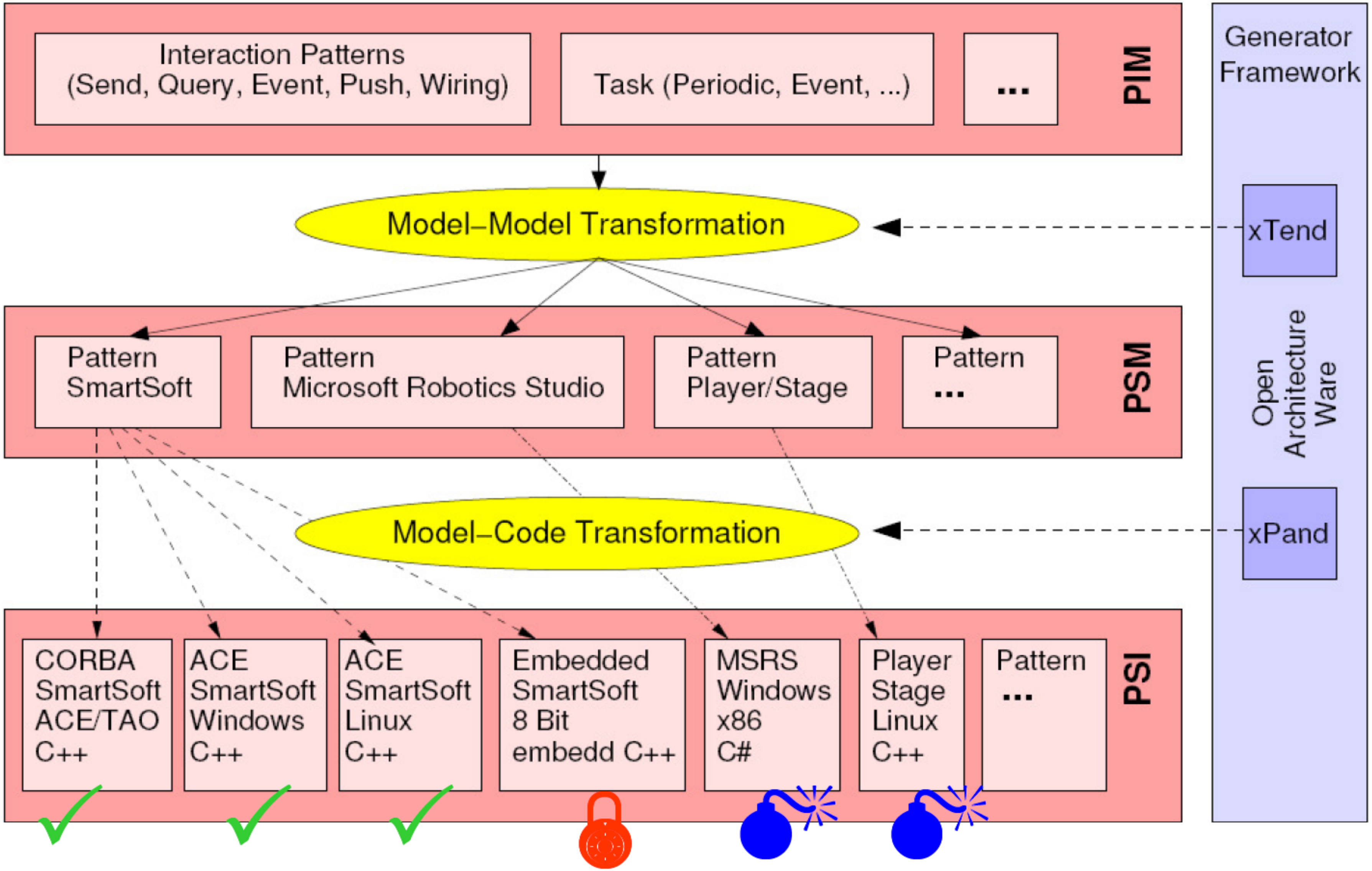
Hochschule Ulm

Demo

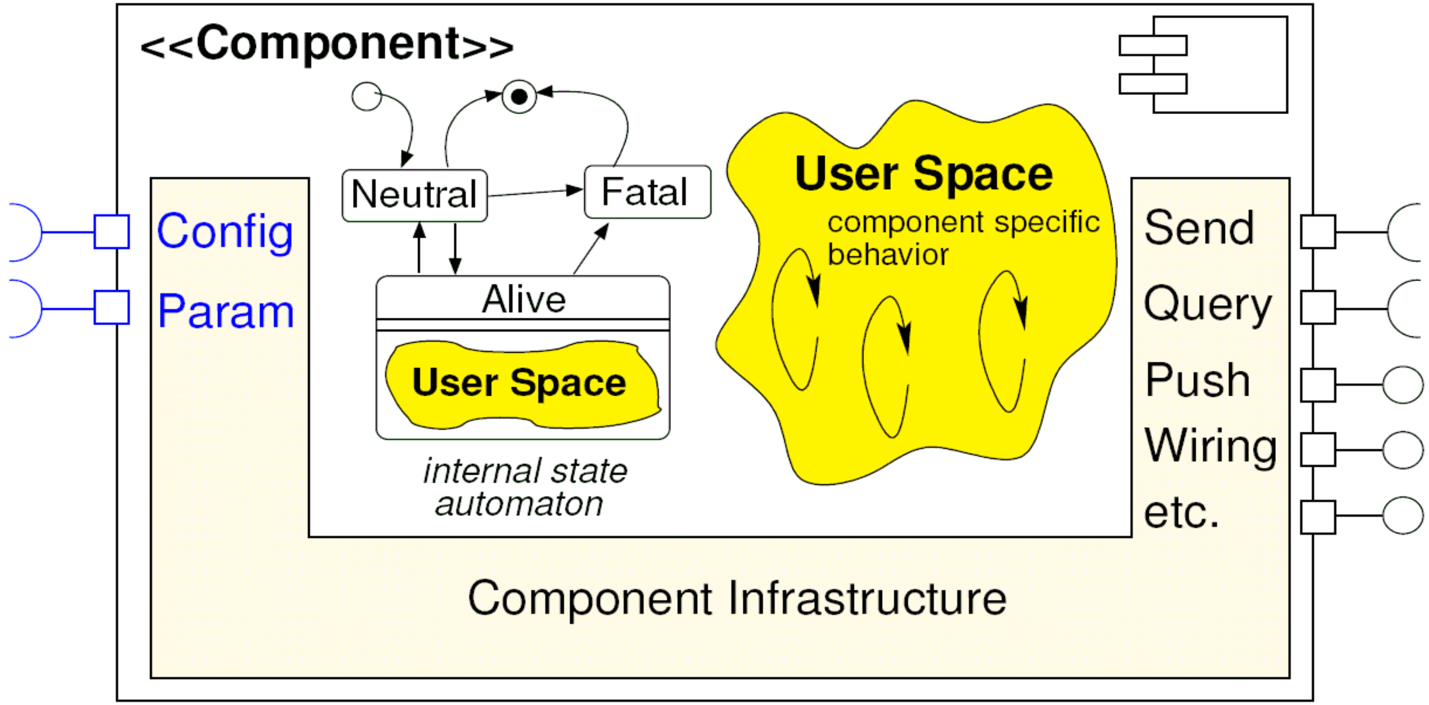
SmartSoft MDSD

Toolchain

Model Driven Software Development
Current Status


Interaction Patterns
(Send, Query, Event, Push, Wiring)
Task (Periodic, Event, ...)
...
PIM
Generator Framework
Model–Model Transformation
xTend
Pattern SmartSoft
Pattern Microsoft Robotics Studio
Pattern Player/Stage
Pattern ...
PSM
Open Architecture Ware
Model–Code Transformation
xPand
CORBA SmartSoft ACE/TAO C++
ACE SmartSoft Windows C++
ACE SmartSoft Linux C++
Embedded SmartSoft 8 Bit embedd C++
MSRS Windows x86 C#
Player Stage Linux C++
Pattern ...
PSI
✓
✓
✓

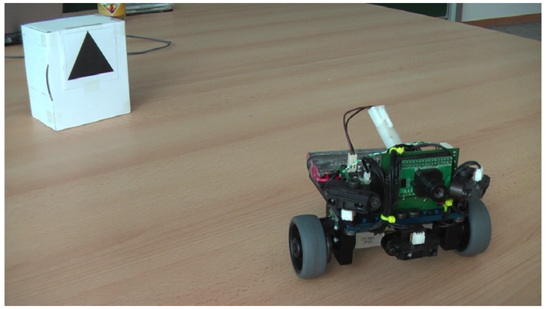
Behavior Modeling
Interfacing to Behavior Component



- **config:** set the component into an appropriate state
  e.g. naviagtion has the states: **neutral; moverobot**

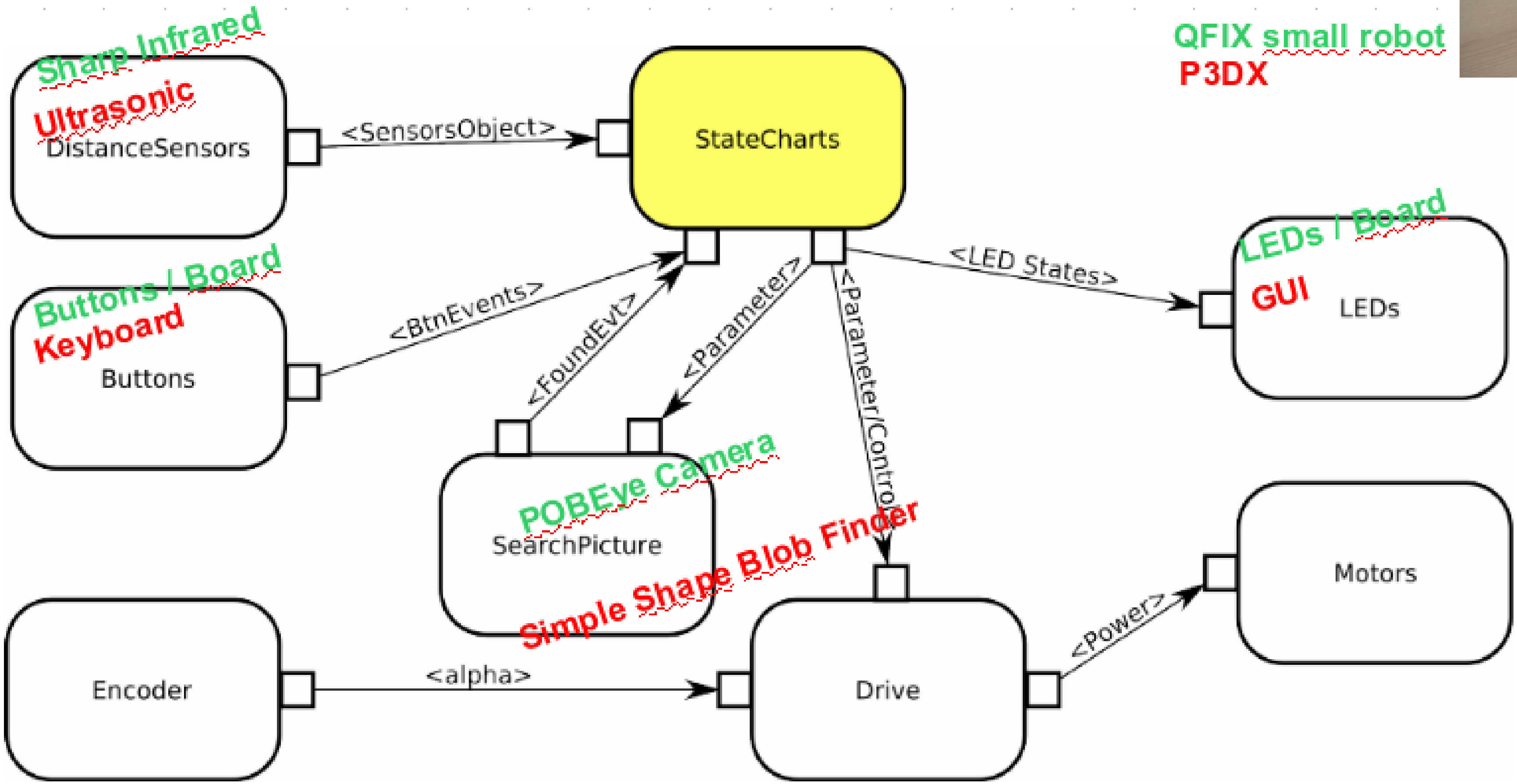
- **param:** send some parameters to the component
  - navigation: **(TRANSVEL(0)(500))**
  - pathPlanner: **(SETDESTINATIONCIRCLE(xPos)(yPos)(dist))**

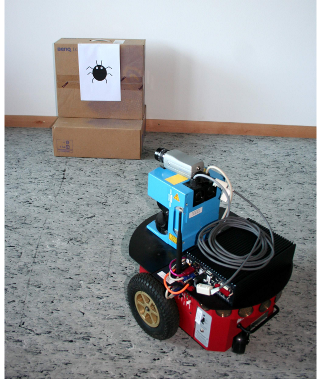
Behavior Modeling Practical Example - Statecharts



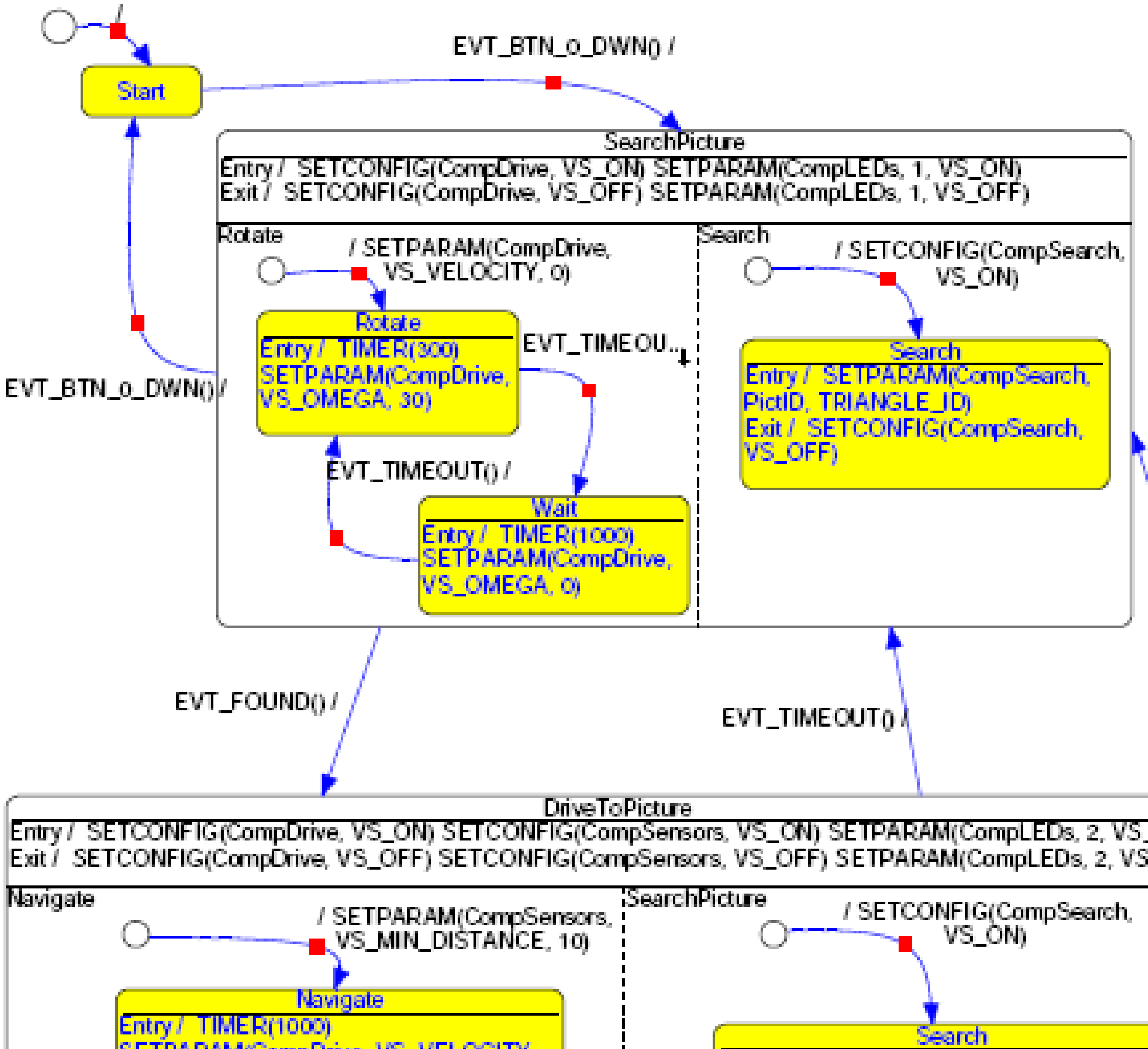
QFIX small robot

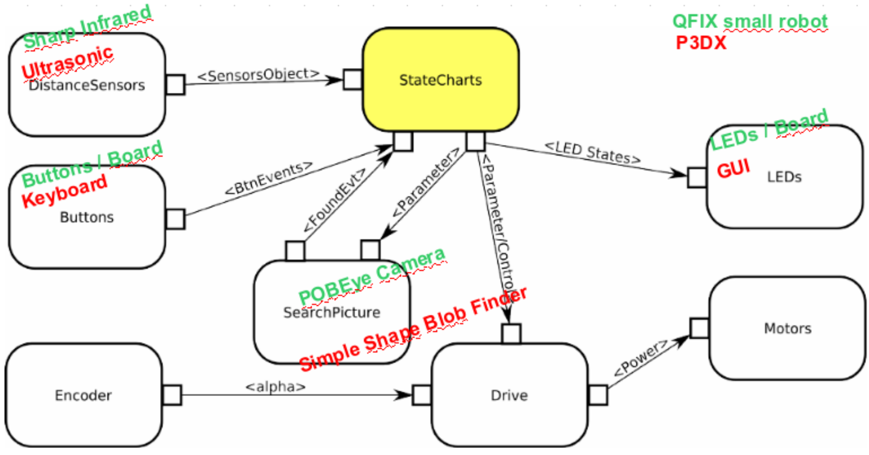
P3DX





Behavior Modeling
Practical Example - Statecharts


/
Start
EVT_BTN_o_DWN() /
SearchPicture
Entry / SETCONFIG(CompDrive, VS_ON) SETPARAM(CompLEDs, 1, VS_ON)
Exit / SETCONFIG(CompDrive, VS_OFF) SETPARAM(CompLEDs, 1, VS_OFF)
Rotate
/ SETPARAM(CompDrive, VS_VELOCITY, 0)
Rotate
Entry / TIMER(300) SETPARAM(CompDrive, VS_OMEGA, 30)
EVT_TIMEOU...
EVT_TIMEOUT() /
Wait
Entry / TIMER(1000) SETPARAM(CompDrive, VS_OMEGA, 0)
Search
/ SETCONFIG(CompSearch, VS_ON)
Search
Entry / SETCONFIG(CompSearch, PictID, TRIANGLE_ID)
Exit / SETCONFIG(CompSearch, VS_OFF)
EVT_BTN_o_DWN() /
EVT_FOUND() /
EVT_TIMEOUT() /
DriveToPicture
Entry / SETCONFIG(CompDrive, VS_ON) SETCONFIG(CompSensors, VS_ON) SETPARAM(CompLEDs, 2, VS_ON)
Exit / SETCONFIG(CompDrive, VS_OFF) SETCONFIG(CompSensors, VS_OFF) SETPARAM(CompLEDs, 2, VS_OFF)
Navigate
/ SETPARAM(CompSensors, VS_MIN_DISTANCE, 10)
Navigate
Entry / TIMER(1000)
SETPARAM(CompDrive, VS_VELOCITY
SearchPicture
/ SETCONFIG(CompSearch, VS_ON)
Search

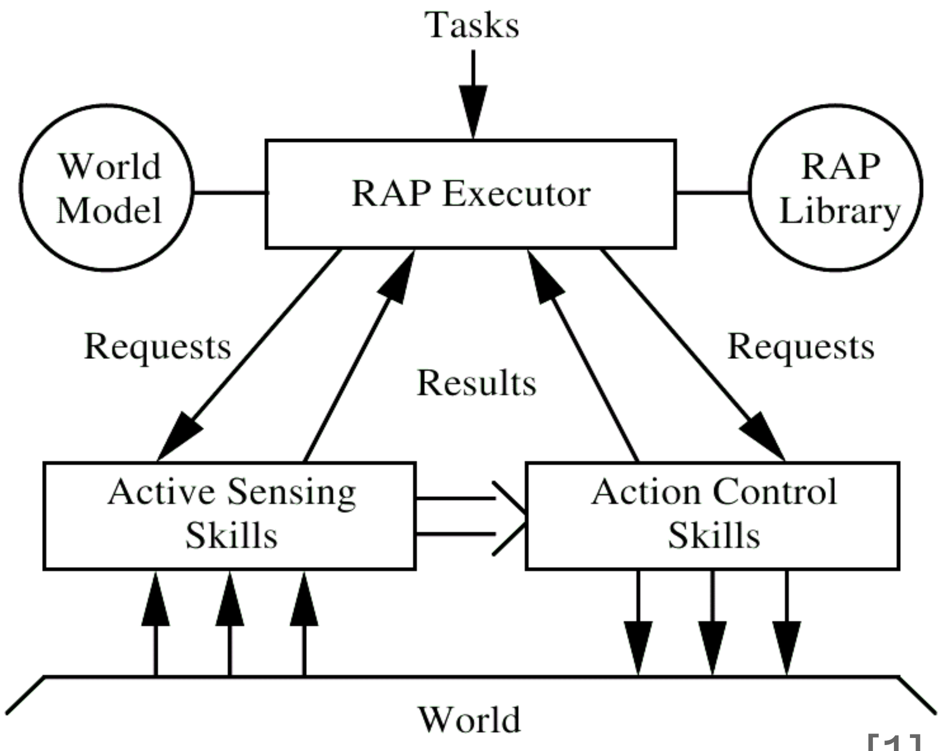


Behavior Modeling
The RAP System

- A RAP is an entity including the methods to achieve a goal

- The RAP system expands sketchy plans into detailed steps during execution dependent on the current state of the world

- A RAP can contain several other RAPs which are then organized in TASK-NETs

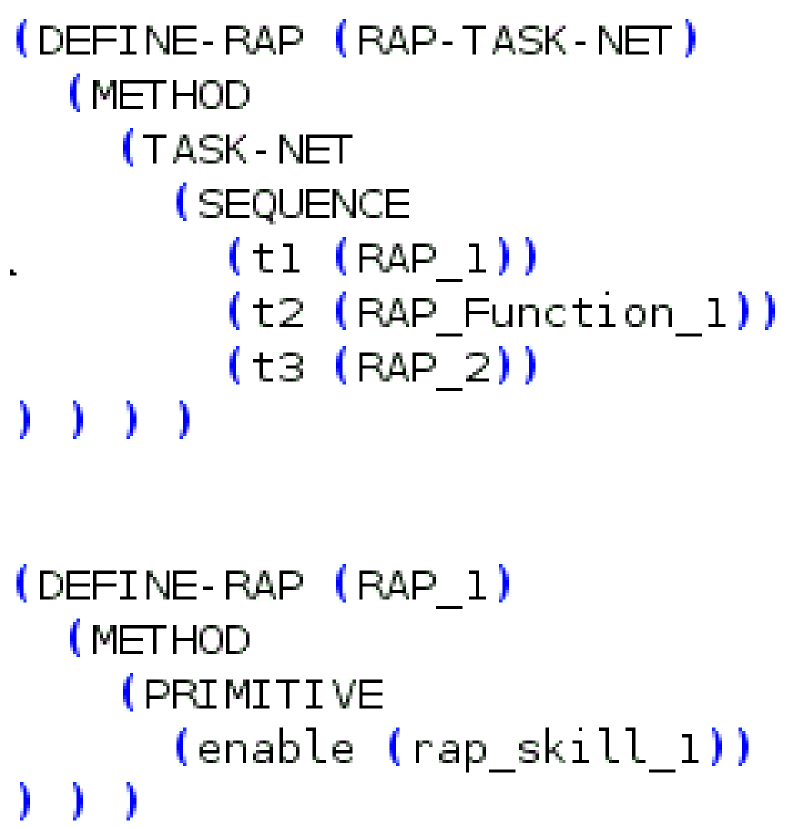


[1]

- Primitive RAPs (skills) build the interface to the robot
  - they can not be used directly in TASK-NETs

Behavior Modeling
The RAP System

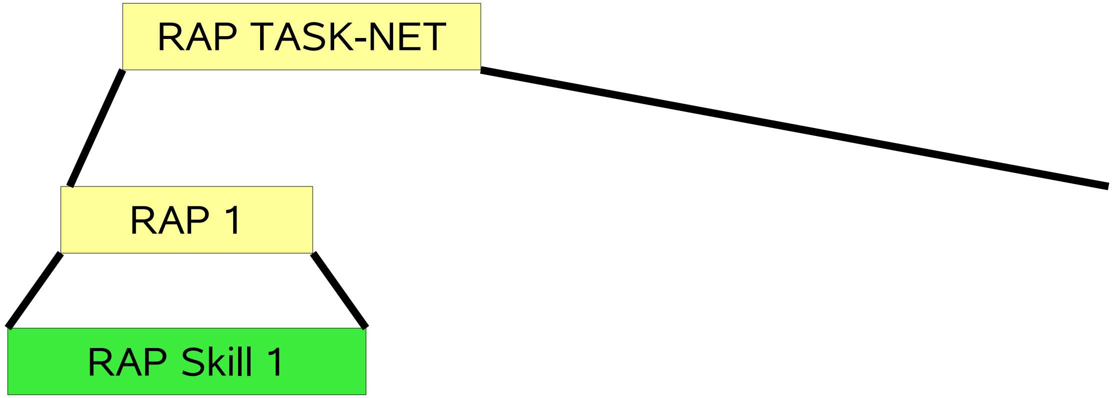
RAP TASK-NET

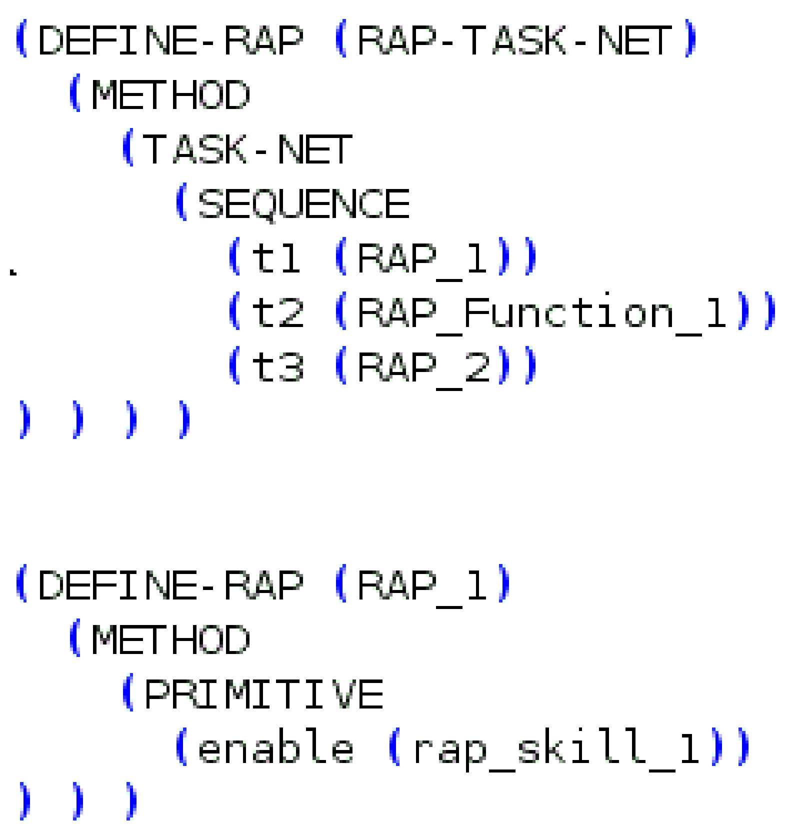


```
(DEFINE-RAP (RAP-TASK-NET)
  (METHOD
    (TASK-NET
      (SEQUENCE
        (t1 (RAP_1))
        (t2 (RAP_Function_1))
        (t3 (RAP_2))
) ) ) )
```

```
(DEFINE-RAP (RAP_1)
  (METHOD
    (PRIMITIVE
      (enable (rap_skill_1))
) ) )
```

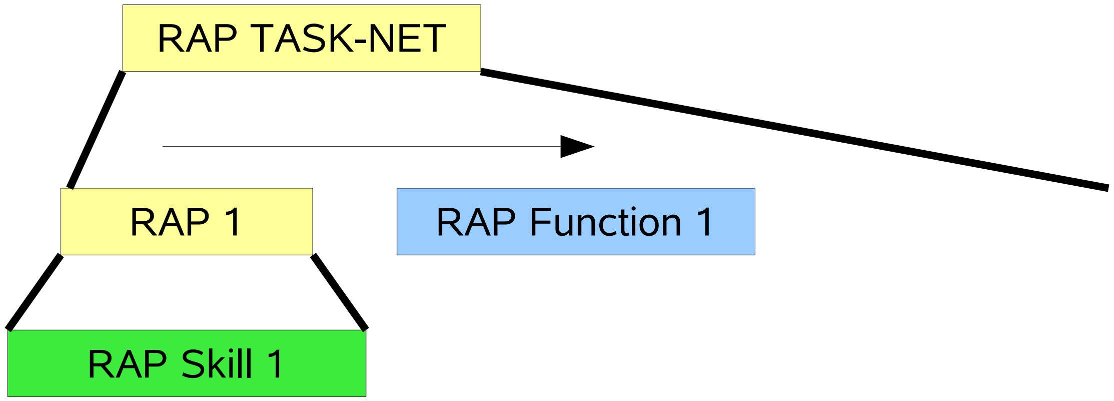
Behavior Modeling
The RAP System

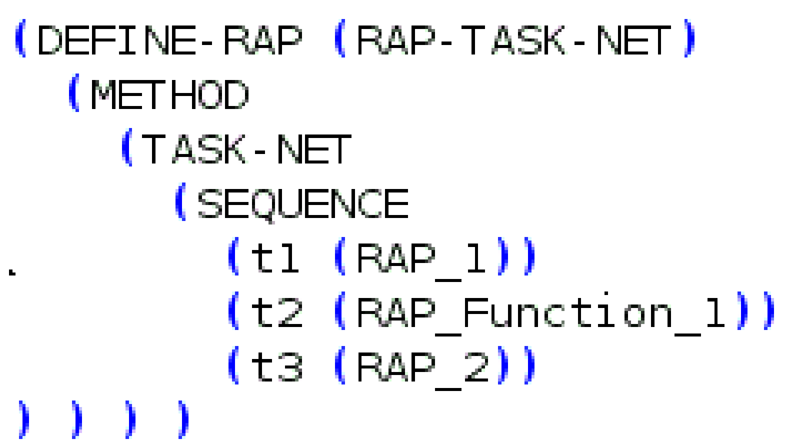




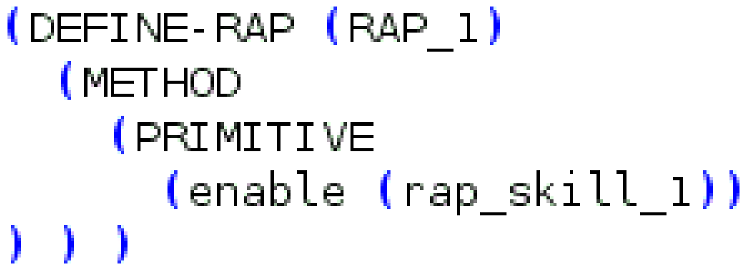
```
(DEFINE-RAP (RAP-TASK-NET)
  (METHOD
    (TASK-NET
      (SEQUENCE
        (t1 (RAP_1))
        (t2 (RAP_Function_1))
        (t3 (RAP_2))
) ) ) )


(DEFINE-RAP (RAP_1)
  (METHOD
    (PRIMITIVE
      (enable (rap_skill_1))
) ) )
```

Behavior Modeling
The RAP System



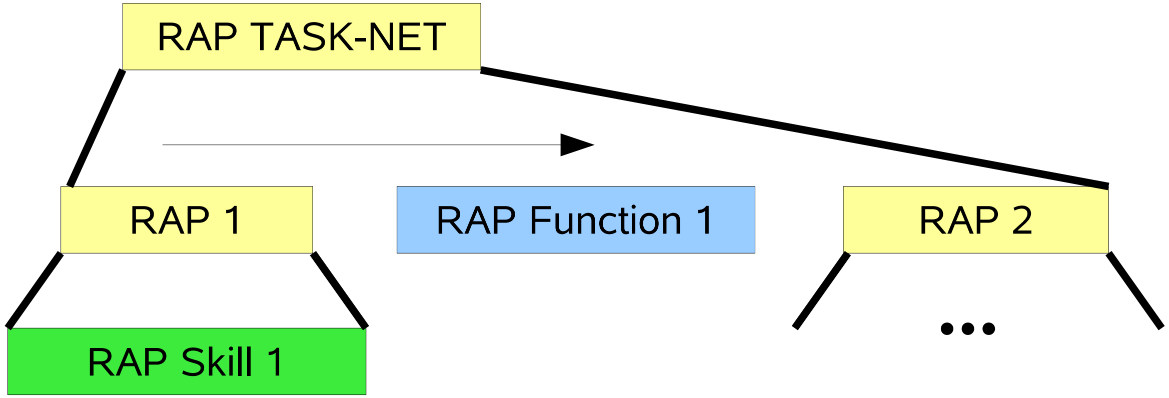


```
(DEFINE-RAP (RAP-TASK-NET)
  (METHOD
    (TASK-NET
      (SEQUENCE
        (t1 (RAP_1))
        (t2 (RAP_Function_1))
        (t3 (RAP_2))
) ) ) )
```



```
(DEFINE-RAP (RAP_1)
  (METHOD
    (PRIMITIVE
      (enable (rap_skill_1))
) ) )
```

Behavior Modeling
The RAP System



RAP TASK-NET

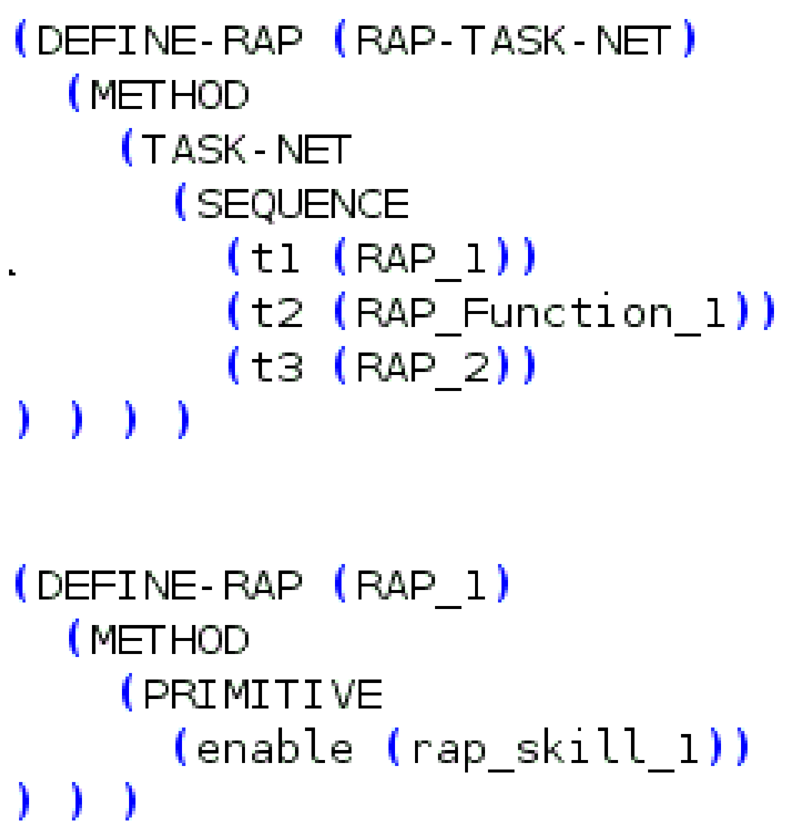
RAP 1

RAP Function 1

RAP 2

RAP Skill 1

...

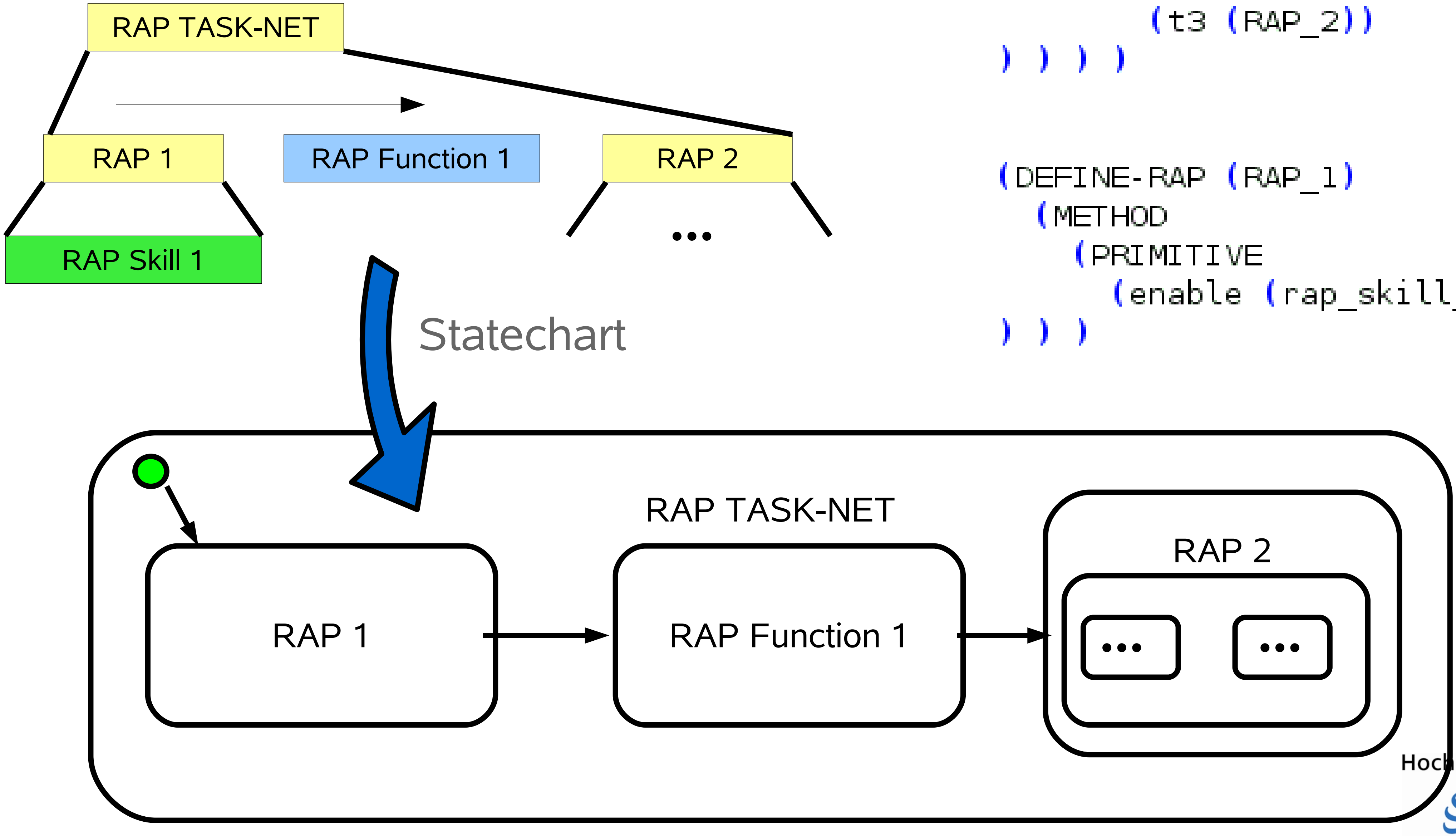


```
(DEFINE-RAP (RAP-TASK-NET)
  (METHOD
    (TASK-NET
      (SEQUENCE
        (t1 (RAP_1))
        (t2 (RAP_Function_1))
        (t3 (RAP_2))
) ) ) )


(DEFINE-RAP (RAP_1)
  (METHOD
    (PRIMITIVE
      (enable (rap_skill_1))
) ) )
```

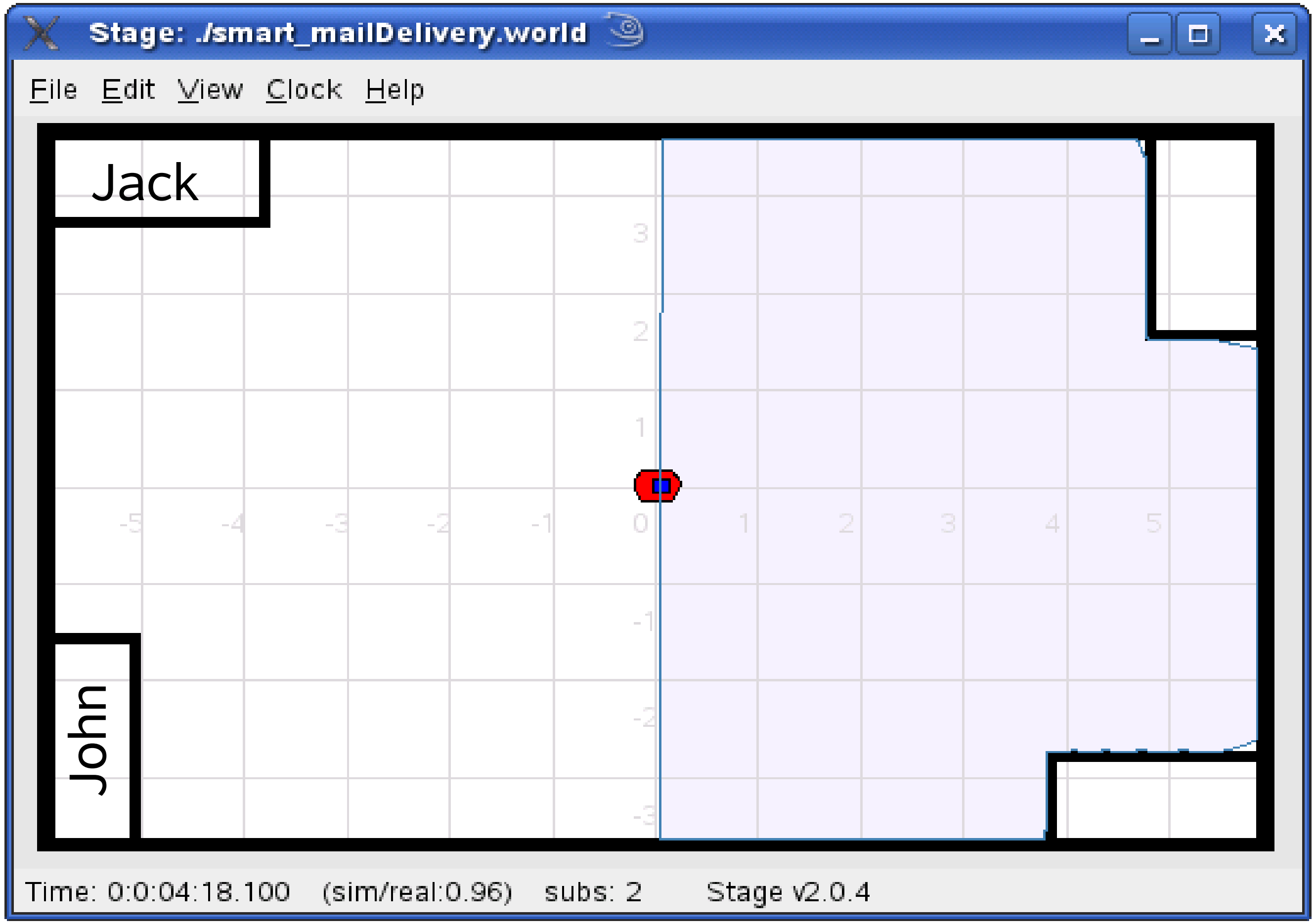
Behavior Modeling
The RAP System

```
(DEFINE-RAP (RAP-TASK-NET)
  (METHOD
    (TASK-NET
      (SEQUENCE
        (t1 (RAP_1))
        (t2 (RAP_Function_1))
        (t3 (RAP_2))
) ) ) )
```

```
(DEFINE-RAP (RAP_1)
  (METHOD
    (PRIMITIVE
      (enable (rap_skill_1))
) ) )
```



Behavior Modeling
Example: Mail Delivery

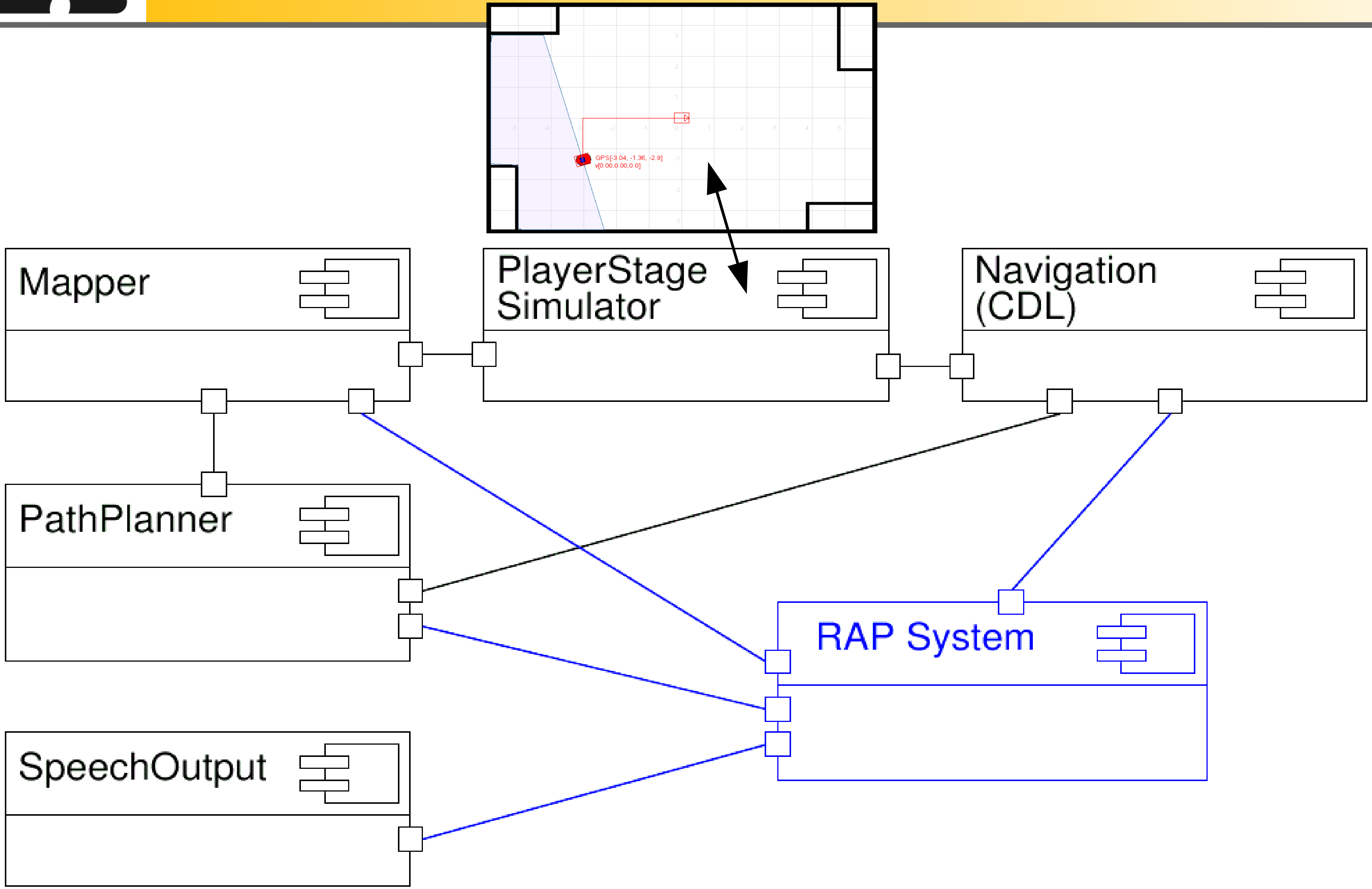


Time: 0:0:04:18.100 (sim/real:0.96) subs: 2 Stage v2.0.4

Jane

Mike

Behavior Modeling
Example Components



Mapper

PlayerStage Simulator

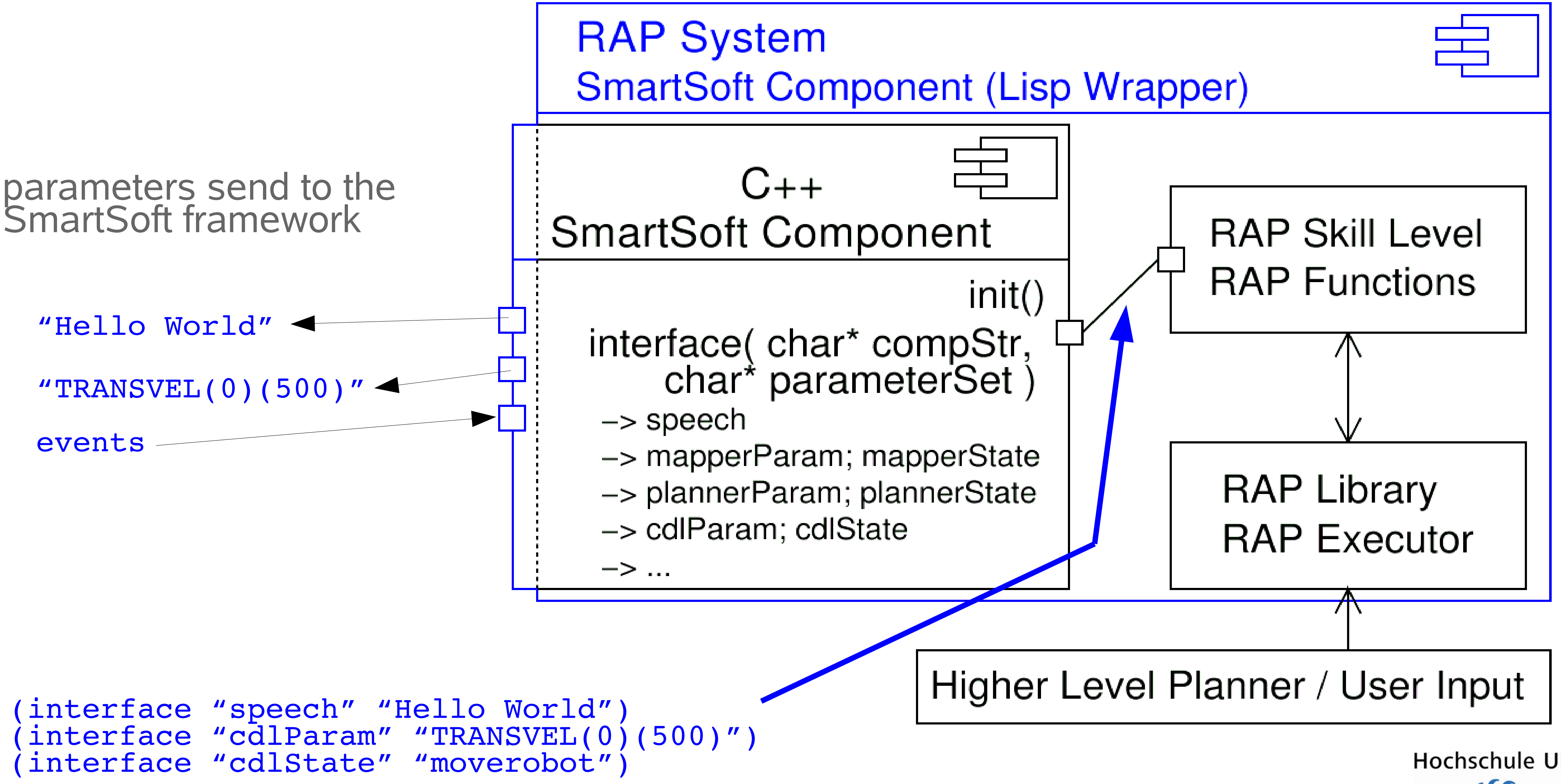
Navigation (CDL)

PathPlanner

SpeechOutput

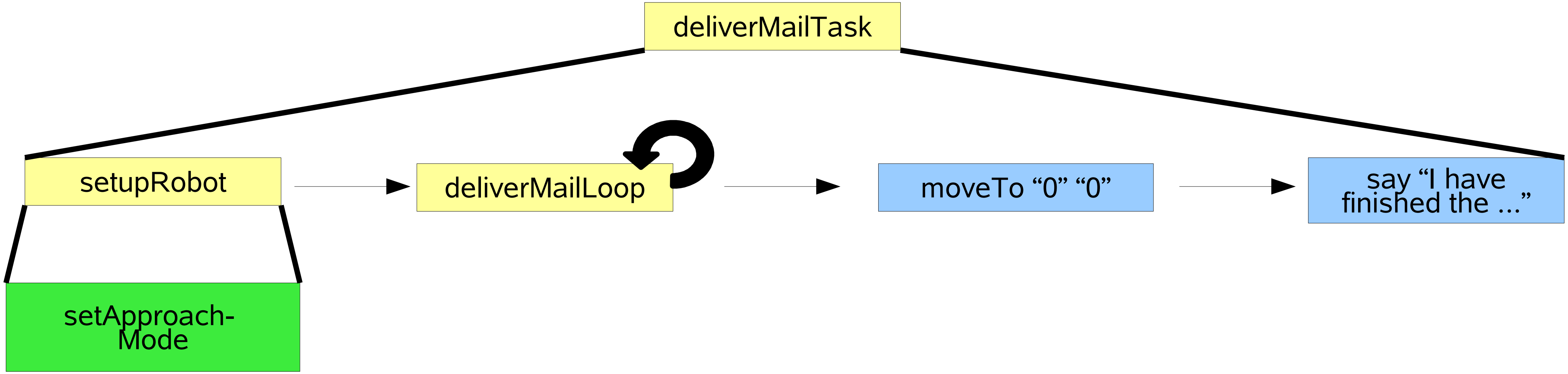
RAP System

Behavior Modeling
Interfacing SmartSoft ↔ RAP



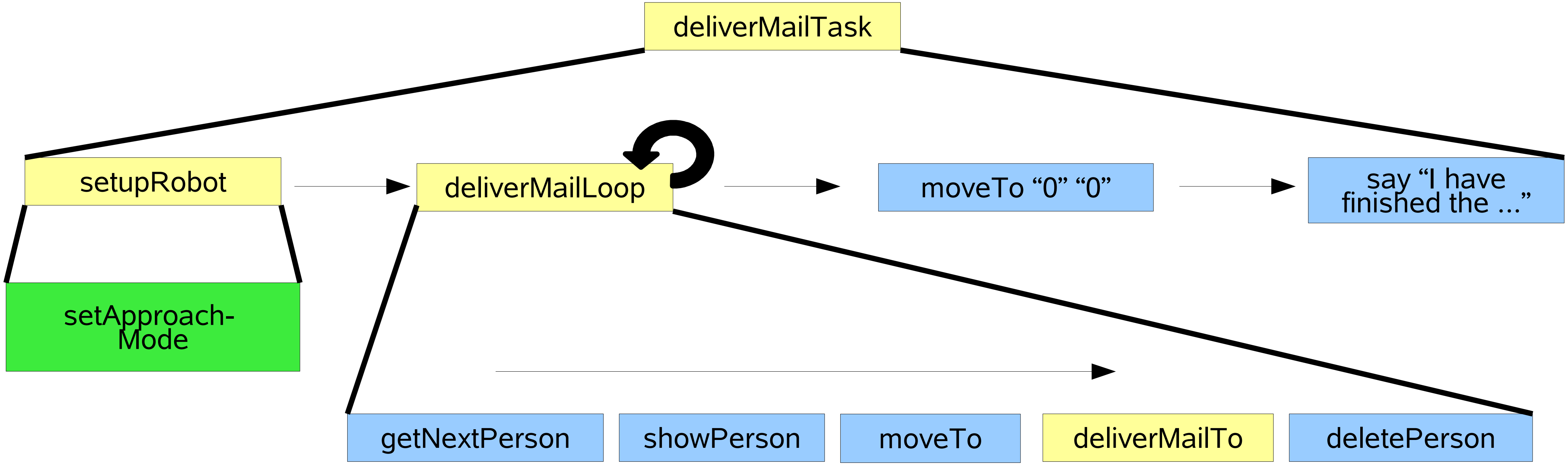
```
(interface “speech” “Hello World”)
(interface “cdlParam” “TRANSVEL(0)(500)”)
(interface “cdlState” “moverobot”)
```

Behavior Modeling Example: Mail Delivery



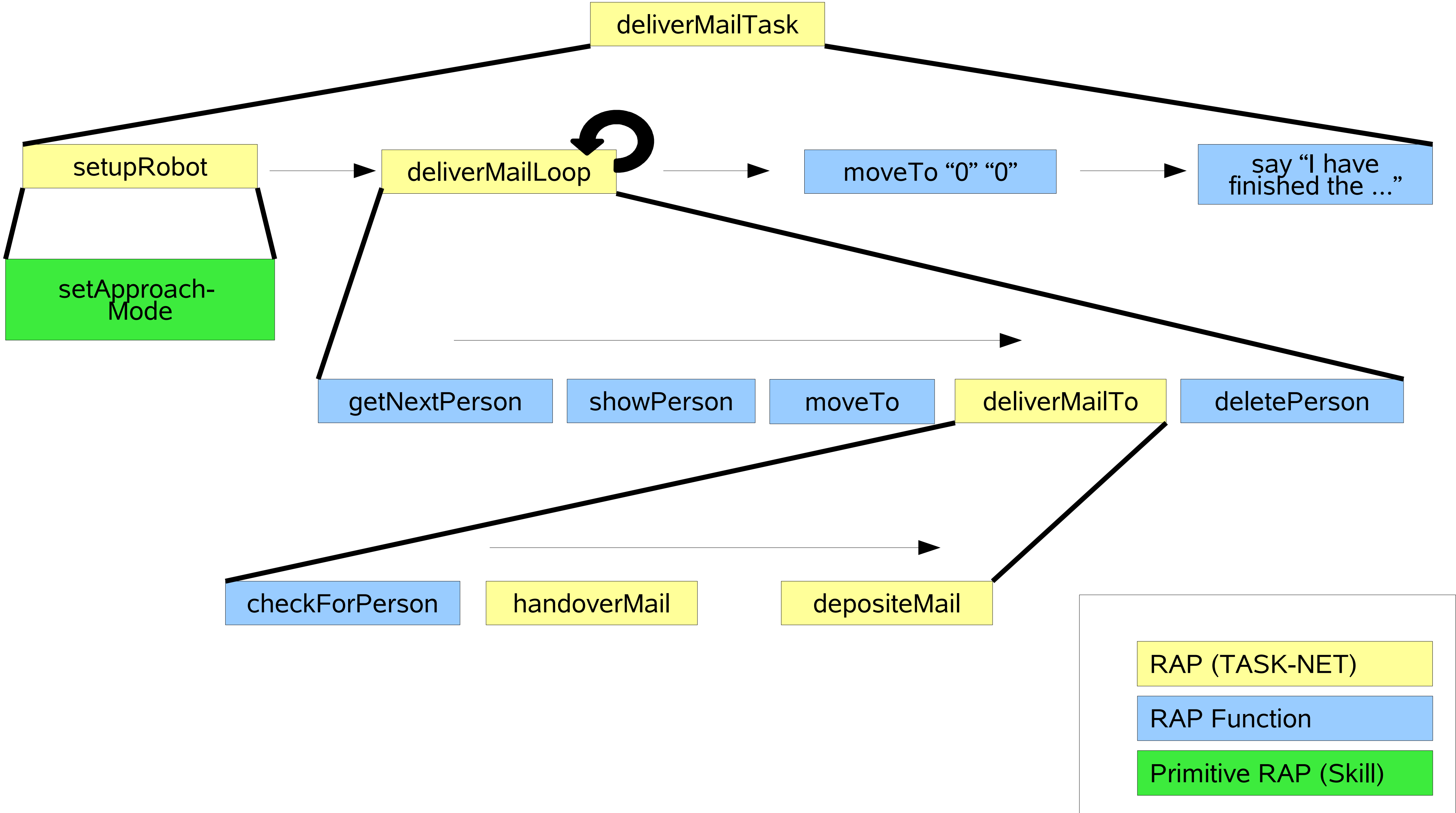
- 🟨 RAP (TASK-NET)
- 🟦 RAP Function
- 🟩 Primitive RAP (Skill)

Behavior Modeling
Example: Mail Delivery



🟨	RAP (TASK-NET)
🟦	RAP Function
🟩	Primitive RAP (Skill)

Behavior Modeling Example: Mail Delivery



deliverMailTask

setupRobot

setApproach-Mode

deliverMailLoop

moveTo “0” “0”

say “I have finished the …”

getNextPerson

showPerson

moveTo

deliverMailTo

deletePerson

checkForPerson

handoverMail

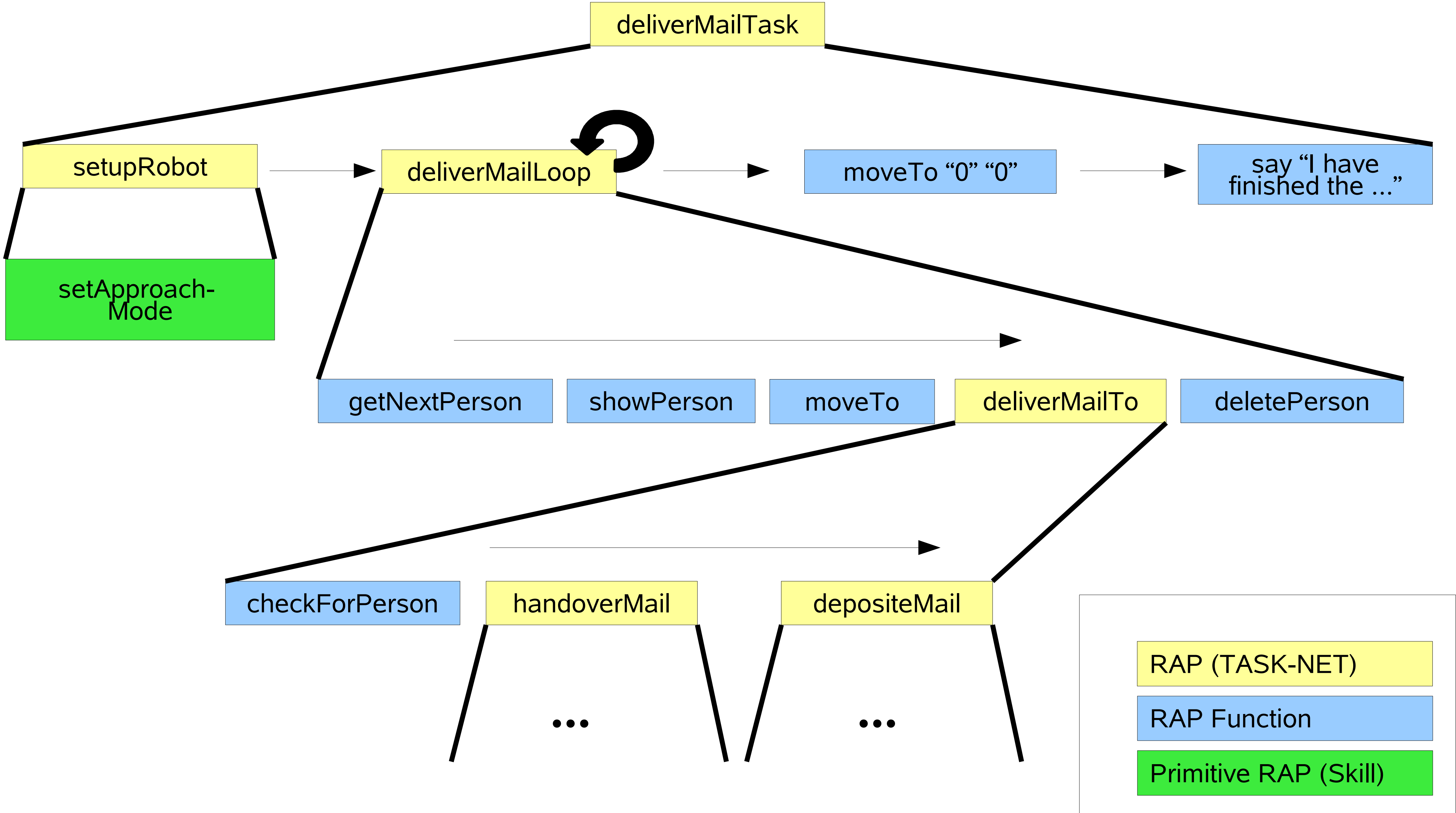
depositeMail

🟨 RAP (TASK-NET)

🟦 RAP Function

🟩 Primitive RAP (Skill)

Behavior Modeling
Example: Mail Delivery



deliverMailTask

setupRobot

setApproach-Mode

deliverMailLoop

moveTo “0” “0”

say “I have finished the …”

getNextPerson

showPerson

moveTo

deliverMailTo

deletePerson

checkForPerson

handoverMail

depositeMail

…

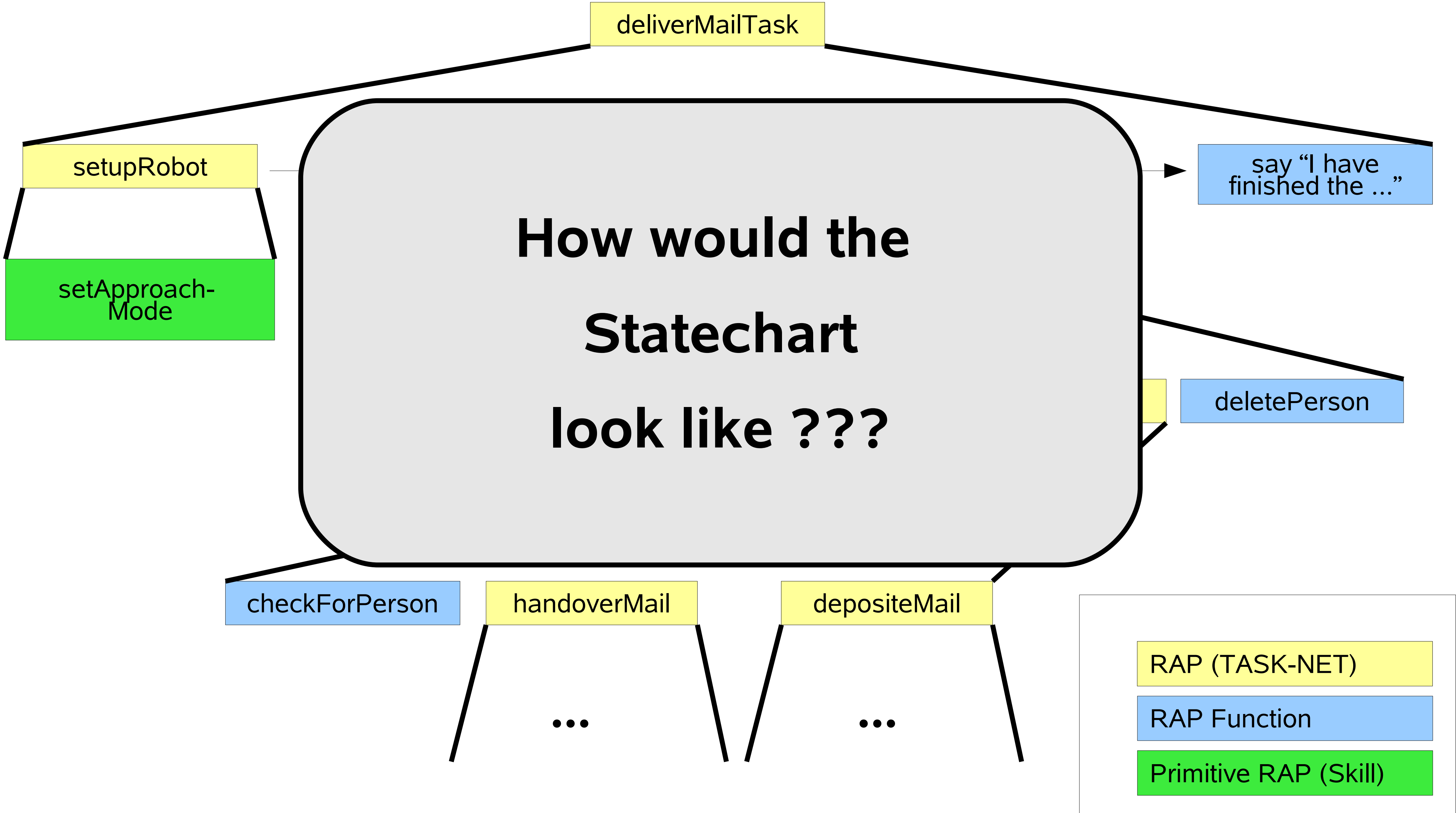
…

🟨 RAP (TASK-NET)

🟦 RAP Function

🟩 Primitive RAP (Skill)

Behavior Modeling
Example: Mail Delivery



deliverMailTask

setupRobot

setApproach-Mode

say “I have finished the …”

deletePerson

checkForPerson

handoverMail

depositeMail

…

…

How would the Statechart look like ???

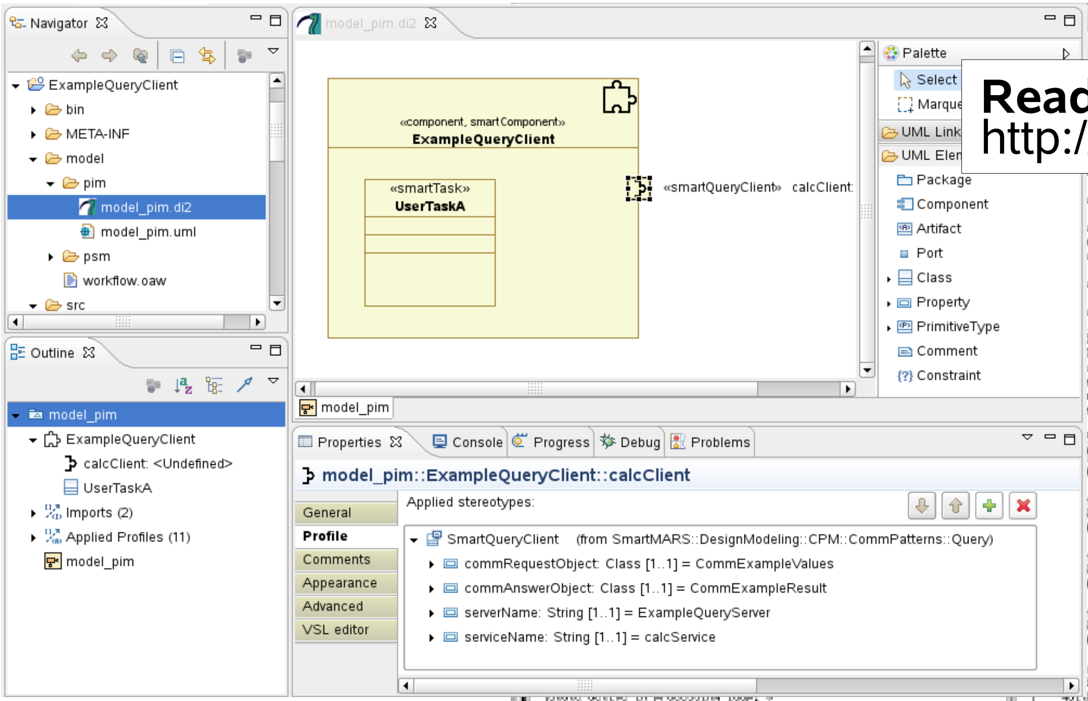
🟨 RAP (TASK-NET)

🟦 RAP Function

🟩 Primitive RAP (Skill)

Summary and Conclusion

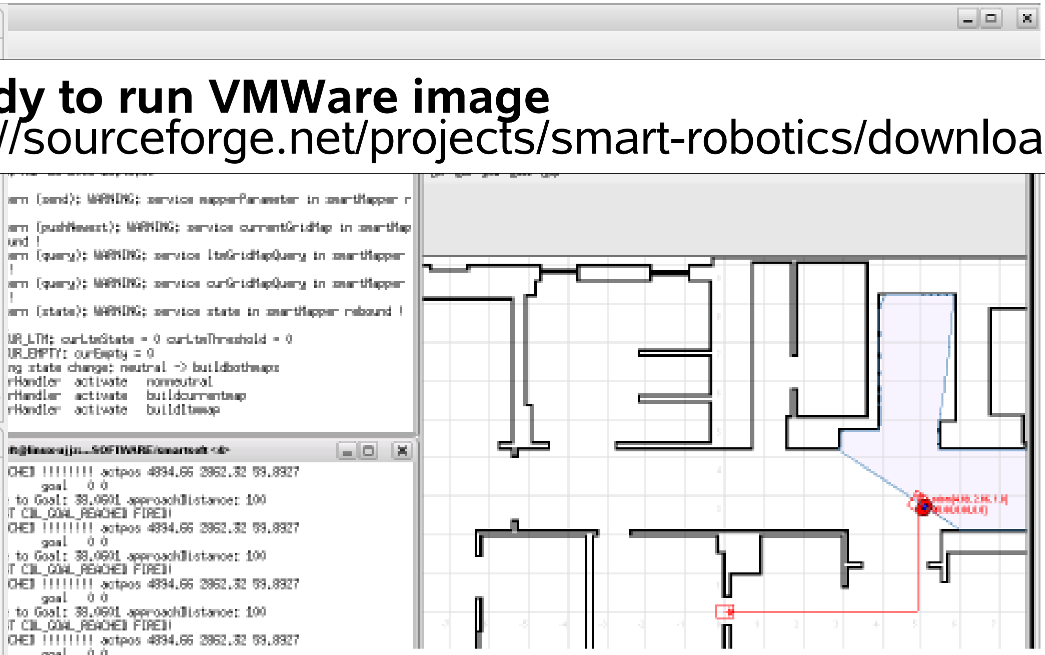
Ready to run VMWare image
http://sourceforge.net/projects/smart-robotics/download



model_pim::ExampleQueryClient::calcClient

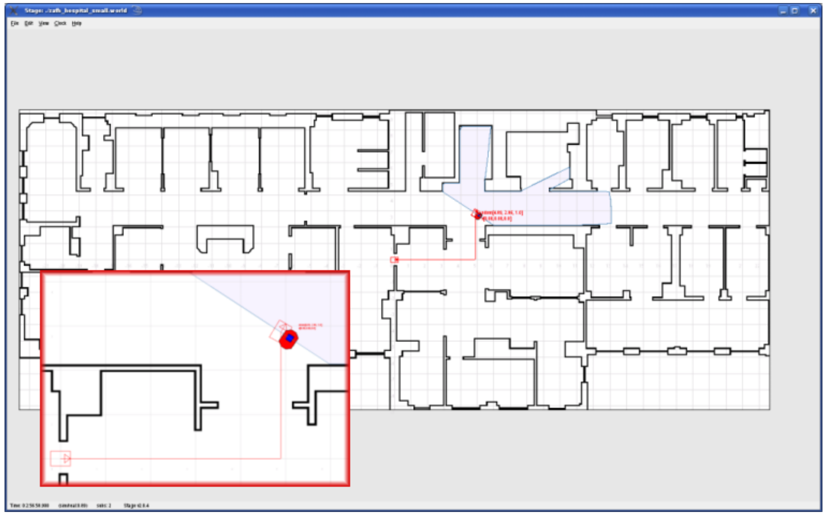
Applied stereotypes:

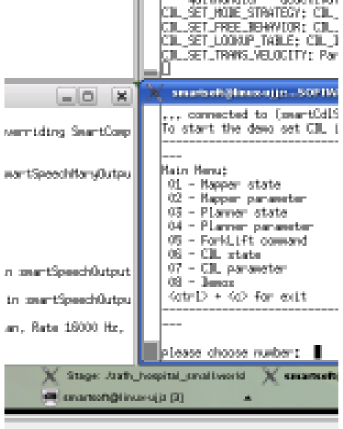
- SmartQueryClient (from SmartMARS::DesignModeling::CPM::CommPatterns::Query)
  - commRequestObject: Class [1..1] = CommExampleValues
  - commAnswerObject: Class [1..1] = CommExampleResult
  - serverName: String [1..1] = ExampleQueryServer
  - serviceName: String [1..1] = calcService



```
arn (send): WARNING: service mapperParameter in smartMapper r

arn (pushNewest): WARNING: service currentGridMap in smartMap
und !
arn (query): WARNING: service ltmGridMapQuery in smartMapper
!
arn (query): WARNING: service curGridMapQuery in smartMapper
!
arn (state): WARNING: service state in smartMapper rebound !

UR_LTM: curLtmState = 0 curLtmThreshold = 0
UR_EMPTY: curEmpty = 0
ng state change: neutral -> buildbothmaps
rHandler  activate   nonneutral
rHandler  activate   buildcurrentmap
rHandler  activate   buildltmmap
```

```
CHED !!!!!!!! actpos 4894.66 2862.32 59.8927
      goal   0 0
 to Goal: 38.0601 approachDistance: 100
T CDL_GOAL_REACHED FIRED!
CHED !!!!!!!! actpos 4894.66 2862.32 59.8927
      goal   0 0
 to Goal: 38.0601 approachDistance: 100
T CDL_GOAL_REACHED FIRED!
CHED !!!!!!!! actpos 4894.66 2862.32 59.8927
      goal   0 0
 to Goal: 38.0601 approachDistance: 100
T CDL_GOAL_REACHED FIRED!
CHED !!!!!!!! actpos 4894.66 2862.32 59.8927
      goal   0 0
```





```
ng state change: mo
Handler   deactivat
Handler   deactivat
CDL_SET_MODE_STRATEGY: CDL_
CDL_SET_FREE_BEHAVIOR: CDL_
CDL_SET_LOOKUP_TABLE: CDL_I
CDL_SET_TRANS_VELOCITY: Par
```

```
verriding SmartComp

martSpeechManyOutpu

n smartSpeechOutput

in smartSpeechOutpu

an, Rate 16000 Hz,
```

```
... connected to (smartCdlS
To start the demo set CDL i
------------------------------
---
Main Menu:
 01 - Mapper state
 02 - Mapper parameter
 03 - Planner state
 04 - Planner parameter
 05 - ForkLift command
 06 - CDL state
 07 - CDL parameter
 08 - Demos
 <ctrl> + <c> for exit
------------------------------
---

please choose number:
```

Summary and Conclusion

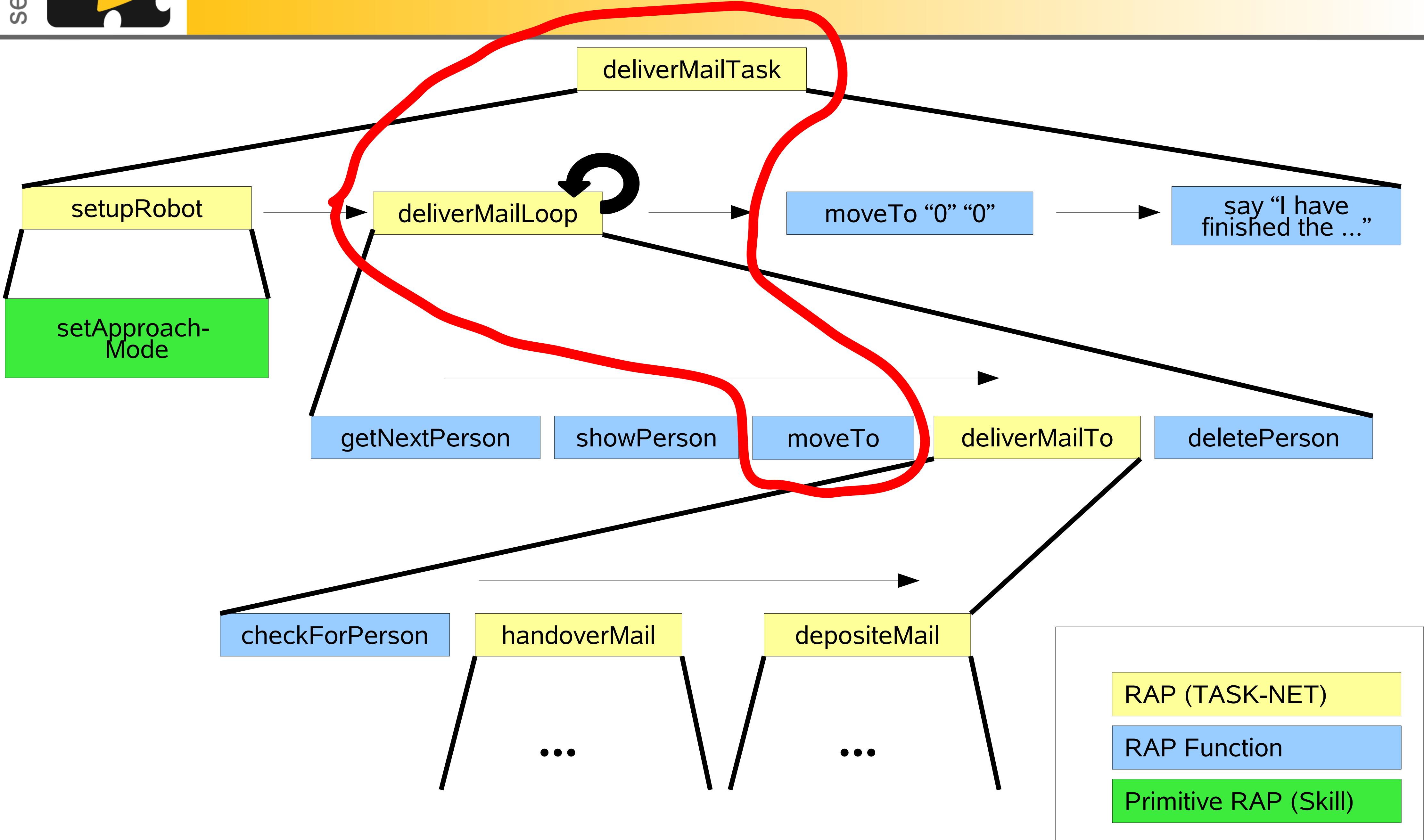
MDSD Toolchain - Screencast

ZAFH Ulm video2 05-2009.swf

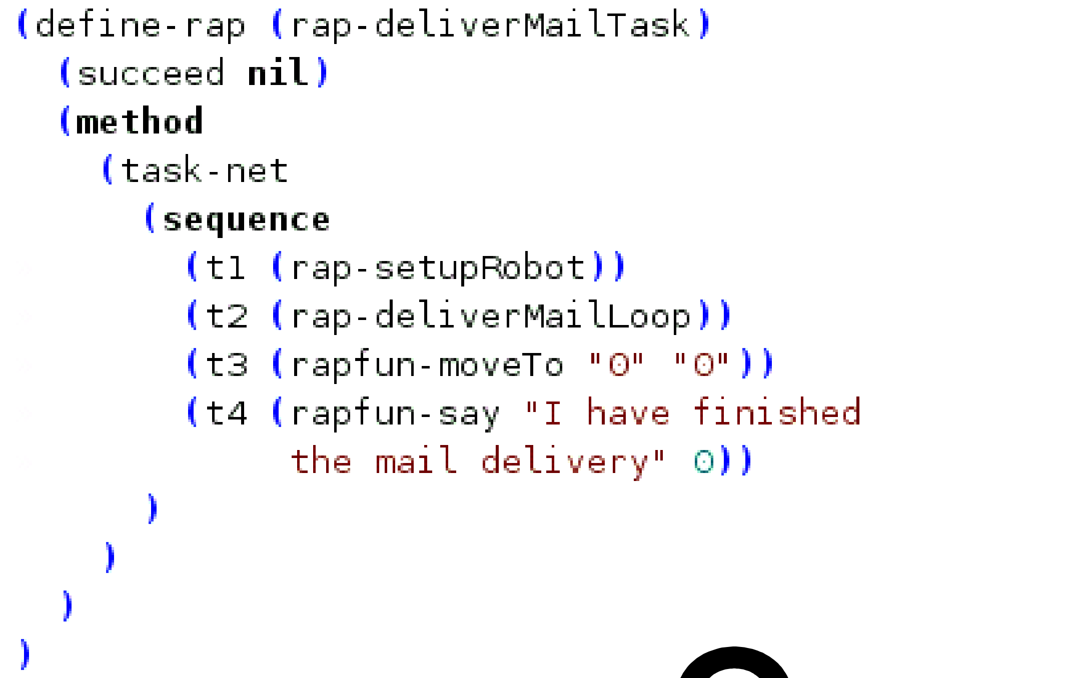
http://www.zafh-servicerobotik.de/ULM/en/dokumente/ZAFH Ulm video2 05-2009.swf

http://smart-robotics.sourceforge.net/

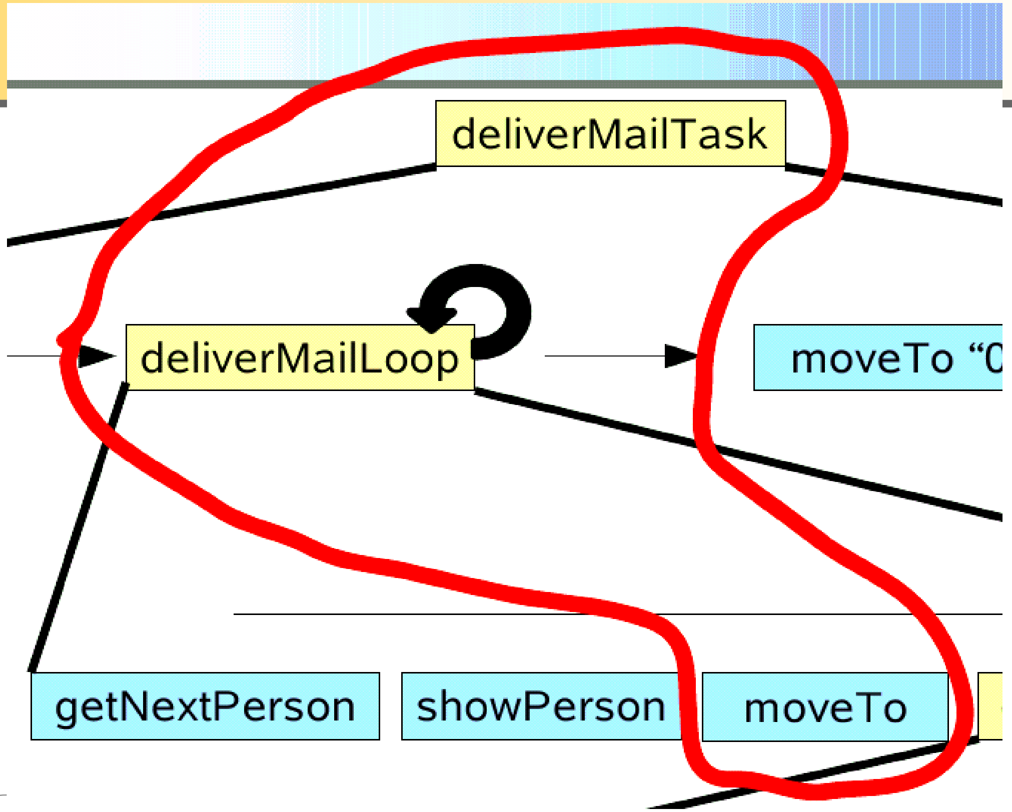
Behavior Modeling
Example: Mail Delivery

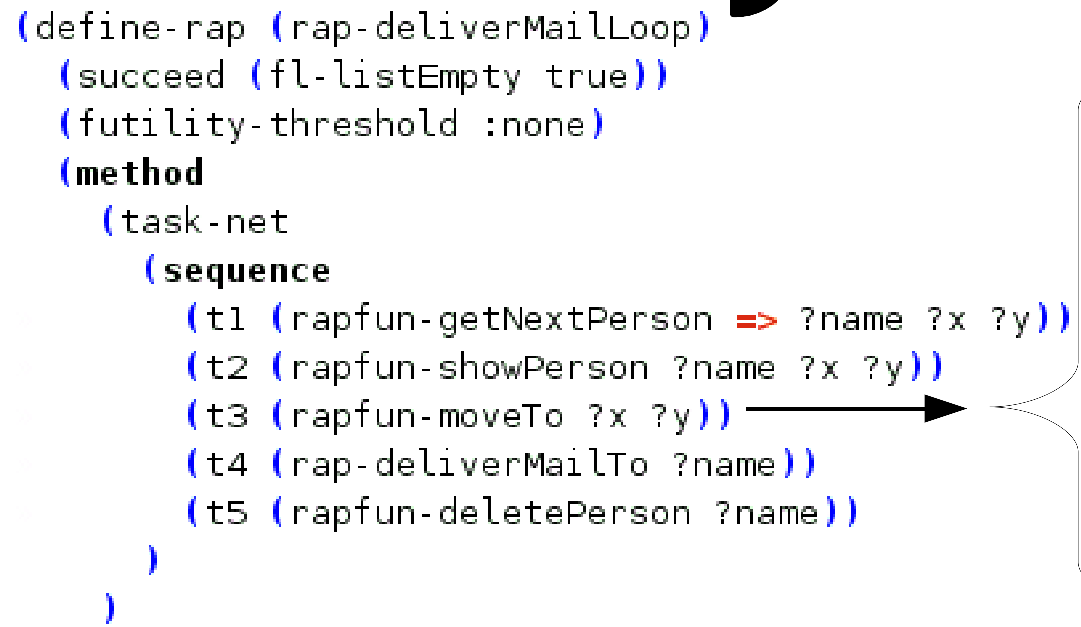


Behavior Modeling

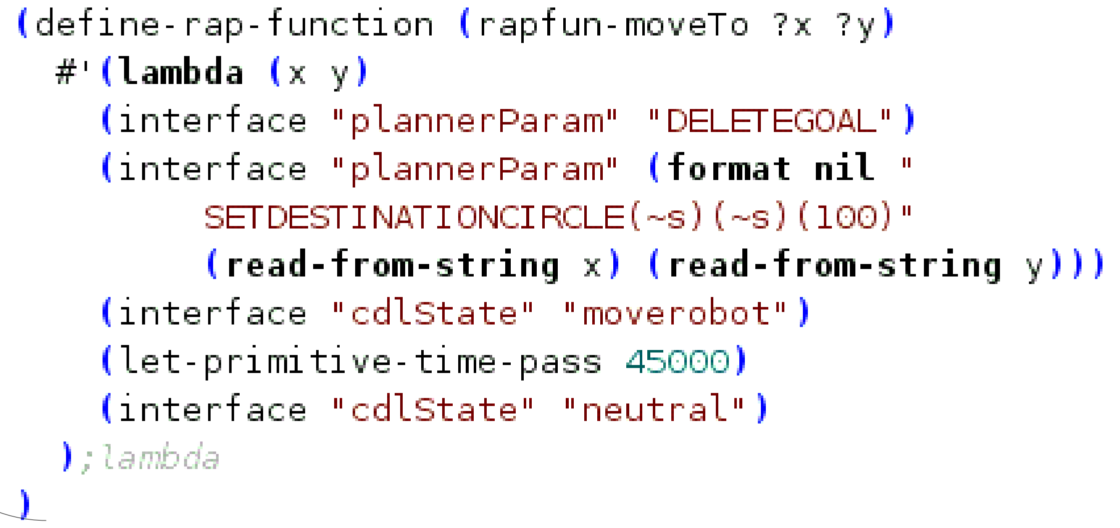


```
(define-rap (rap-deliverMailTask)
  (succeed nil)
  (method
    (task-net
      (sequence
        (t1 (rap-setupRobot))
        (t2 (rap-deliverMailLoop))
        (t3 (rapfun-moveTo "0" "0"))
        (t4 (rapfun-say "I have finished
             the mail delivery" 0))
      )
    )
  )
)
```





```
(define-rap (rap-deliverMailLoop)
  (succeed (fl-listEmpty true))
  (futility-threshold :none)
  (method
    (task-net
      (sequence
        (t1 (rapfun-getNextPerson => ?name ?x ?y))
        (t2 (rapfun-showPerson ?name ?x ?y))
        (t3 (rapfun-moveTo ?x ?y))
        (t4 (rap-deliverMailTo ?name))
        (t5 (rapfun-deletePerson ?name))
      )
    )
```



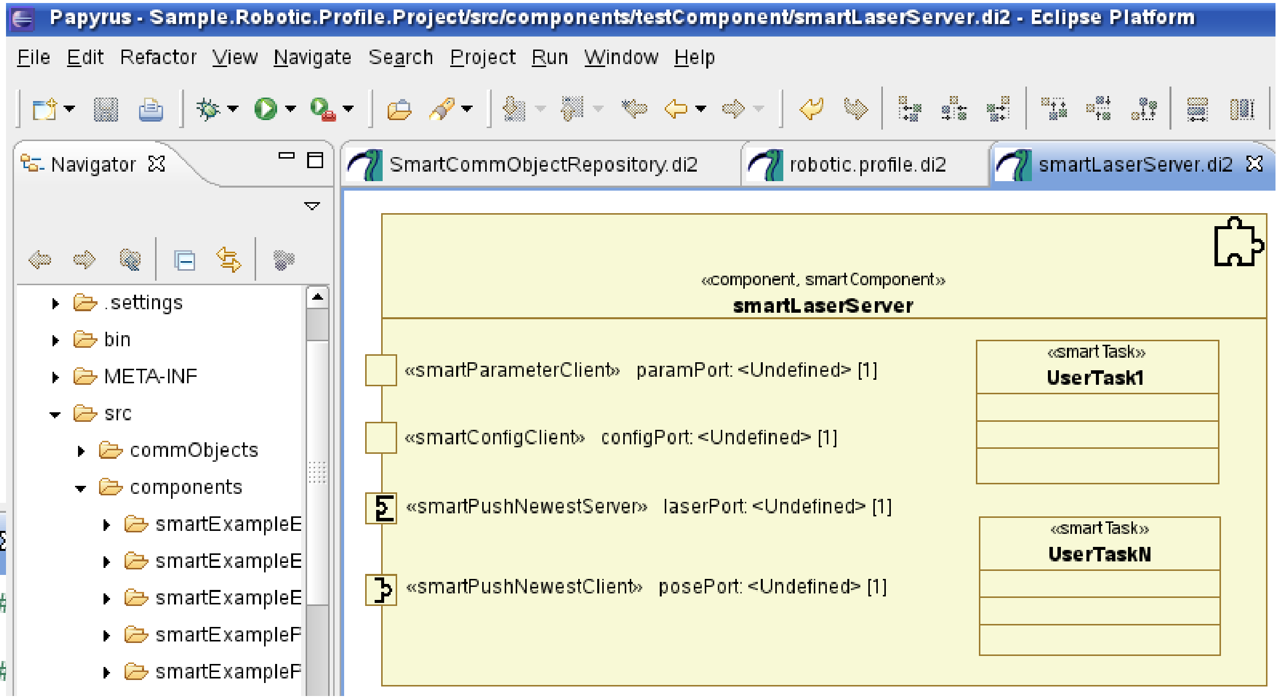
```
(define-rap-function (rapfun-moveTo ?x ?y)
  #'(lambda (x y)
    (interface "plannerParam" "DELETEGOAL")
    (interface "plannerParam" (format nil "
         SETDESTINATIONCIRCLE(~s)(~s)(100)"
         (read-from-string x) (read-from-string y)))
    (interface "cdlState" "moverobot")
    (let-primitive-time-pass 45000)
    (interface "cdlState" "neutral")
  );lambda
)
```

References

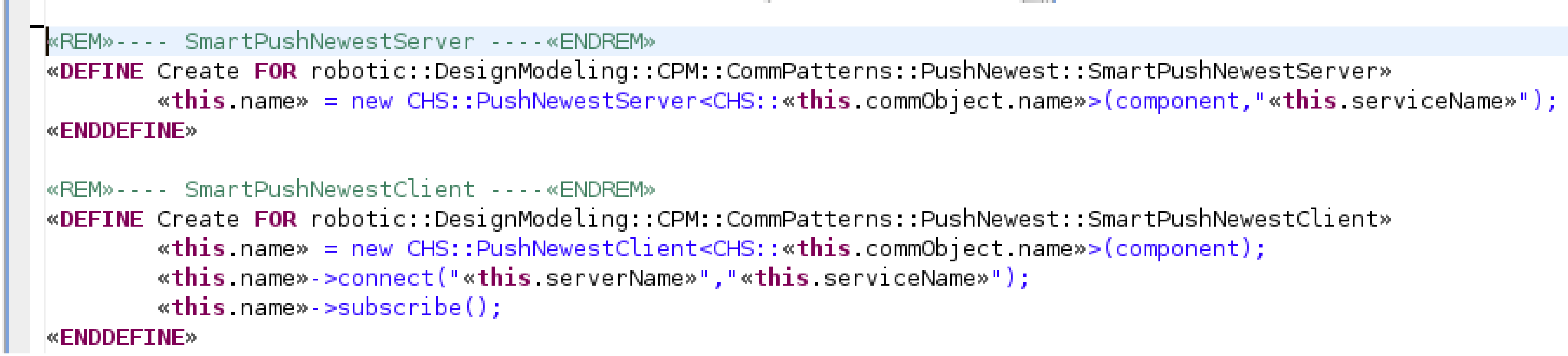
[1] R. James Firby. **Architecture, Representation and Integration**: An Example From Robot Navigation. *Fall Symposium on the Control of the Physical World by Intelligent Agents*, 1994.

Model Driven Software Development
Idea and Approach

Example of a
Model Transformation Template


Papyrus - Sample.Robotic.Profile.Project/src/components/testComponent/smartLaserServer.di2 - Eclipse Platform
File Edit Refactor View Navigate Search Project Run Window Help
Navigator
.settings
bin
META-INF
src
commObjects
components
smartExampleE
smartExampleE
smartExampleE
smartExampleF
smartExampleF
SmartCommObjectRepository.di2
robotic.profile.di2
smartLaserServer.di2
«component, smartComponent»
smartLaserServer
«smartParameterClient» paramPort: <Undefined> [1]
«smartConfigClient» configPort: <Undefined> [1]
«smartPushNewestServer» laserPort: <Undefined> [1]
«smartPushNewestClient» posePort: <Undefined> [1]
«smart Task»
UserTask1
«smart Task»
UserTaskN

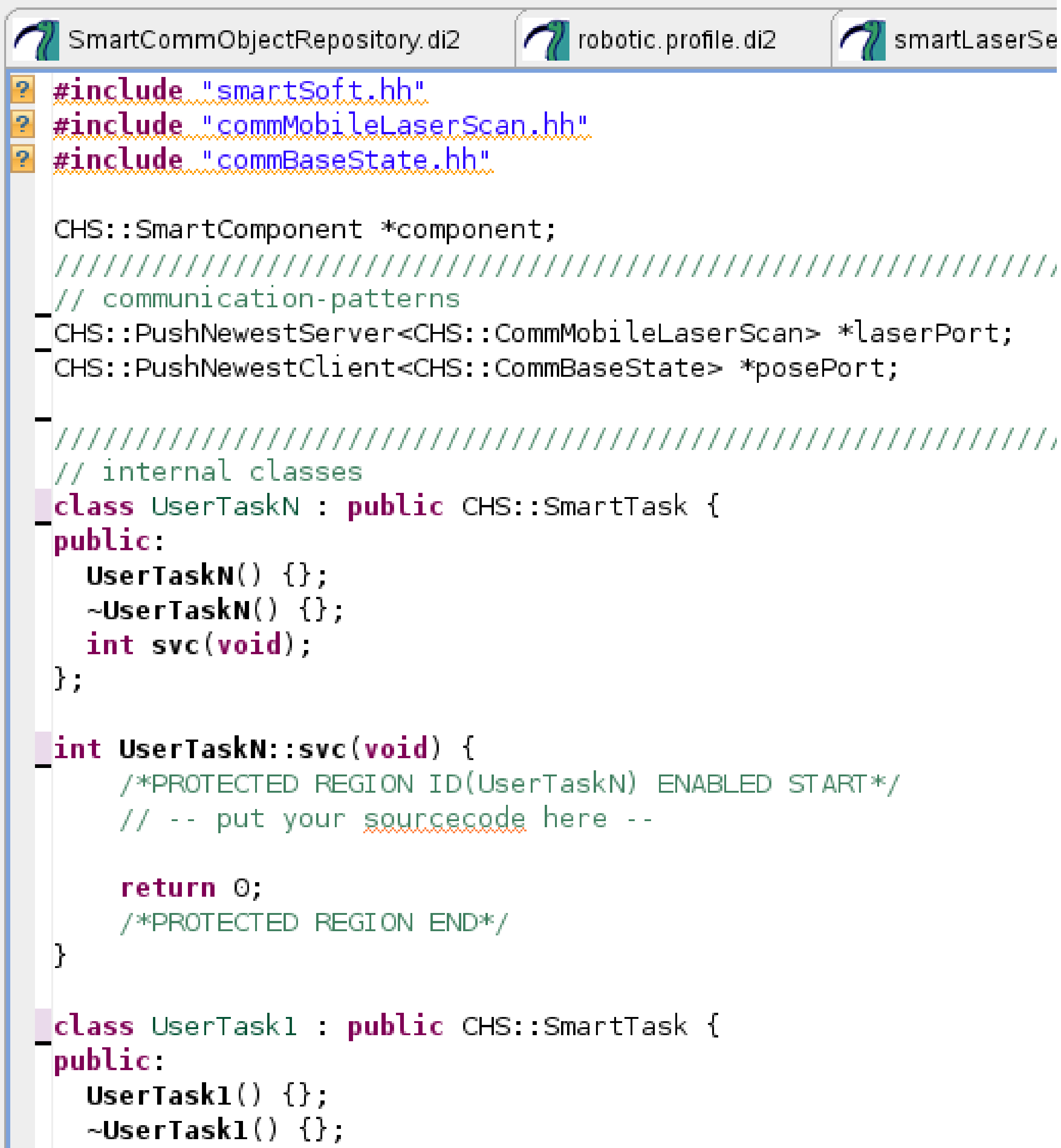
smartLaserServer.di2 | smartLaserServer.cc | *Root.xpt



```
«REM»##############################################
## creation of communication patterns
###############################################

«REM»---- SmartPushNewestServer ----«ENDREM»
«DEFINE Create FOR robotic::DesignModeling::CPM::CommPatterns::PushNewest::SmartPushNewestServer»
        «this.name» = new CHS::PushNewestServer<CHS::«this.commObject.name»>(component,"«this.serviceName»");
«ENDDEFINE»

«REM»---- SmartPushNewestClient ----«ENDREM»
«DEFINE Create FOR robotic::DesignModeling::CPM::CommPatterns::PushNewest::SmartPushNewestClient»
        «this.name» = new CHS::PushNewestClient<CHS::«this.commObject.name»>(component);
        «this.name»->connect("«this.serverName»","«this.serviceName»");
        «this.name»->subscribe();
«ENDDEFINE»
```

Hochschule Ulm

Model Driven Software Development
Idea and Approach

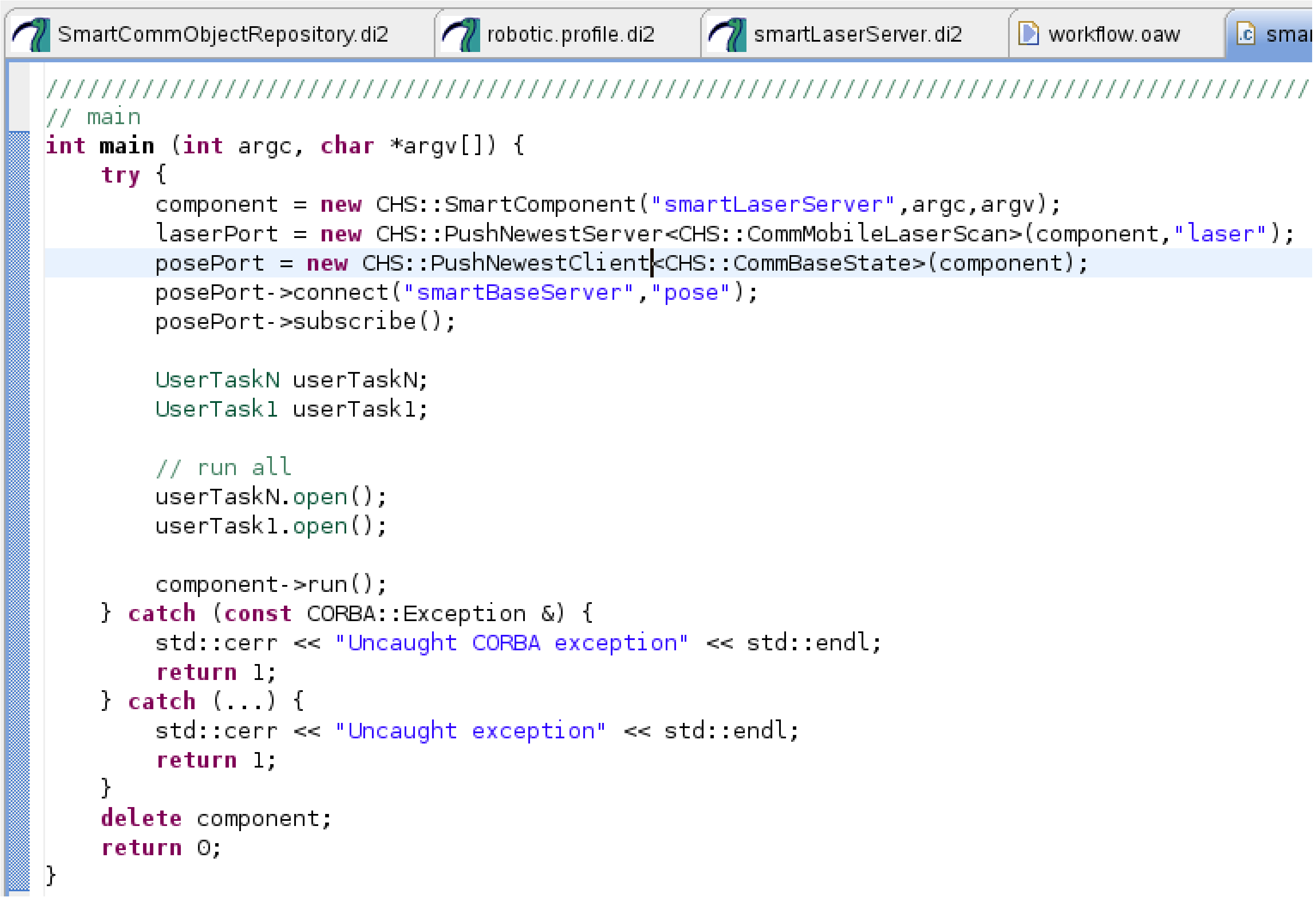


```
#include "smartSoft.hh"
#include "commMobileLaserScan.hh"
#include "commBaseState.hh"

CHS::SmartComponent *component;
/////////////////////////////////////////////////////////////////
// communication-patterns
CHS::PushNewestServer<CHS::CommMobileLaserScan> *laserPort;
CHS::PushNewestClient<CHS::CommBaseState> *posePort;

/////////////////////////////////////////////////////////////////
// internal classes
class UserTaskN : public CHS::SmartTask {
public:
  UserTaskN() {};
  ~UserTaskN() {};
  int svc(void);
};

int UserTaskN::svc(void) {
    /*PROTECTED REGION ID(UserTaskN) ENABLED START*/
    // -- put your sourcecode here --

    return 0;
    /*PROTECTED REGION END*/
}

class UserTask1 : public CHS::SmartTask {
public:
  UserTask1() {};
  ~UserTask1() {};
```

Example of generated code with protected user sections not touched by the code generator

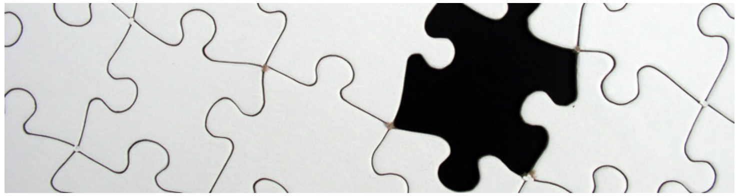
Hochschule Ulm

Model Driven Software Development
Idea and Approach



```
///////////////////////////////////////////////////////////////////////////////////////////////
// main
int main (int argc, char *argv[]) {
    try {
        component = new CHS::SmartComponent("smartLaserServer",argc,argv);
        laserPort = new CHS::PushNewestServer<CHS::CommMobileLaserScan>(component,"laser");
        posePort = new CHS::PushNewestClient<CHS::CommBaseState>(component);
        posePort->connect("smartBaseServer","pose");
        posePort->subscribe();

        UserTaskN userTaskN;
        UserTask1 userTask1;

        // run all
        userTaskN.open();
        userTask1.open();

        component->run();
    } catch (const CORBA::Exception &) {
        std::cerr << "Uncaught CORBA exception" << std::endl;
        return 1;
    } catch (...) {
        std::cerr << "Uncaught exception" << std::endl;
        return 1;
    }
    delete component;
    return 0;
}
```

Model Driven Software Development Introduction and Motivation



What is this talk about ?

- not just another software framework
- not just another middleware wrapper

➔ we have plenty of those ...

But

- separation of robotics knowledge from short-cycled implementational technologies
- providing sophisticated and optimized software structures to robotics developers not requiring them to become a software expert

How to achieve this ?

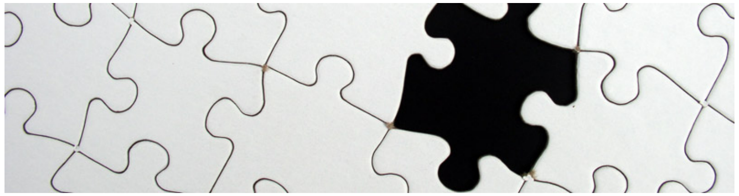
- make the step from code-driven to model-driven designs
- using common open source tools for robotics !

Model Driven Software Development Introduction and Motivation

Why is Model Driven Software Development important in Robotics ?

- get rid of hand-crafted single unit service robot systems
- compose them out of standard components with explicitly stated properties
- be able to reuse / modify solutions expressed at a model level
- take advantage from the knowledge of software engineers that is encoded in the code transformation rules / hidden structures
- be able to verify (or at least provide conformance checks) properties

and many many more good reasons



Engineering the software development process in robotics is one of the basic necessities towards industrial-strength service robotic systems

Model Driven Software Development
Idea and Approach

That sounds good but give me an example ...

we made some very simple but pivotal decisions:

- granularity level for system composition:
  - loosely coupled components
  - services provided and required
- strictly enforced interaction patterns between components
  - precisely defined semantics of intercomponent interaction
  - these are policies (and can be mapped onto any middleware mechanism)
  - ➔ *independent of a certain middleware*
- minimum component model to support system integration
  - dynamic wiring of the data flow between components
  - state automaton to allow for orchestration / configuration
  - ➔ *ensures composability / system integration*
- execution environment independently
  - tasks (periodic, non-periodic, hard real-time, no realtime), synchronization, resource access
  - ➔ *again, can be mapped onto different operating systems*