AWASE 2017

Architectures for Adaptation

Jeff Kramer

Imperial College London

change

…. the challenge of **change** …

environment **E**
goals **G**
capabilities **I**

…. to be aware and monitor these sources of change.

off-line …
requirements analysis, design, implementation, redeployment ….

run-time …
adaptation to unforeseen changes ….

change

Off-line software evolution

Run-time software adaptation

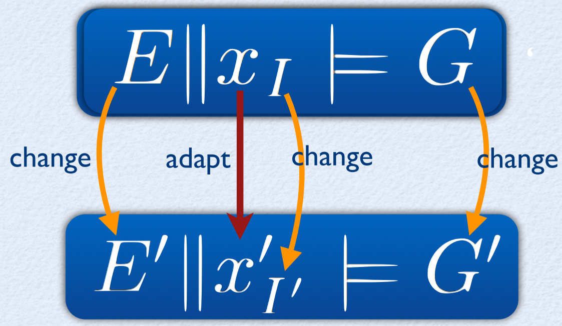
pre-planned change (maintenance)

unforeseen change

…. to automate and run on-line what is currently done off-line!

Adaptive and Self-Managed Systems

E - assumed environment behaviour
G - requirements goals of system
I - interface capabilities of the system **x**



Adaptive and Self-Managed Systems

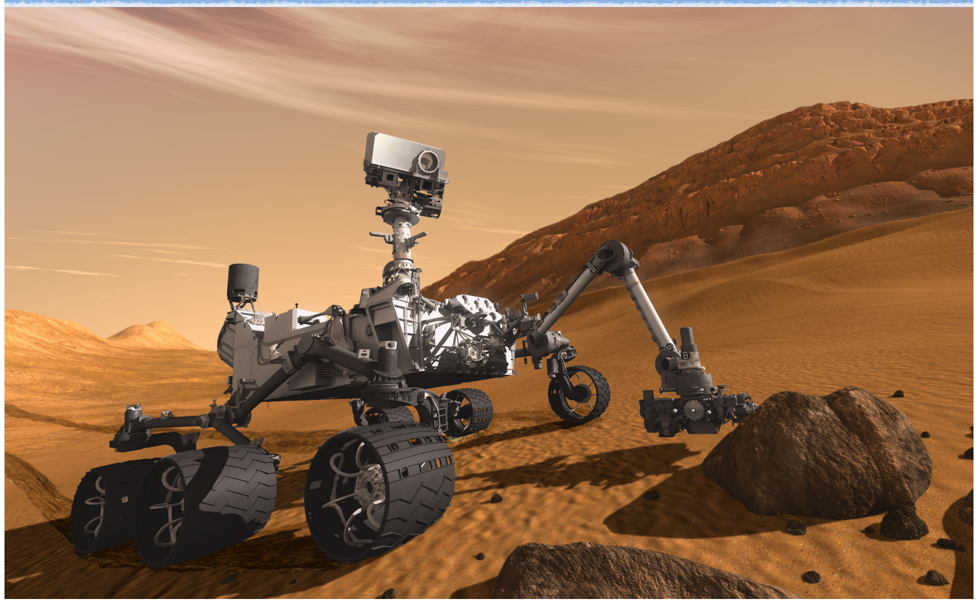
Adaptive *light* : adjustment of runtime parameters in response to degraded performance or failure

Adaptive *full fat* : changes in functionality and performance in response to **unforeseen** changes in the environment, goals and/or capabilities of the system

Adaptive and S

Disruptive change!

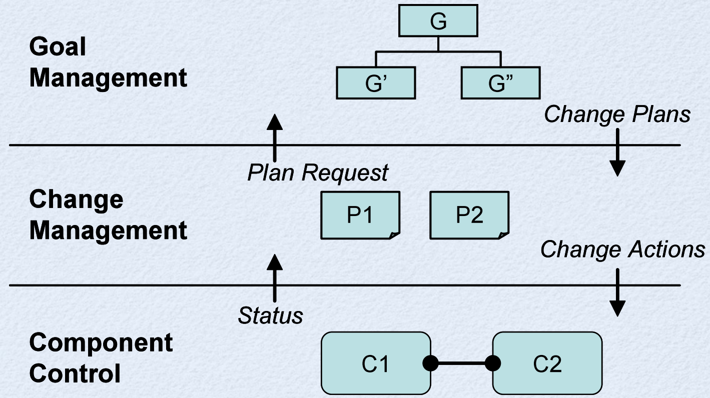
Adaptive and Self-Managed Systems



Adaptive and Self-Managed Systems

architecture is important

three layer architecture



1. Planning over abstract domain

2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

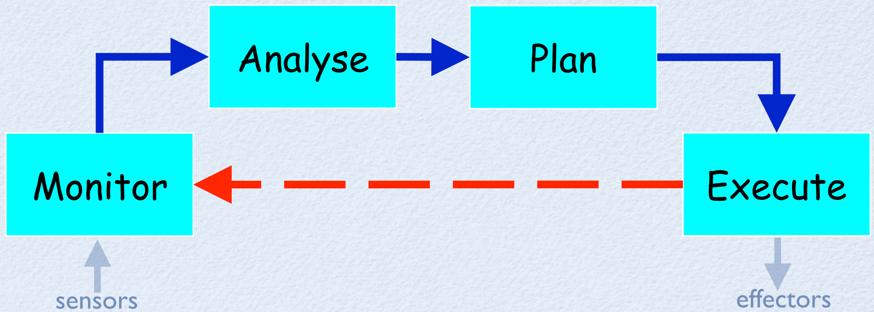
ICSE FOSE '07

three layer architecture

?

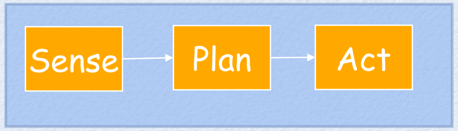
- why this architecture?
- how did we get here?
- where are we going?

MAPE cycle

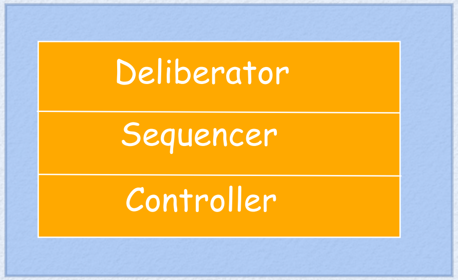


- a single feedback loop?
- response times?
- complexity?

inspiration from robotics



- 1970's



- 1998 (Gat)

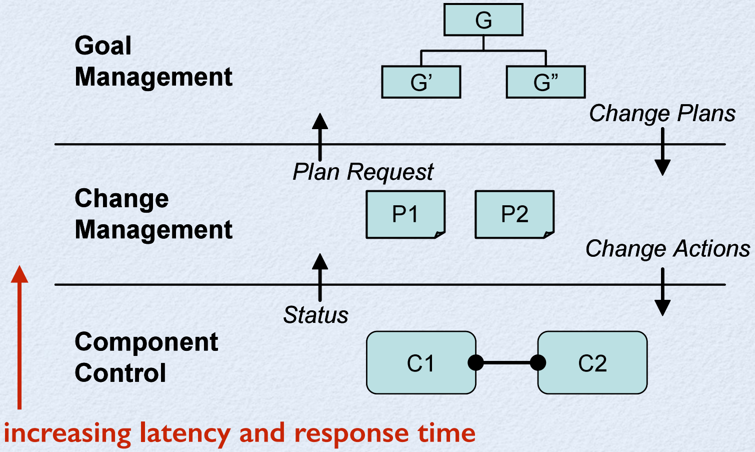
1. Planning

2. plan execution

3. component feedback control

- layering according to response times

three layer architecture



1. Planning over abstract domain

2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

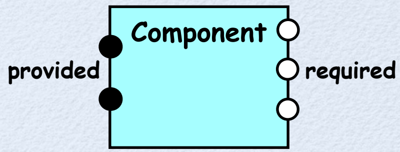
- a separation of concerns

ICSE FOSE '07

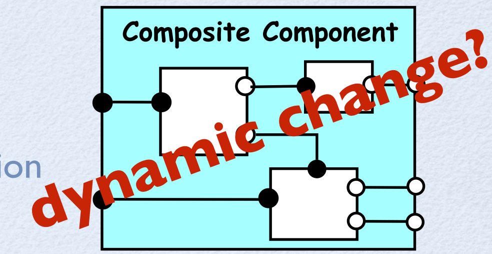
… some of our earlier research …

CONIC and Darwin

- distributable, context-independent components
  - interaction via a well-defined interface



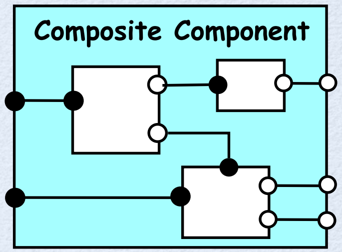
- an explicit configuration description (ADL)
  - third party instantiation and binding



TSE 1985, TSE 1989, ESEC/FSE 1995, FSE 1996

CONIC and Darwin

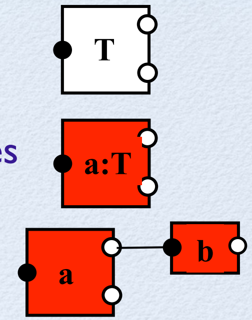
- on-line dynamic change
  - once installed, the software could be dynamically modified without stopping the entire system



TSE 1985, TSE 1989, ESEC/FSE 1995, FSE 1996

on-line dynamic change

- **load** component type
- **create/delete** component instances
- **bind/unbind** component services

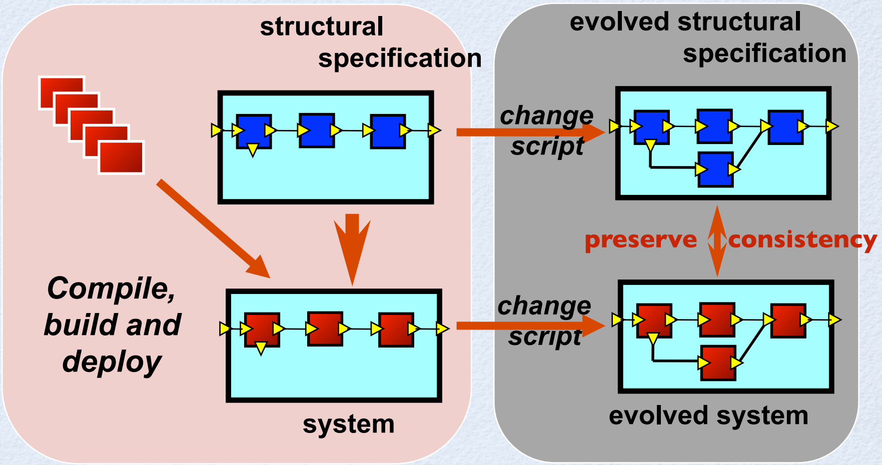


How can we do this *safely?*

How can we maintain *configuration consistency* and *behaviour consistency* during the change?

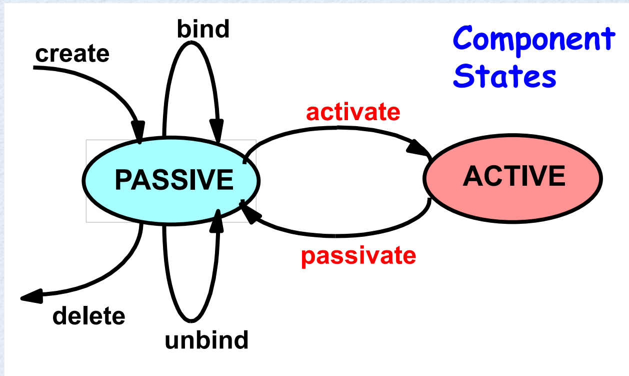
TSE 1985

configuration consistency



TSE 1985

behaviour consistency



General change model:

Separate the specification of *structural change* from the component *application behaviour.*

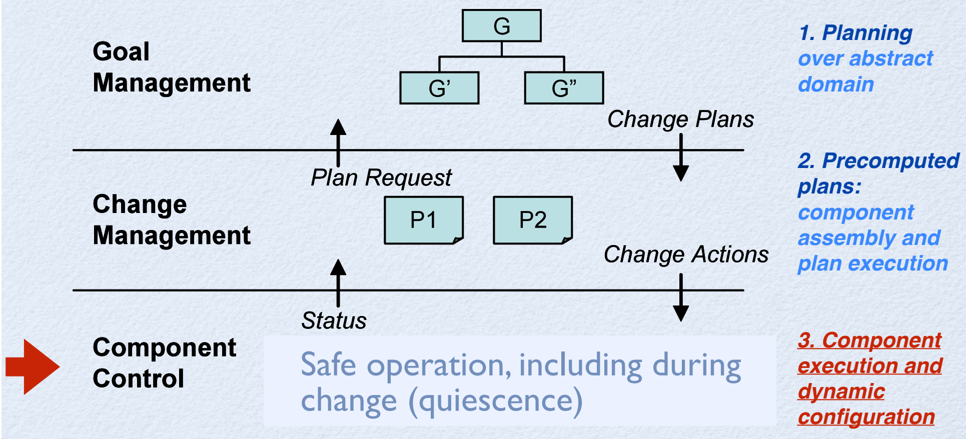
Passive component services interactions, but does not initiate new ones i.e. acts to preserve consistency.

Quiescent : passive and no transactions will be initiated on it (ie. the environment is passive)

TSE 1990

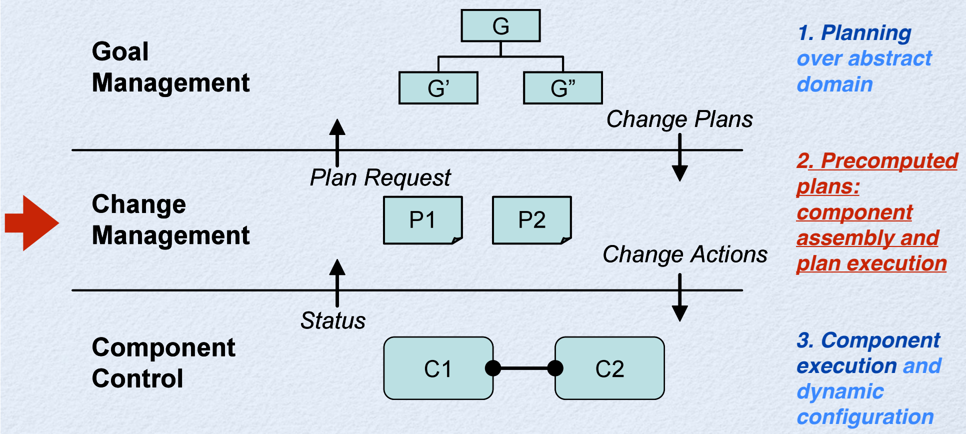
safe configuration and reconfiguration of components

three layer architecture



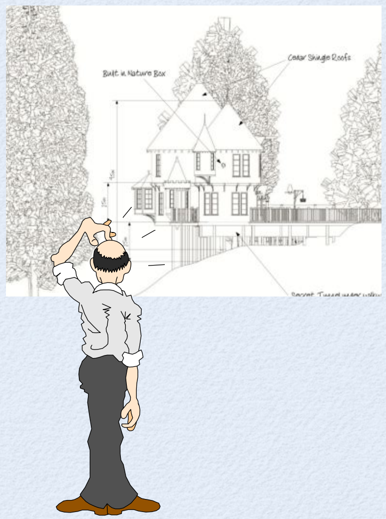
ICSE FOSE '07, SAVCBS 2007, SEAMS 2008

three layer architecture



ICSE FOSE '07, SAVCBS 2007, SEAMS 2008

component assembly? plan execution?



plan execution

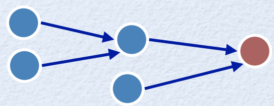
plan execution

```
...
AT.loc1  && !LOADED
      -> pickup
AT.loc1  && LOADED
      -> moveto.loc2
AT.loc2  && LOADED
      -> putdown
AT.loc2  && !LOADED
      -> moveto.loc1
...
```

Reactive plans

- condition-**action** rules over an alphabet of plan actions

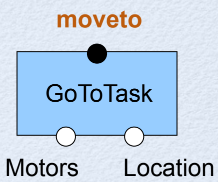
Includes alternative paths to the goals if there are unpredicted environment changes



component assembly

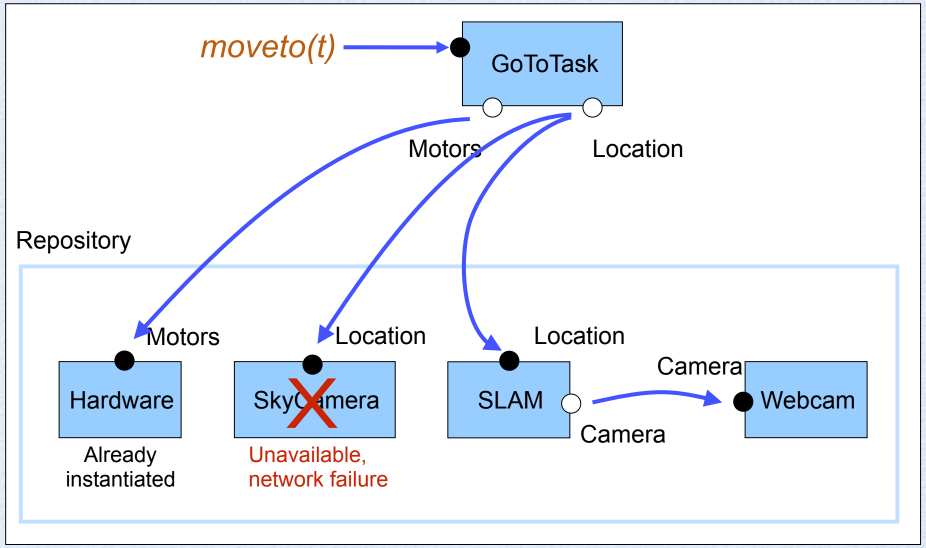
Derive configurations by mapping plan actions to components :

- primitive plan actions (pickup, moveto,…) are associated with the provided services of components which the plan interpreter can call
- elaborate and assemble components using dependencies (required services)

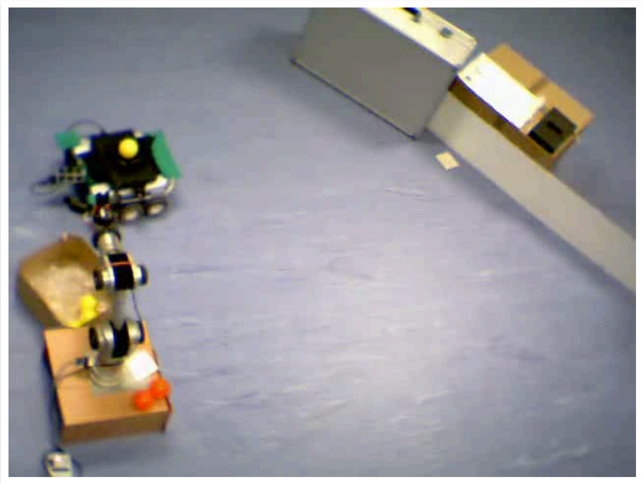


Mapping is a many to many relationship, providing alternatives

component assembly



adaptation demonstration



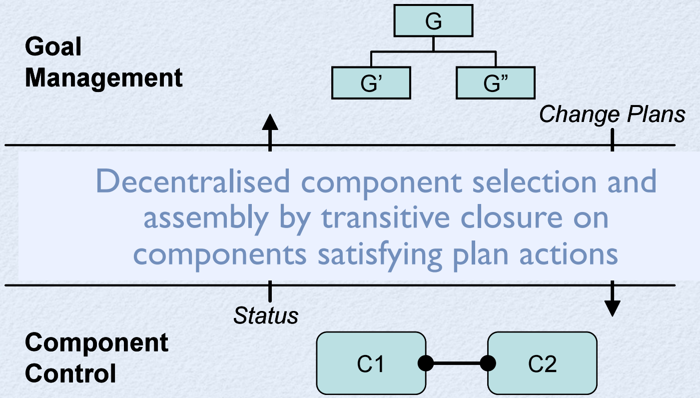
Adaptation may require component reselection
or
alternative plan selection
or
replanning

… other assembly explorations …

- **Flashmob - distributed adaptive self-assembly**
  - gossip algorithm
- **Exploiting NF preferences in architectural adaptation for self-managed systems**
  - component annotations and utility function optimisation

SEAMS 2011, SAC 2010

three layer architecture



Decentralised component selection and assembly by transitive closure on components satisfying plan actions

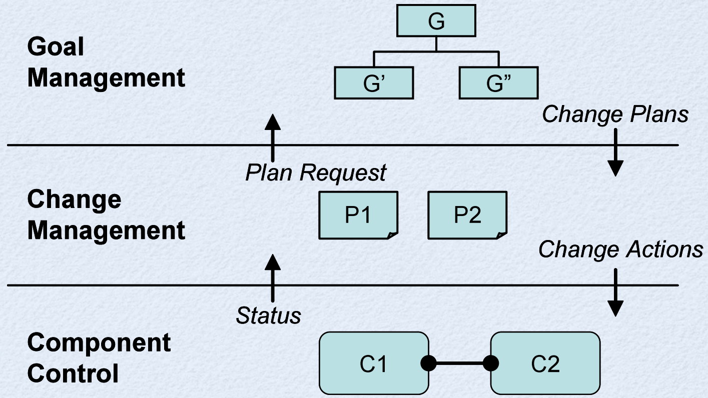
1. Planning over abstract domain

2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

ICSE FOSE '07, SEAMS 2008, SEAMS 2011

three layer architecture



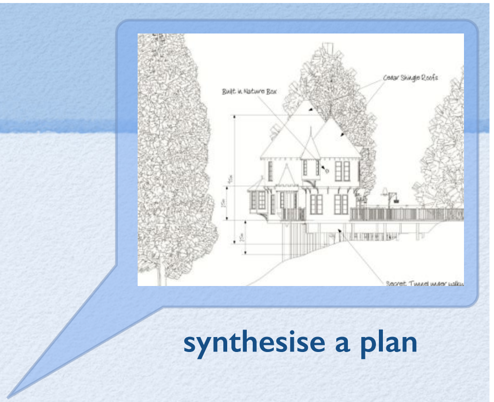
1. Planning over abstract domain

2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

ICSE FOSE '07, SEAMS 2008, SEAMS 2011

goal

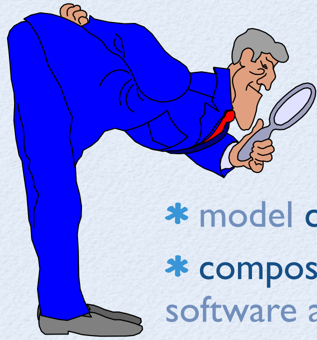


synthesise a plan

build a model

model-based planning

…earlier modelling research…



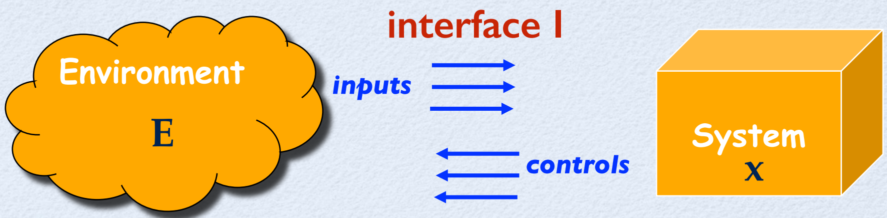
- model **component** behaviour as **LTS** in **FSP**
- **compose** behaviours according to the software architecture configuration

… **model check** properties using **LTSA**

ICSE '96, TOSEM '96, FSE '97, ESEC/FSE '99, book '99/2006

plan (controller) synthesis

Consider a plan as a winning strategy in an infinite two player game between the **environment E** and the **system x** with **interface I** such that goal G is always satisfied no matter what the order of inputs from environment.



|| composition of LTS

$$E \| x_I \models G$$

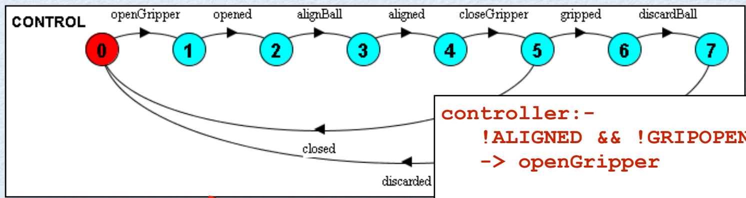
synthesise x

Goal G: Linear Temporal Logic property

Symbolic Controller Synthesis for Discrete and Timed Systems, Asarin, Maler & Pnueli, LNCS 999, 1995.

plan (controller) synthesis

Environment model (as || LTS)




```
controller:-
   !ALIGNED && !GRIPOPEN && !PICKEDUP
   -> openGripper

   !ALIGNED && GRIPOPEN && !PICKEDUP
   -> alignBall

   !ALIGNED && !GRIPOPEN && PICKEDUP
   -> discardBall

   ALIGNED && GRIPOPEN && !PICKEDUP
   -> closeGripper
```

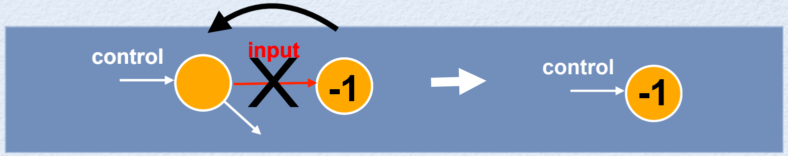
```
ltl_property SAFE4 =
    [](closeGripper -> ALIGNED)
ltl_property GETBALL =
    [](alignBall -> X closeGrippe
ltl_property PROGRESS =
    [](openGripper -> X alignBall)
```

Plan (as a controller)

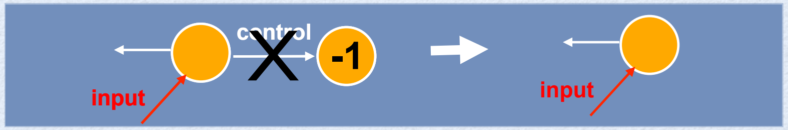
Goal specification (as LTL properties)

computing "winning" states

- By backward propagation of error state for **inputs**:



- … for **controls**:

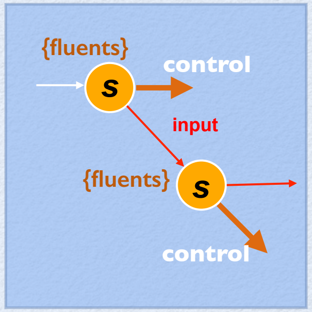


plan extraction

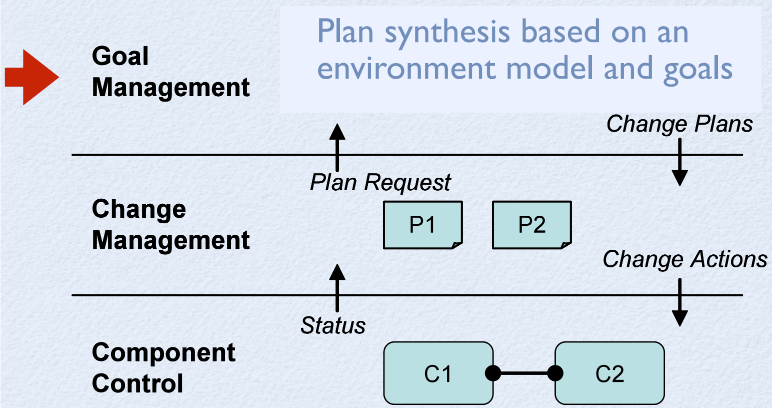
Reactive Plan computed from set of control states **S**
(has outgoing transition labelled with control)

- Label states with fluent values
- Fluents form the preconditions for the control actions.

```
controller:-
   !ALIGNED && !GRIPOPEN && !PICKEDUP
   -> openGripper

   !ALIGNED && GRIPOPEN && !PICKEDUP
   -> alignBall

   !ALIGNED && !GRIPOPEN && PICKEDUP
   -> discardBall

   ALIGNED && GRIPOPEN && !PICKEDUP
   -> closeGripper
```



three layer architecture

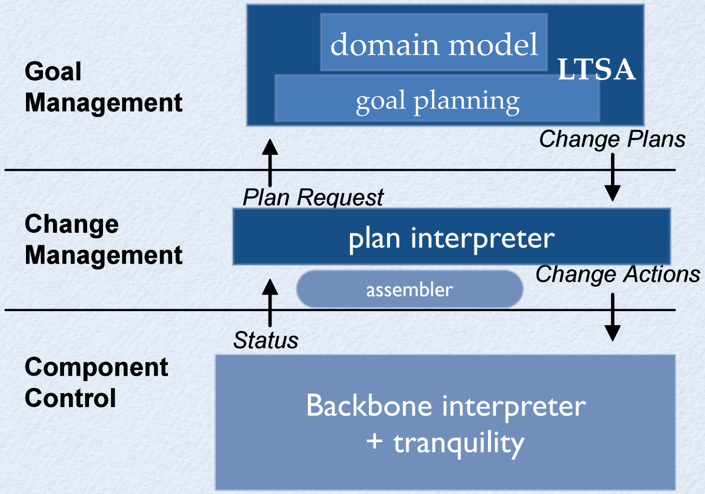


1. Planning over abstract domain

2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

three layer architecture **realisation**

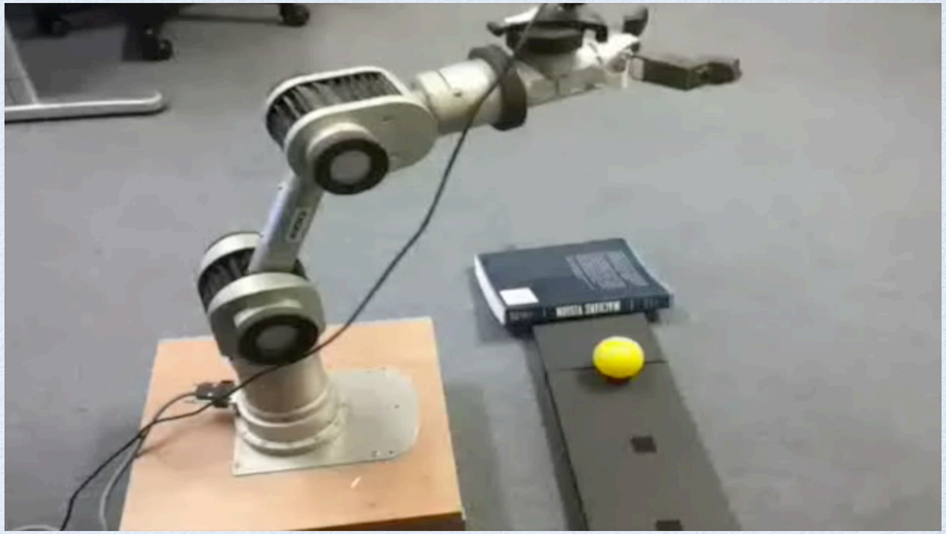


1. Planning over abstract domain

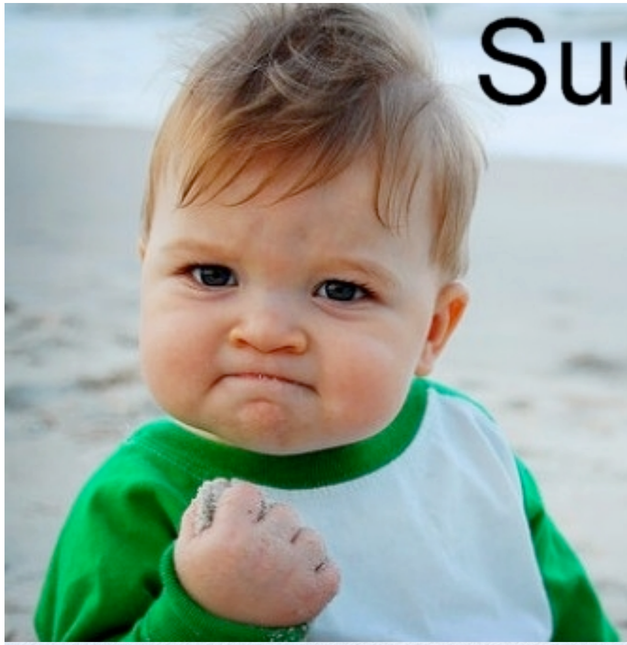
2. Precomputed plans: component assembly and plan execution

3. Component execution and dynamic configuration

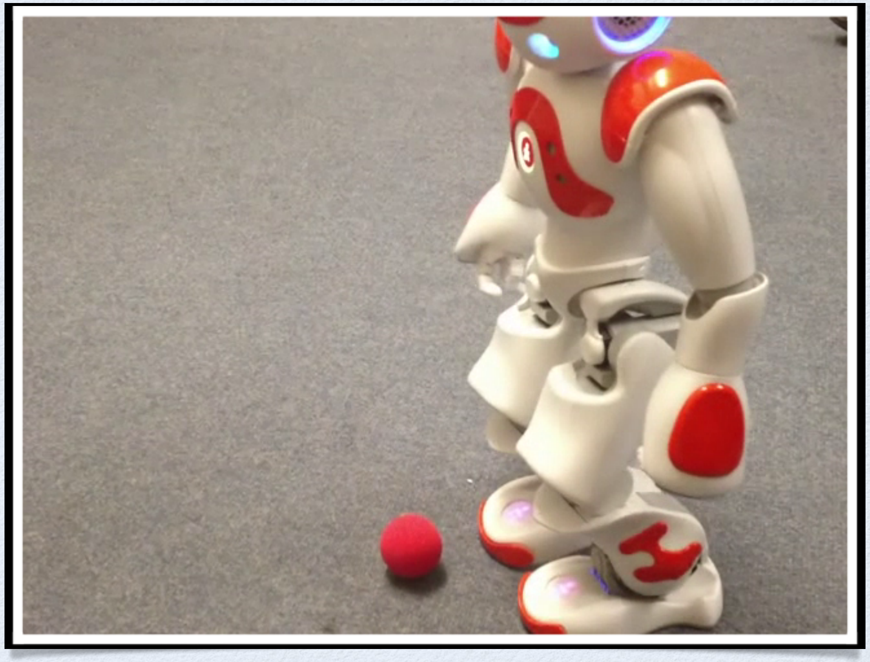
three layer architecture **realisation**



ICSE FOSE '07, SEAMS 2008, SEAMS 2011



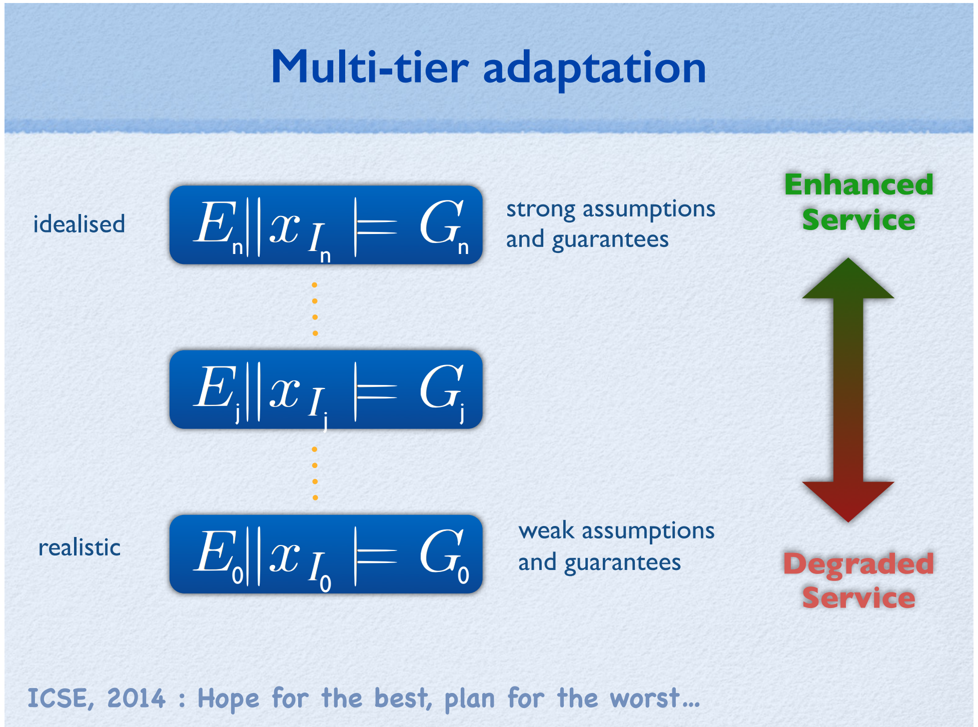
… mostly …

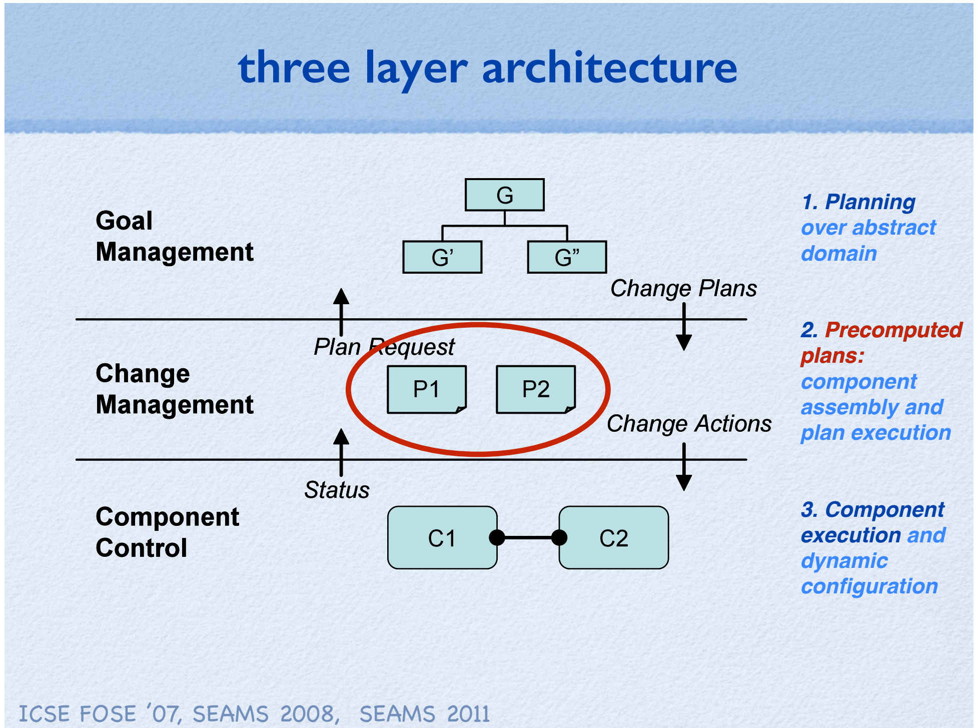


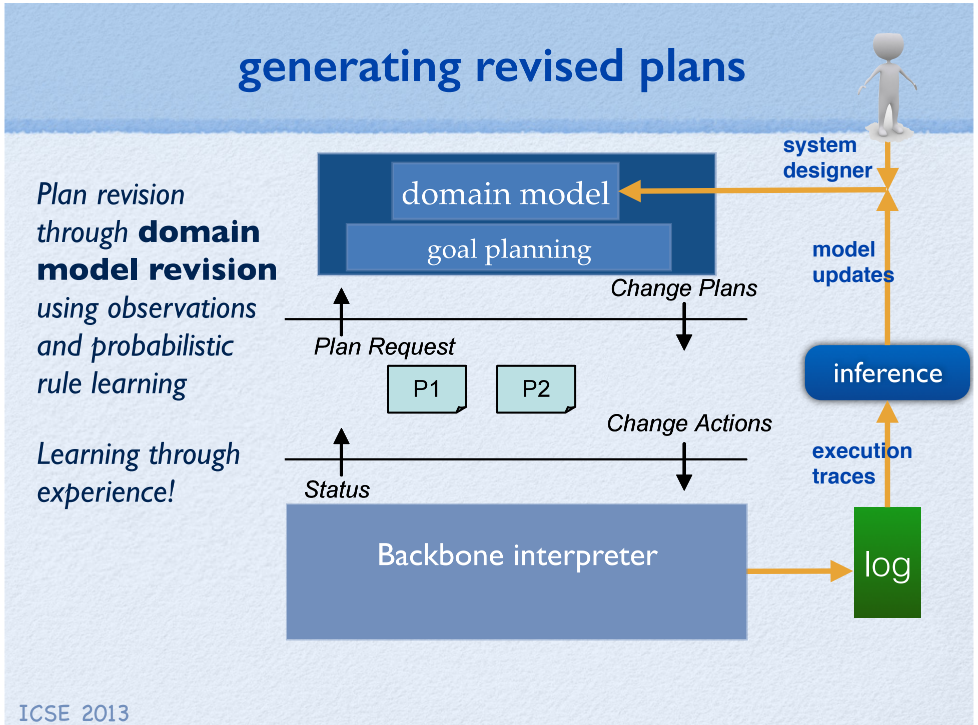
ICSE 2013 teaser demo

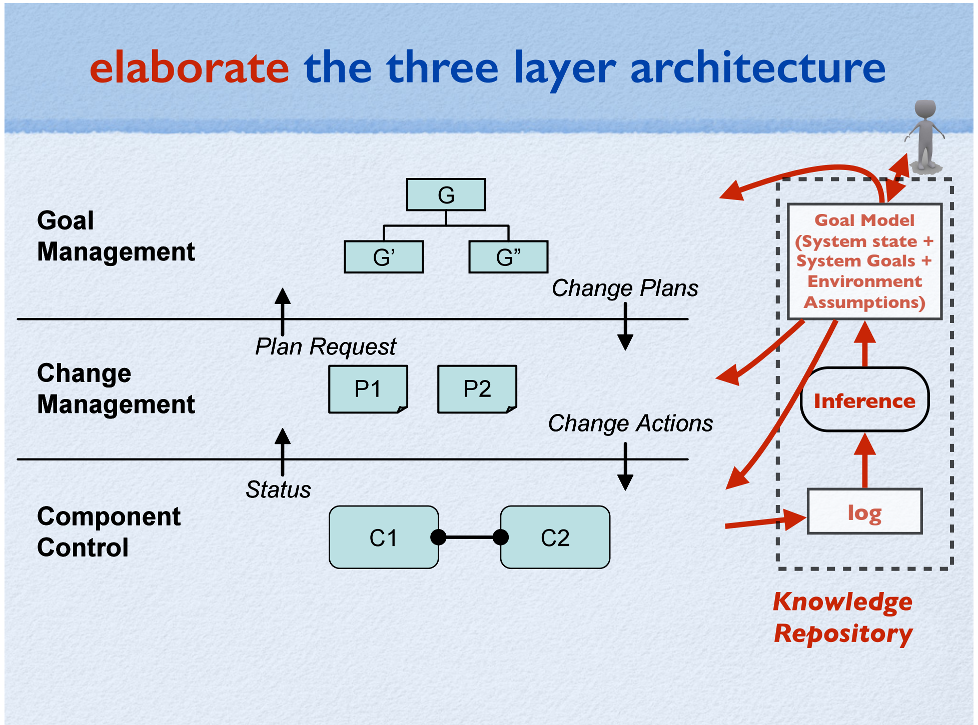
!

- provided basis for further research …


Multi-tier adaptation
idealised
$E_n \| x_{I_n} \models G_n$
strong assumptions and guarantees
$E_j \| x_{I_j} \models G_j$
realistic
$E_0 \| x_{I_0} \models G_0$
weak assumptions and guarantees
Enhanced Service
Degraded Service
ICSE, 2014 : Hope for the best, plan for the worst…


three layer architecture
Goal Management
G
G'
G"
Change Plans
Plan Request
Change Management
P1
P2
Change Actions
Status
Component Control
C1
C2
1. Planning over abstract domain
2. Precomputed plans: component assembly and plan execution
3. Component execution and dynamic configuration
ICSE FOSE '07, SEAMS 2008, SEAMS 2011


generating revised plans
Plan revision through domain model revision using observations and probabilistic rule learning
Learning through experience!
domain model
goal planning
system designer
model updates
Change Plans
Plan Request
P1
P2
Change Actions
inference
execution traces
Status
Backbone interpreter
log
ICSE 2013

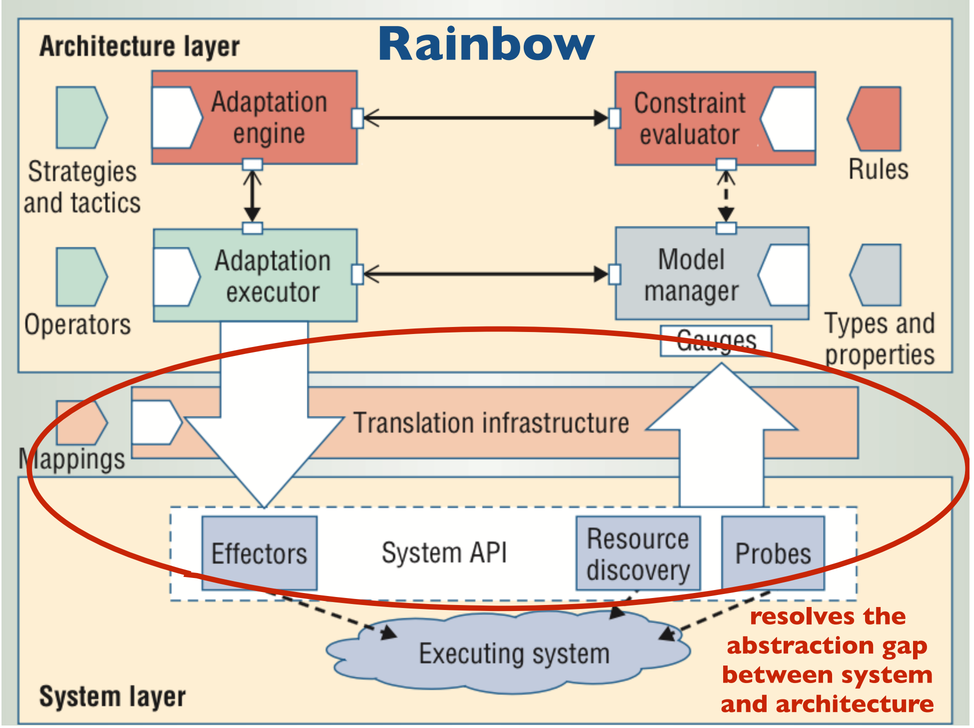

elaborate the three layer architecture
Goal Management
G
G'
G"
Change Plans
Plan Request
Change Management
P1
P2
Change Actions
Status
Component Control
C1
C2
Goal Model (System state + System Goals + Environment Assumptions)
Inference
log
Knowledge Repository

our current vision

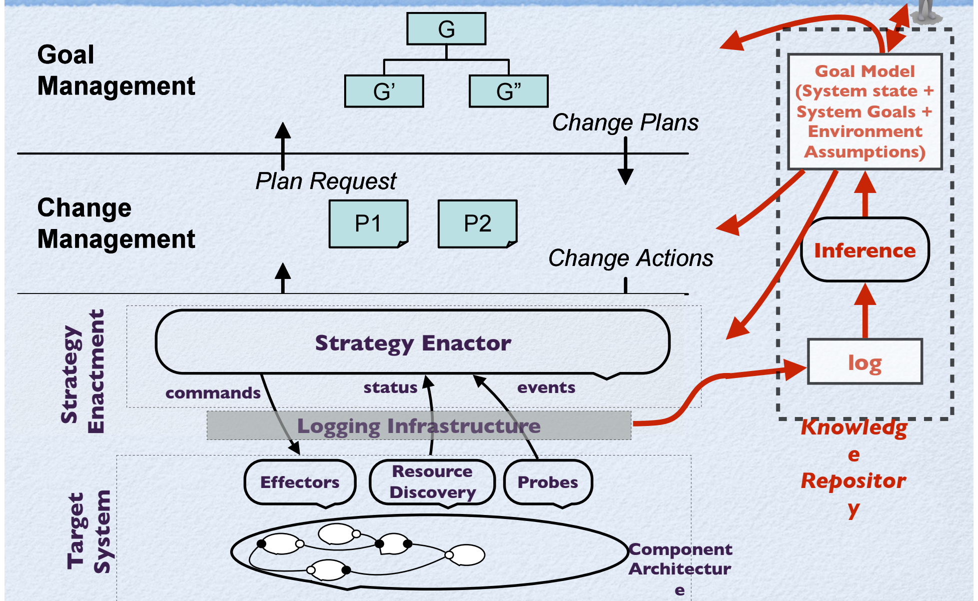
Provide a reference architecture which …

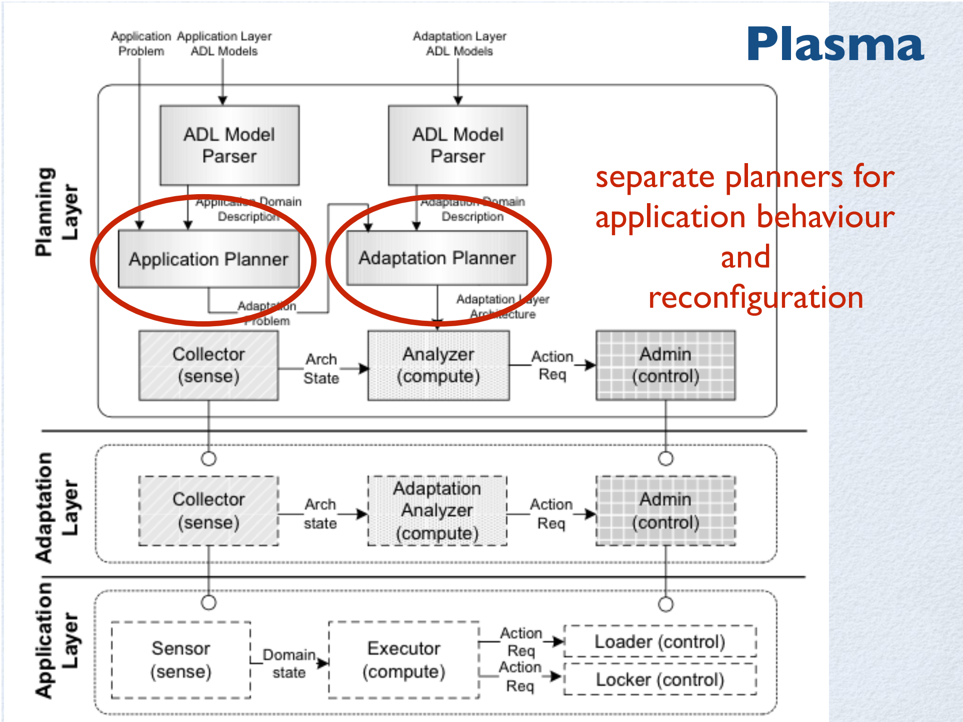
- accommodates specific research aspects more clearly
- facilitates comparison of specific approaches
- provides a pick-and-mix (plug-and-play) architecture

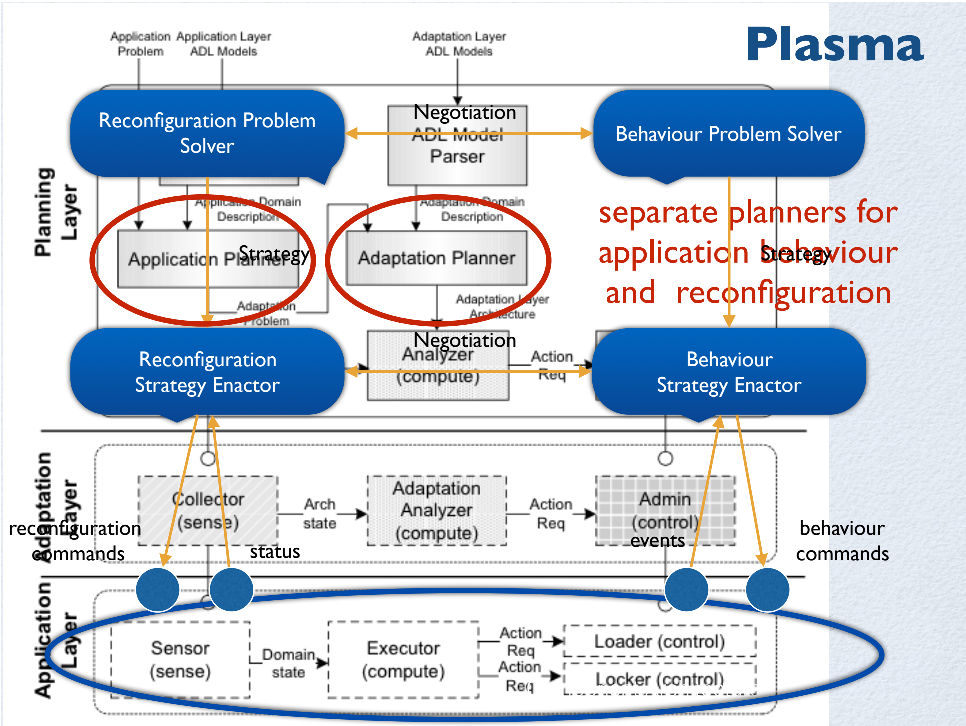
… a playground for adaptive engineers!

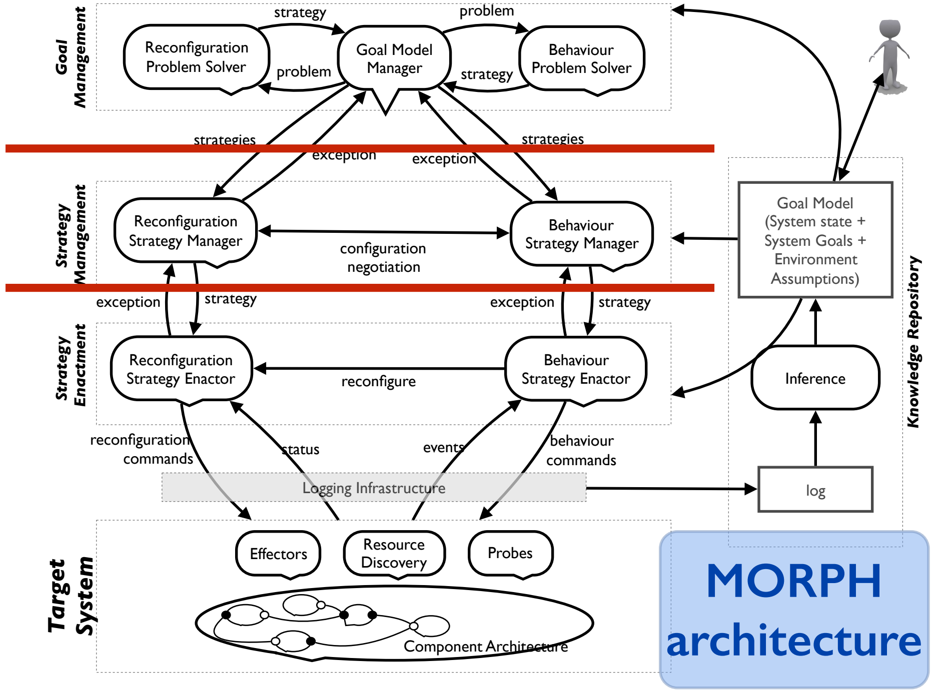


elaborating the three layer architecture



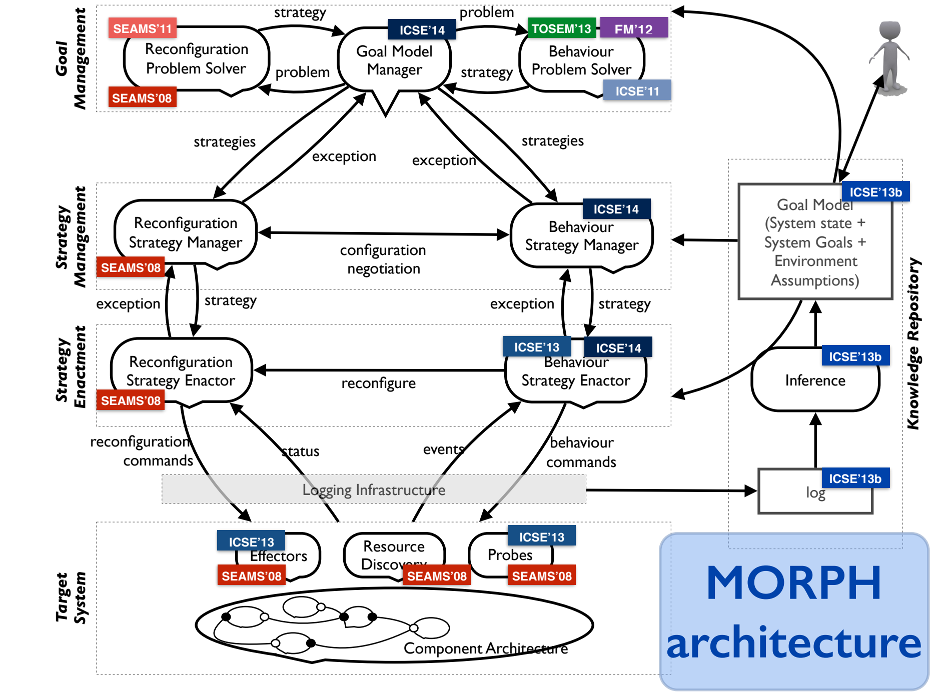






Vision: architectural reference model

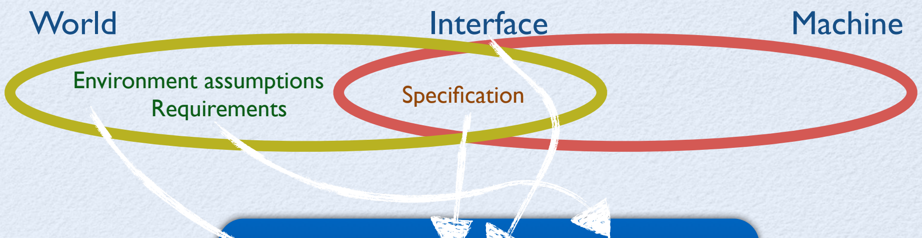
- identify and accommodate specific research concerns,
- facilitate comparisons between approaches, and
- provide a framework for potential implementations (plug-and-play)



challenging case studies

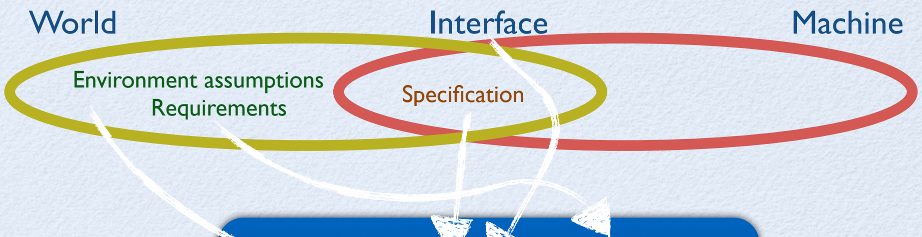
- evaluation
- validation
- comparison

Requirements Engineering



$$E, x_i \models R$$

Requirements@runtime



$$E, x_i \models R$$

$$E \,||\, x_i \models G$$

the challenge of change

- **model revision in response to updates and change in the environment**
- **online Requirements Engineering in response to updates and changes in goals (RE@runtime)**

- automated support for diagnosis and repair using a combination of model checking and machine learning
- automated support for requirements elaboration and obstacle analysis

ASE 2008, ICSE 2009, ICSE 2012, CACM 2015

in conclusion ...

Adaptive and Self-Managed Systems

…. the challenges of change …

environment
goals
capabilities

…. to automate and run on-line what is currently off-line!

…. a sound foundation can be provided by an appropriate architecture.

AWASE

architecture provides an adaptive engineering playground!

Bliss