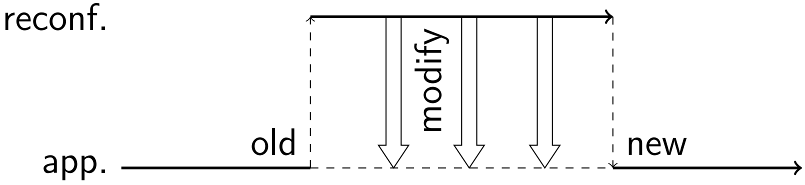
Acme vs PDDL: support for dynamic reconfiguration of software architectures

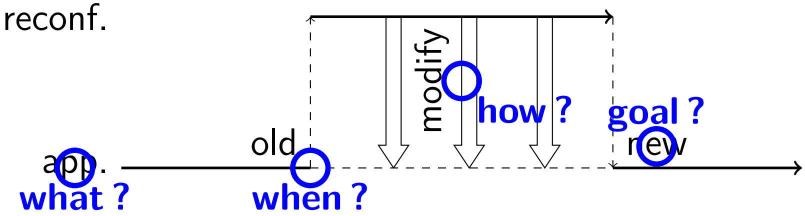
Jean-Eudes Méhus, Thais Batista & Jérémy Buisson
jean-eudes.mehus@st-cyr.terre-net.defense.gouv.fr
thais@dimap.ufrn.br
jeremy.buisson@st-cyr.terre-net.defense.gouv.fr

UNIVERSITÉ EUROPÉENNE DE BRETAGNE
Ecoles de Saint-Cyr Coëtquidan
DIMAp UFRN
UBS Université de Bretagne-Sud

Dynamic reconfiguration



Dynamic reconfiguration



Summary of contributions and directions

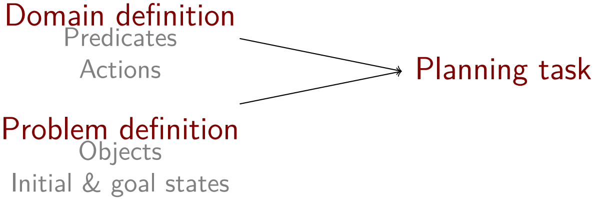
- Experiments with IA action planning
  - Improvements with respect to the state-of-the-art :
    - Account for constraints, styles & types
    - Verification of invariants
    - Systematic evaluation of International Planning Competition (IPC) planners
- Towards improved reconfiguration language

ACME, Armani & Plastik

- ACME [Garlan et al.(2010)]
  - Architecture description language
  - Components, connectors & attachments
  - Focused on the structure of the software architecture
  - Aimed as an interchange language
- Armani [Monroe(2001)]
  - Constraints over the architecture
  - Based on first-order predicate logic
- Plastik [Batista et al.(2005)]
  - Reconfigurations for ACME architectures
  - Triggering on Armani conditions
  - Primitive operations

PDDL & Action planning

- Action planning
  - Automatically find a sequence of actions that brings a system from an initial state to a goal state
- Planning Domain Definition Language [Ghallab et al.(1998)]
  - Based on first-order logic
  - Designed for the International Planning Competition (IPC)



Domain definition

Predicates

Actions

Problem definition

Objects

Initial & goal states

Planning task

PDDL & Dynamic reconf. – state-of-the-art

- [Arshad and Heimbigner(2005), Arshad et al.(2007)]
  - A PDDL domain for reconfiguration
- [André et al.(2010)]
  - PDDL as a pivot language
  - Planners may generate optimal reconf.
- [Ingstrup and Hansen(2009), Hansen and Ingstrup(2010)]
  - Planning for OSGi deployment
  - Using Alloy to plan & verify, but not using the same specifications as with PDDL
- [El Maghraoui et al.(2006)]
  - Planning Tivoli deployment

PDDL & Dynamic reconf. – sum. of state-of-the-art

- PDDL is a relevant option
- State-of-the-art planners provide good results
- Several PDDL domains for reconfiguration

- But...
  - Poor type support
  - No constraint, no style support
  - One PDDL domain per ADL / component model
  - Are we sure that the PDDL domains are correct ?
  - Only few planners tested

PDDL domain for reconfiguration

- **PDDL types :** what kinds of objects do we reify ?
- **Predicates :** how do we represent an architecture ?
- **Actions :** what operations do we define ?

Types

- Kinds of (reified) elements

	Type	Instance
System		×
Component	×	×
Connector	×	×
Port	×	×
Role	×	×

Predicates

- Bindings & containment relations
  - Component – port, connector – role
  - System – component, system – connector
  - Instance – type
- Existence
  - Because the PDDL world is closed

Design choices

- Negative predicates vs negation+quantification
  - E.g., unbound port, unbound role
- Each component has its own unique type

Actions

- Example : attach a port `p` of a component `c` and a role `r` of a connector `co`
  - **Preconditions :**
    - `c` exists ; `co` exists
    - `c` has port `p` ; `co` has role `r`
    - `p` on `c` is not bound ; `r` on `co` is not bound
  - **Positive effects :**
    - `p` on `c` is bound to `r` on `co`
  - **Negative effects :**
    - `p` on `c` is not bound
    - `r` on `co` is not bound

Invariants & constraints

- We can check statically that the actions preserve some invariants
- Example : one port is bound to at most one role
  - Check each action
  - E.g., the `attach` action binds only unbound ports

Invariants & constraints

- Some invariants cannot be checked statically
- Example : the client-server style

- Either we design a specific domain for the reconfiguration of client-server applications
- Or the domain is general, but the constraint is not statically enforced
  - Planning time verification : PDDL state trajectory constraints
  - Temporal modal operators : constraints over what happens during reconfiguration

Some experimental results

Why systematic tests of IPC planners ?

- International Planning Competition (IPC) promotes fastest planning time for ≃100-actions plans
- No IPC planner implements the whole PDDL

- Summary of experiments

IPC planners	55
Successful (simplified problem)	17
Shortest plan	14
With derived predicates	1
With constraints	0

Lessons learned

- Reconfiguration
  - Architectural constraints can be enforced
  - How do we specify what we expect to be true during reconfiguration ?
  - How do we reconfigure architectural constraints ?
- Planning
  - PDDL seems expressive enough
    - Some temporal operators shall be missing
    - Semantics of PDDL ?
  - No state-of-the-art off-the-shelf planner implements the needed PDDL fragments
  - The planning community focuses on other issues

Future directions

- Towards next reconfiguration language
  - Reconfiguration « style »
  - Operations to reconfigure types & constraints
- Further inspection of existing planners
  - More precise characterization of implemented PDDL features
  - Compilation strategies of advanced PDDL features to core PDDL
  - Impact on planning time

Acme vs PDDL: support for dynamic reconfiguration of software architectures

Merci de votre attention !
Avez-vous des question ?

Françoise André, Erwan Daubert, Grégory Nain, Brice Morin, and Olivier Barais.
F4Plan : an approach to build efficient adaptation plans.
In **7th International ICST Conference on Mobile and Ubiquitous Systems**, Sydney, Australia, December 2010.

Naveed Arshad and Dennis Heimbigner.
A comparison of planning based models for component reconfiguration.
Technical Report CU-CS-995-05, University of Colorado, Boulder, Colorado, USA, 2005.

Naveed Arshad, Dennis Heimbigner, and Alexander Wolf.