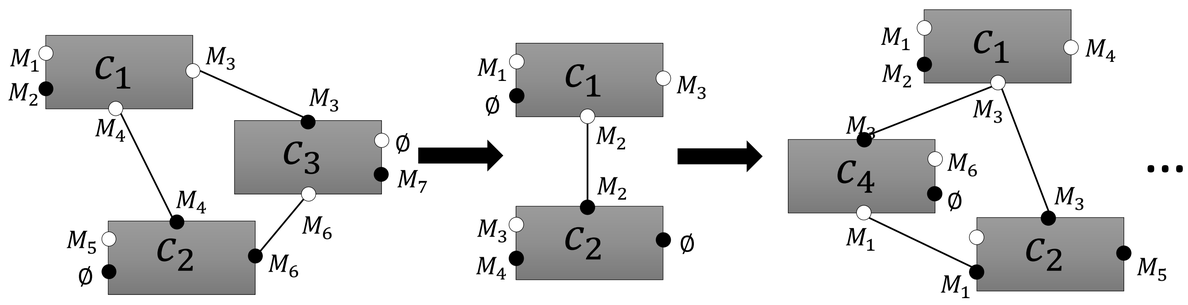
Software and Systems Engineering
Fakultät für Informatik
Technische Universität München

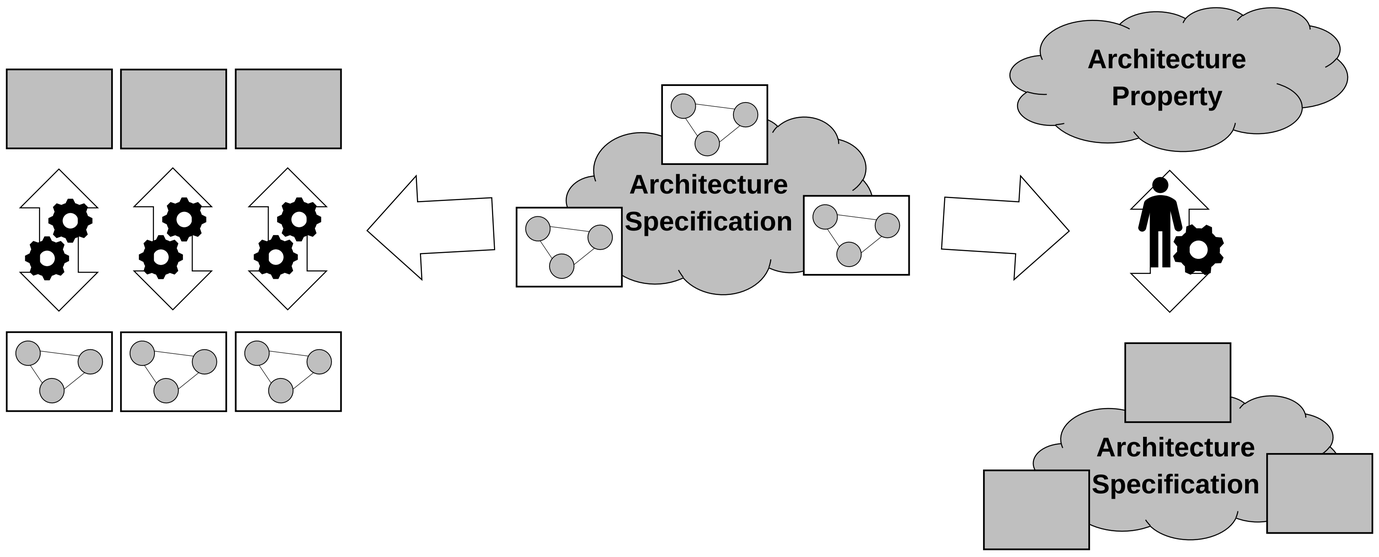
Verifying Dynamic Architectures using Model Checking and Interactive Theorem Proving

Verification of Dynamic Architectures

- Components can appear and disappear over time
- Connections between components may change over time
- Unbounded number of components
- Evolving state space



Verification based on Model Checking and Interactive Theorem Proving



Verification of Component Types using Model Checking

- Restricted state space
- Frequent implementation changes

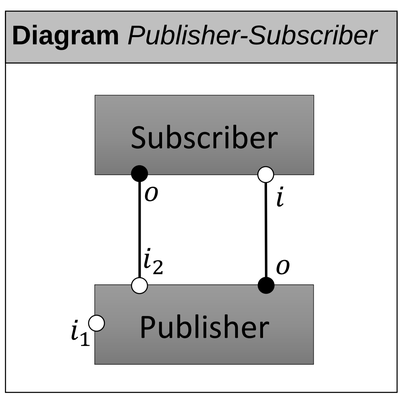
Verification of Component Integration using theorem proving

- Axiomatic reasoning
- Robust verification results

Specification of Dynamic Architectures in FACTum

Example

Diagram *Publisher-Subscriber*



BSpec *Publisher*
var m
$\square(i_1 = m \Rightarrow \bigcirc(o = m))$

Interfaces
- Abstract Datatypes
- Input, Output Ports
- Component Parameters

Component Behavior
- Linear Temporal Logics
- Ports from interfaces as free Variables
- Functions from Datatypes

Architecture Diagrams
- Graphical Notation
- Interfaces
- Annotations
  - Connection
  - Activation

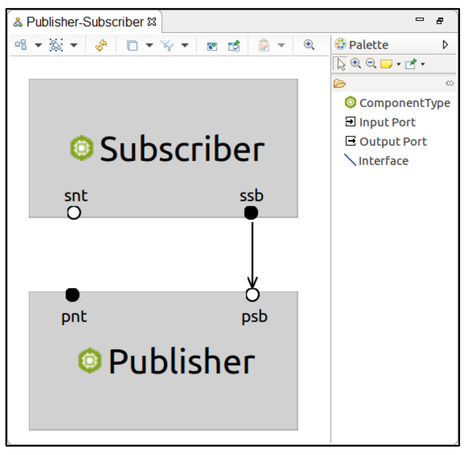
Architectural Assertions
- Linear Temporal Logics
- Component Variables
  - Flexible: $flex\ c\colon CT$
  - Rigid: $rig\ c\colon CT$
- Architecture Predicates
  - Activation: $\{\!|c|\!\}$
  - Connection: $c_1.i \rightsquigarrow c_2.o$

Example

DTSpec *subscription(id, evt)*
generated by $sub\ id\ \wp(evt)$ $unsub\ id\ \wp(evt)$

ASpec *Publisher-Subscriber*
var $p, p'\colon Publisher, s\colon Subscriber$ E: $\wp(evt), e\colon evt$
$\square(\{\!|p|\!\} \wedge \forall p'.\{\!|p'|\!\} \Rightarrow p' = p)$ $\square\Big(p.i_2 = sub(s,E)$ $\Rightarrow \big((s.i \rightsquigarrow p.o)\ \boldsymbol{W}\ p.i_2 = unsub(s,E)\big)\Big)$

FACTum Studio




```
pattern.pmodel
DTSpec {
    DT Event (
        Sort EVT, ESET, MSG
        Operation evt: MSG => EVT
        Predicate in: ESET, EVT
    ),
    DT Subscription (
        Sort SID, SBS
        Operation sub: SID,Event.EVT => SBS,
                  unsub: SID, Event.EVT => SBS
    )
}
```

```
pattern.pmodel
//Container for Archite
ArchSpec {
    //Component Variable
    rig p : Publisher,
    flex q: Publisher,
    rig s : Subscriber

    //Data Type Variables
    rig e: Event.EVT

    //Configuration Trace Assertions
    act_pb: G(cAct(p) ∧ ∀q. cAct(q) ⇒ eq(q,p))
    conn_sb: G(cAct(p) ∧ val(p.psb, Subscription.sub(s.id,e))
        ⇒((cAct(s) ∧ cAct(p) ∧ e = Event.evt(p.pnt) ⇒ conn(p.pnt,s.snt))
            W val(p.psb, Subscription.unsub(s.id,e))))
}
```

```
pattern.pmodel
//Container for Architectural Guarantees
ArchGuarantee {
    //Component Variables
    flex p : Publisher,
    rig  s : Subscriber

    //Data Type Variables
    rig e: Event.EVT

    //Configuration Trace Assertions
    delivery: G(cAct(s) ∧ val(s.ssb, Subscription.sub(s.id,e))
        ⇒ ((cAct(s) ∧ e = Event.evt(p.pnt) ⇒ val(s.snt, p.pnt) W cAct(p) ∧ val(p.psb, Subscription.unsub(s.id,e)))))
}
```

Export to Isabelle/HOL

Interface

locale `dynamic_component` $(tCMP,\ active)$

Model
- $proj$: component projection
- $nAct$: number of activations
- $lNAct$: least deactivation
- $nxtAct$: next activation
- $lActive$: last activation
- $latestAct$: latest activation
- Mapping time points
  - $cnf2bhv$: from cnf.t. to bhv.t.
  - $bhv2cnf$: from bhv.t. to cnf.t.
- Lemmata for each definition

`Configuration_Traces.thy`

extends

Calculus
- Behavior trace assertions
  - ass: behavior assertions
  - nxt: next
  - evt: eventually
  - $glob$: globally
  - $until$: until
  - $wuntil$: weak until
- evaluation function
- Lemmata for calculus

`Dynamic_Architecture_Calculus.thy`

Co-inductive Lists

`Coinductive_List.thy`

Export to NuSMV

- Component types specify assertions for component behavior
- Component behavior is modeled using state machines
- Automatic verification of components against their types

Calculus to support verification

$\mathrm{NxtE_{a1}}$

$$\frac{(t,t',n)\models^{\delta}_{c}\text{“}\bigcirc\gamma\text{”} \quad n\le n' \quad \exists!\, n\le i<n'\colon |c|_{t(i)}}{(t,t',n')\models^{\delta}_{c}\text{“}\gamma\text{”}} \ \exists i>c\xrightarrow{n}t\colon |c|_{t(i)}$$

Diego Marmsoler
Blozmannstr. 3
Garching bei München
+49 89 289 - 17882
diego.marmsoler@tum.de
https://marmsoler.com