EASSy 2015

Modularity for Uncertainty in Adaptive Software Systems

Naoyasu Ubayashi

Kyushu University, Japan

September 9, 2015

九州大学
KYUSHU UNIVERSITY

Outline

- Motivation
- Modularity for Uncertainty
- Towards Uncertainty-Aware Self-Adaptive Systems
- Summary

Motivation

What is Uncertainty?

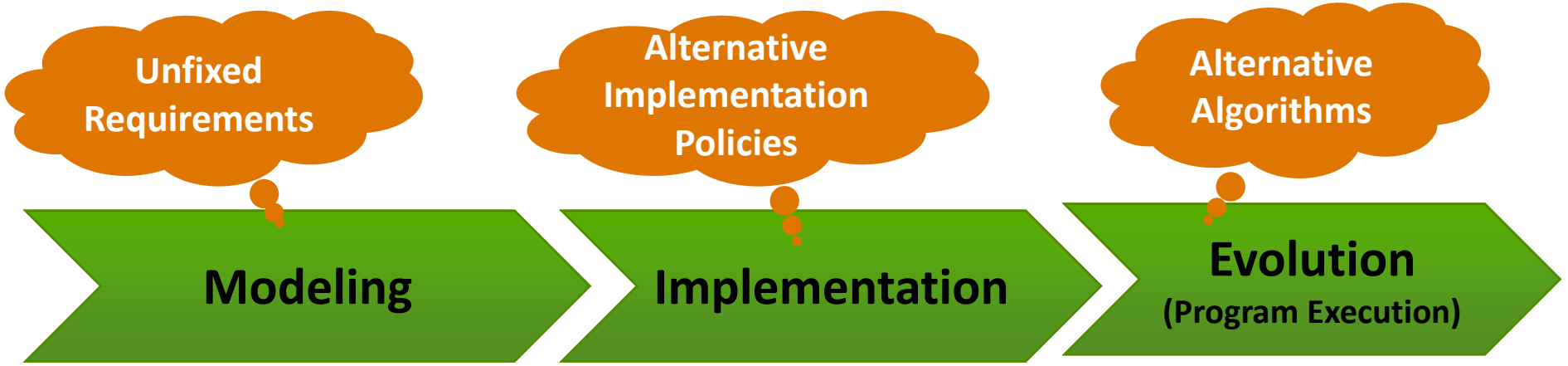
- The most used definitions of uncertainty simply distinguish between natural variability of physical processes (i.e., ***aleatory or stochastic uncertainty***) and the uncertainties in knowledge of these processes (i.e., ***epistemic or state-of-knowledge uncertainty***).

- For proposing such a taxonomy for uncertainties in computer systems models, we base on a general definition of uncertainty in modeling given in [2] as: "***any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system***".

[1] Diego Perez-Palacin, Raffaela Mirandola: Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation, ICPE 2014.
[2] W. Walker, P. Harremo´ Ss, J. Romans, J. van der Sluus, M. van Asselt, P. Janssen, and M. Krauss: Defining uncertainty. a conceptual basis for uncertainty management in model-based decision support, Integrated Assessment, 4(1):5–17, 2003.

Uncertainty in Software Development

- Requirements or design models tend to contain **uncertainty**, because all of the concerns cannot be captured at the early phase.
- **Uncertainty** can appear in all aspects of software development.

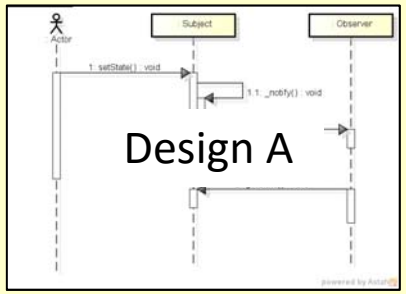


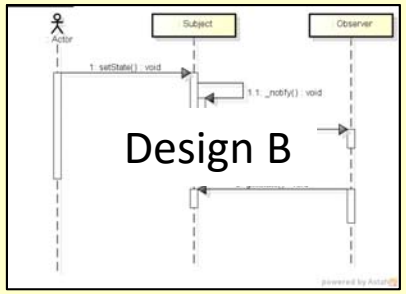
Two Types of Uncertainty

Known Unknowns

- Known what is unknown

e.g., Design Alternatives





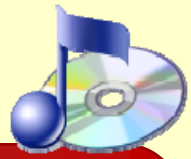
Which should be selected ?

Unknown Unknowns

- Unknown what is unknown

e.g., Users' Preference

recommend



Unknown Unknowns

Users

Accepted!

Unaccepted...

Sebastian, E., and David S. R.: Known Unknowns: Testing in the Presence of Uncertainty, 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering(FSE 2014), 2014.

Problem: Ad-hoc Approach to Uncertainty

```
1  public void setState(State s){
2      this.state = s;
3  //  _notify();
4      for(Observer o : this.observers)
5          o.update();
6  }
7  /** temporary removed
8  public void _notify(){
9      for(Observer o : this.observers)
10         o.update();
11 }
12 */
```

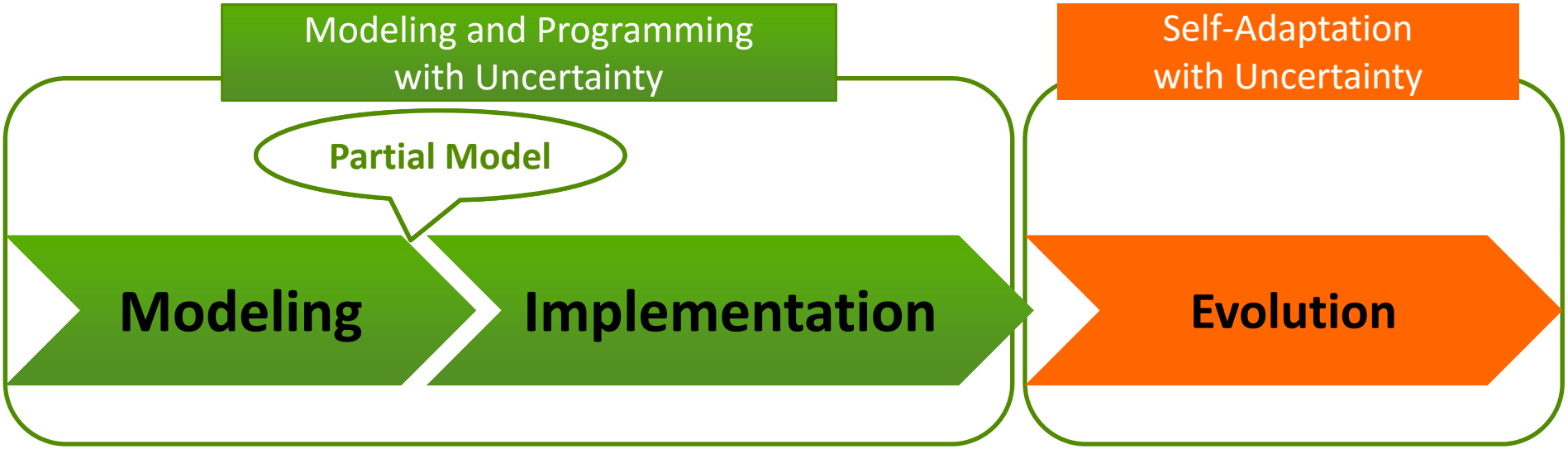
e.g., Approach with using comments

Problem

- Invade Separation of Concerns
- Uncertain concerns are not separated!
- The exploratory modification process may be repeated again and again until all uncertain concerns are fixed.

Uncertainty is not dealt with properly in current programming languages.

Our Research Goal



Current Research

Next Step

Can our idea deal with dynamic evolution?

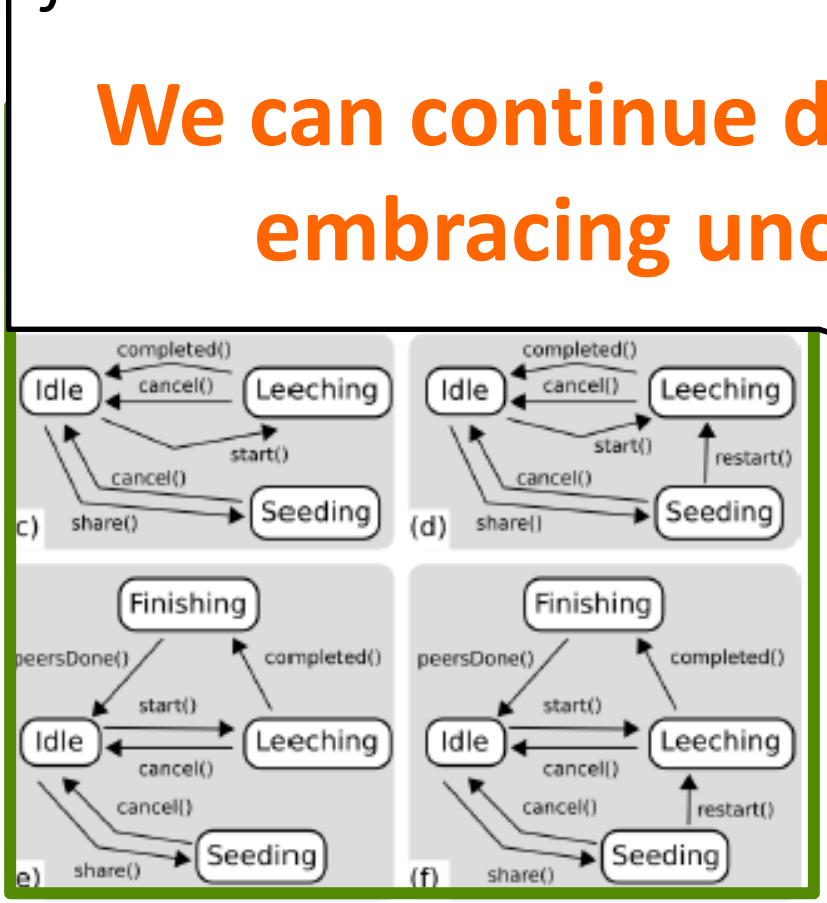
Modularity for Uncertainty

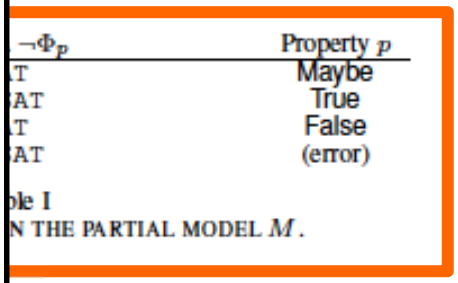
Modeling and Programming
for
Living with Uncertainty

Partial Model: Modeling with Uncertainty

If True…

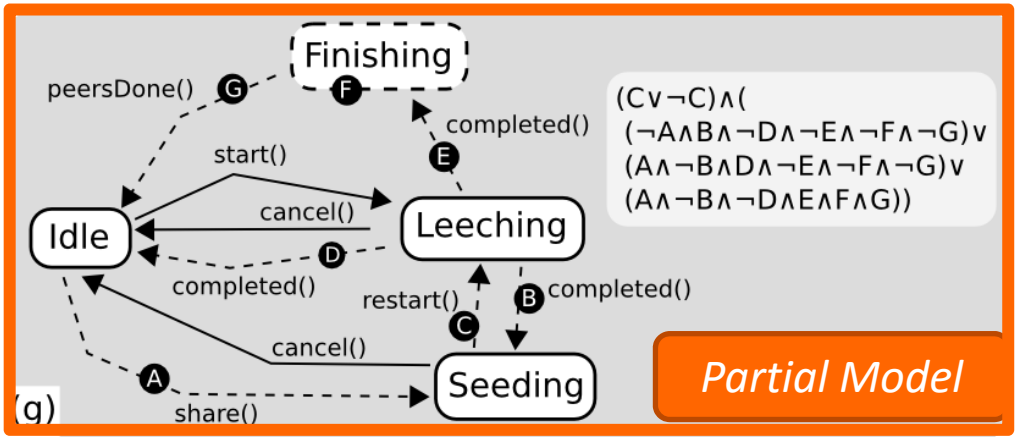
We can continue development embracing uncertainty





SAT Solver

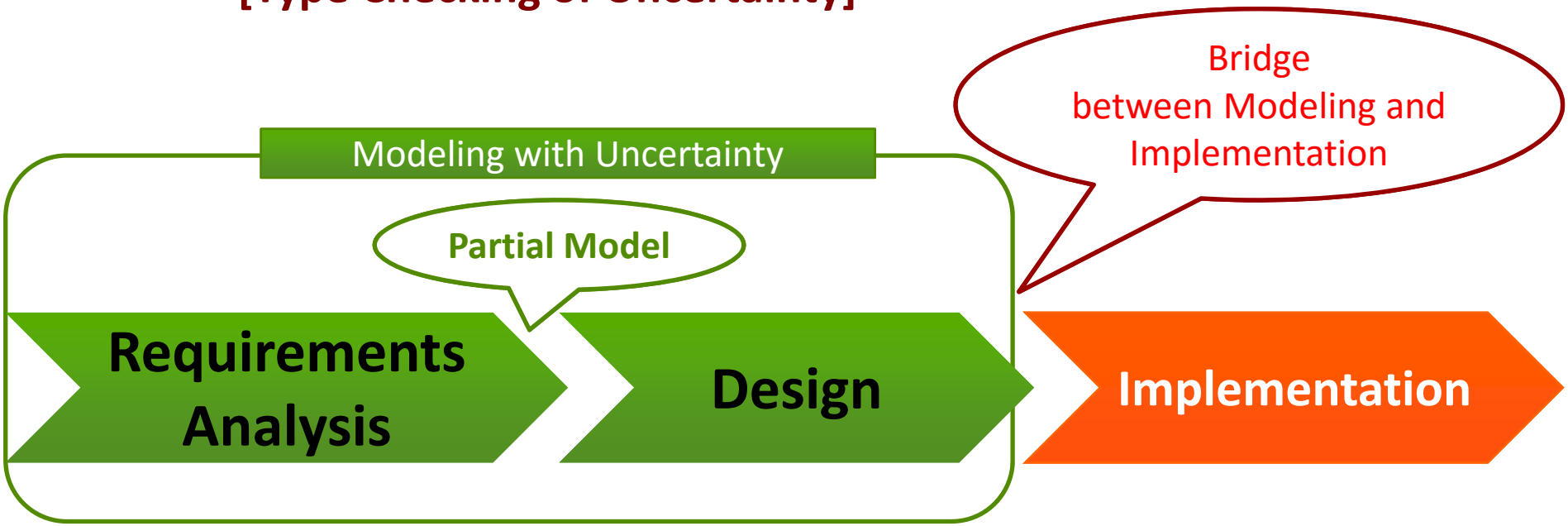
Check a Property



Famelis, M., Salay, R., Chechik, M.: Partial Models: Towards Modeling and Reasoning with Uncertainty,34th International Conference on Software Engineering (ICSE 2012), 2012.

Modular Programming with Uncertainty

- Idea 1: Uncertainty as Pluggable Interface
  [Expression of Uncertainty]
- Idea 2: Modular Reasoning based on Partial Model
  [Type Checking of Uncertainty]



Two Types of Uncertainty

Uncertain whether used or not

```
1  public void notify(){
2      for(Observer observer : observers){
3          observer.update();
4      }
5  }
6
7
8
9
10 }
```

Optional

e.g.,
Is `notify()` really used?

Uncertain which is used

```
public void BubbleSort(){
                . . .
}
public
}
```

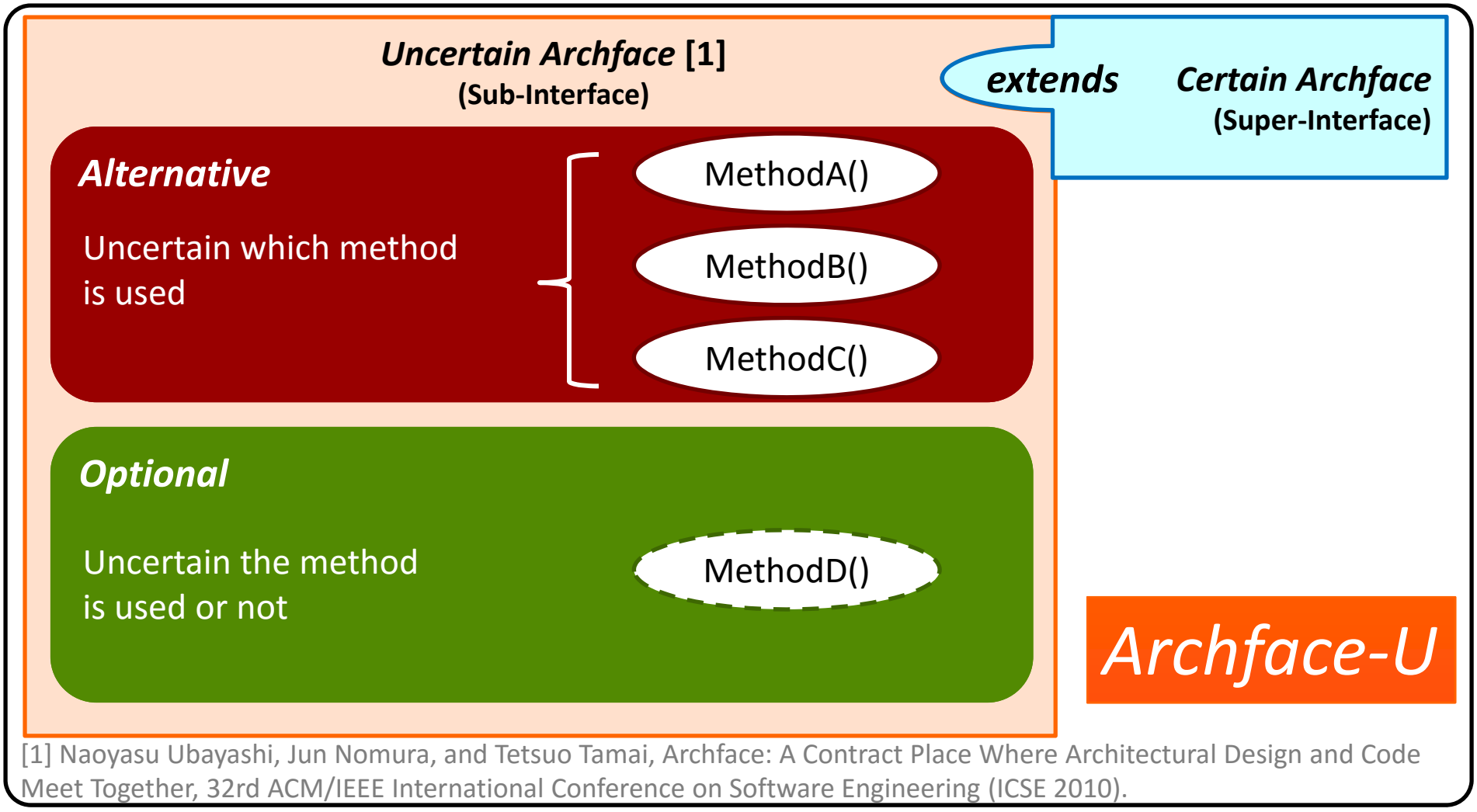
Alternative

e.g.,
Which sort algorithm should be used?

Requirements Analysis → **Design** → **Implementation**

Idea 1

Uncertainty as Pluggable Interface



[1] Naoyasu Ubayashi, Jun Nomura, and Tetsuo Tamai, Archface: A Contract Place Where Architectural Design and Code Meet Together, 32rd ACM/IEEE International Conference on Software Engineering (ICSE 2010).

Archface-U : Component Interface

Java Interface based

Certain Archface

```
interface component Subject {
    public void addObserver();
    public void setState();
}
```

{ } *Alternative*
[ ] *Optional*

Uncertain Archface

```
interface uncertain uSubject extends Subject {
  public void {removeObserver(), deleteObserver()};
  [public void notify();]
}
```

Archface-U : Connector Interface

Finite State Process (FSP) [2] *based*

Certain Archface

```
interface connector cObserverPattern(cSubject, cObserver)
{
  cSubject = (cSubject.setState->cObserver.update
                        ->cSubject.getState->cSubject);
}
```

Uncertain Archface

```
interface connector uObserverPattern extends cObserverPattern
(cSubject, cObserver) {
  cSubject = (cSubject.setState-> [uSubject.notify]
               ->cObserver.update->cSubject.getState->cSubject);
}
```

[2] Magee, J. and Kramer, J.: State Models and Java Programs (1999).

Type Check with Archface-U

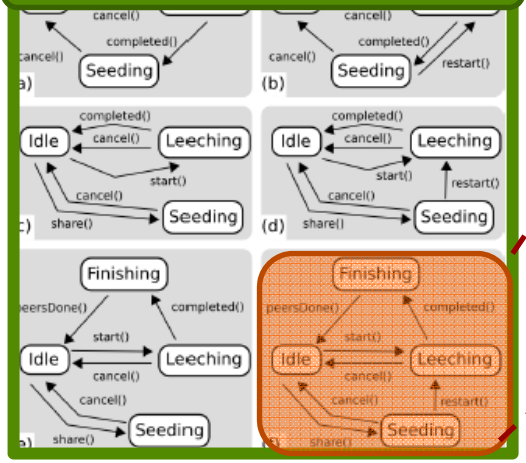
Simulation?

Type Check

Archface-U

Generate

Alternative Models




```
class MyNode{
        void cancel(){
                …
        }
        void completed(){
                …
        }
}
```

Java Code

Type Check is OK !

Tool for Type Check : *iArch-U*

Eclipse Plugin

Java - ObserverPattern/arch/observer.arch - Eclipse Platform

Java Code

Main.java

```
1  public class Main {
2
3      public static void main(String[] args) {
4          Subject subject = new Subject();
5          Observer observer = new Observer(subject);
6          subject.addObserver(observer);
7
8          subject.setState(State.START);
9
10         subject.setState(State.END);
11     }
12 }
```

Archface-U

Observer.java | observer.arch

```
1  interface component Subject{
2      void setState (State state);
3      State getState ();
4      void addObserver  (Observer observer);
5      void removeObserver  (Observer observer);
6  }
7
8  interface component Observer{
9      void update ();
10 }
11
12 uncertain component uSubject extends Subject{
13     [void _notify();]
14     {void deleteObserver(Observer observer),void removeObserver(Observer observer)};
15 }
16
17 Subject=(Subj .setState->Observer.update->Subject.getState->Subject);
   .update->Subject.getState->Observer);

   ObserverPattern{
   etState -> Observer.update -> Subject.getState -> Subject);
```

Type Check

Errors & Info.

Problems | Javadoc | Declaration | Console | Abstraction Ratio View

2 errors, 0 warnings, 10 others

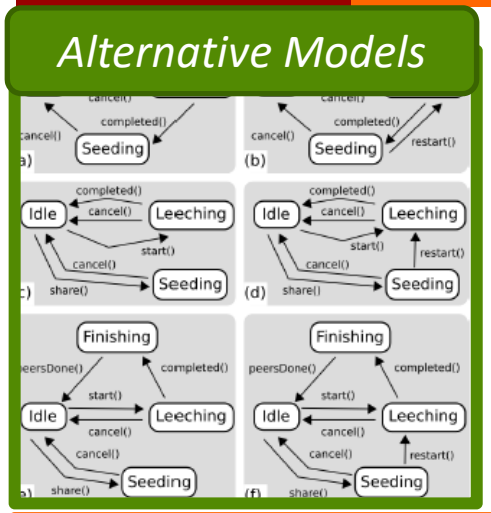
Description	Resource	Path	Location
Errors (2 items)			
_notify is not defined	Observer.java	/ObserverPattern/src	Subject
deleteObserver is not defined	Observer.java	/ObserverPattern/src	Subject
Infos (10 items)			
addObserver is defined	Observer.java	/ObserverPattern/src	Subject
Behavior : Observer : Observer.update ->Subject.getState	Observer.java	/ObserverPattern/src	Observe
Behavior : Subject : Observer.update ->Subject.getState	Observer.java	/ObserverPattern/src	Observe
Behavior : Subject : Subject.setState ->Observer.update	Subject.java	/ObserverPattern/src	Subject
deleteObserver is defined	Observer.java	/ObserverPattern/src	Subject
getState is defined	Observer.java	/ObserverPattern/src	Subject
Observer is defined	Observer.java	/ObserverPattern/src	Observe
setState is defined	Observer.java	/ObserverPattern/src	Subject
Subject is defined	Observer.java	/ObserverPattern/src	Subject
update is defined	Observer.java	/ObserverPattern/src	Observe

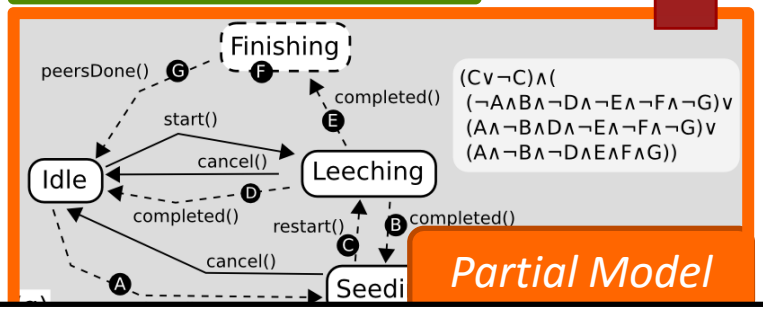
Archface View

Component Name	Uncertain Type	Impl
Subject	-	✔
setState	Certain	✔
getState	Certain	✔
addObserver	Certain	✔
removeObse...	Alternative	✔
_notify	Optional	×
deleteObserver	Alternative	×
Observer	-	✔
update	Certain	✔

Writable | Insert | 22 : 2

Bridging between Model and Code





Property Φ_P

(¬A∧B)∨...

OK!

Translate

Archface-U

OK!

Type Check

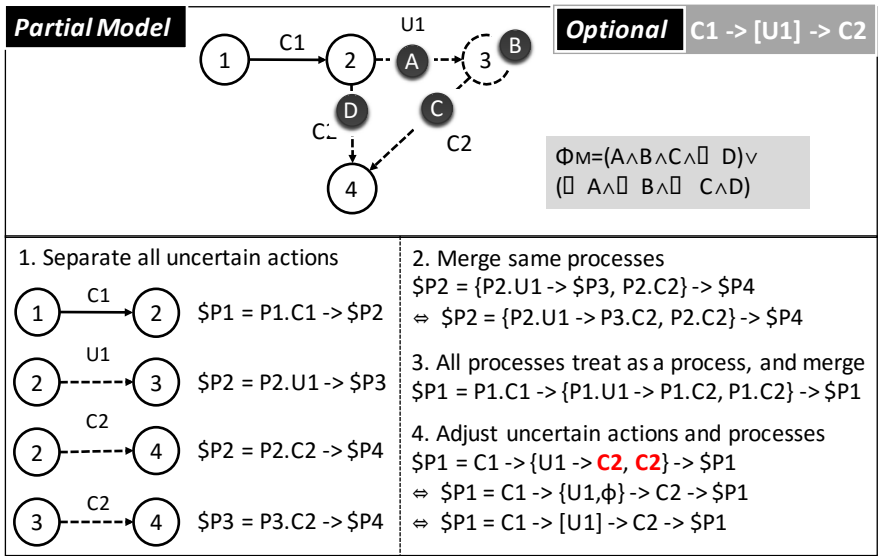
This Code
Satisfies the Property

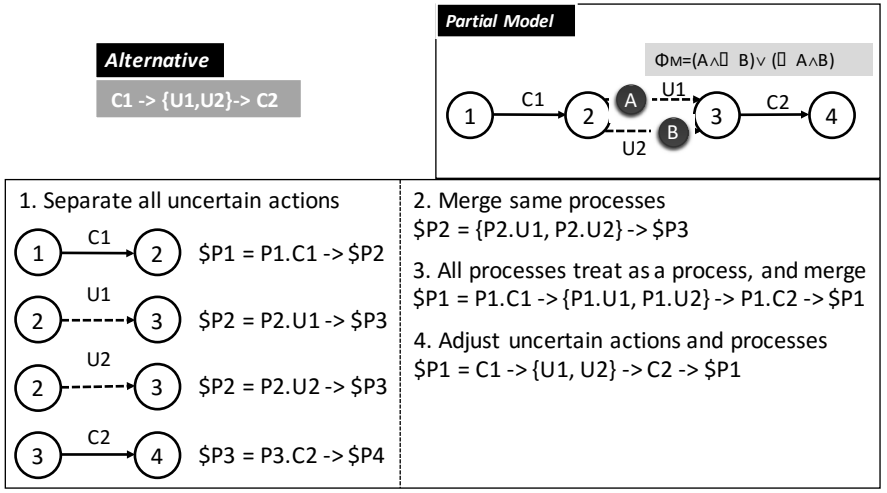
Java Code

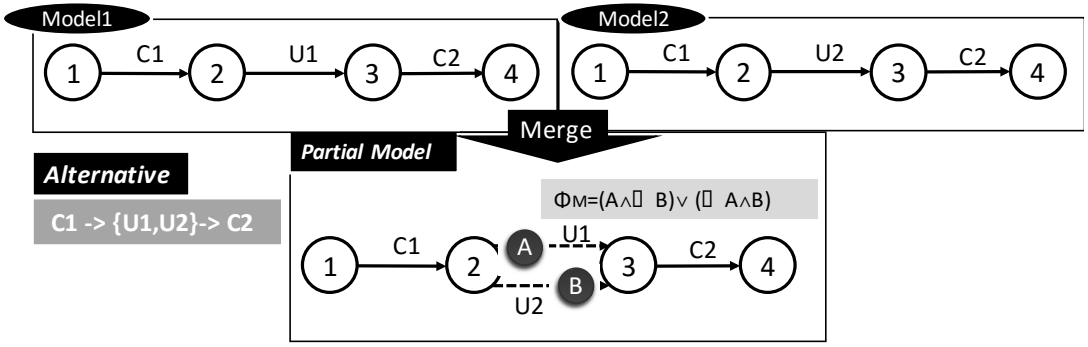
Partial Model & Property Checking

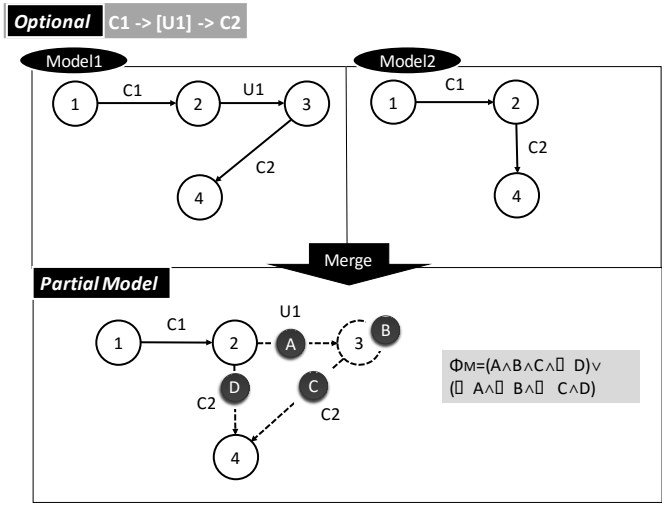
Requirement Analysis → **Design** → **Implementation**

Algorithms for Partial Model ⇔ Archface-U







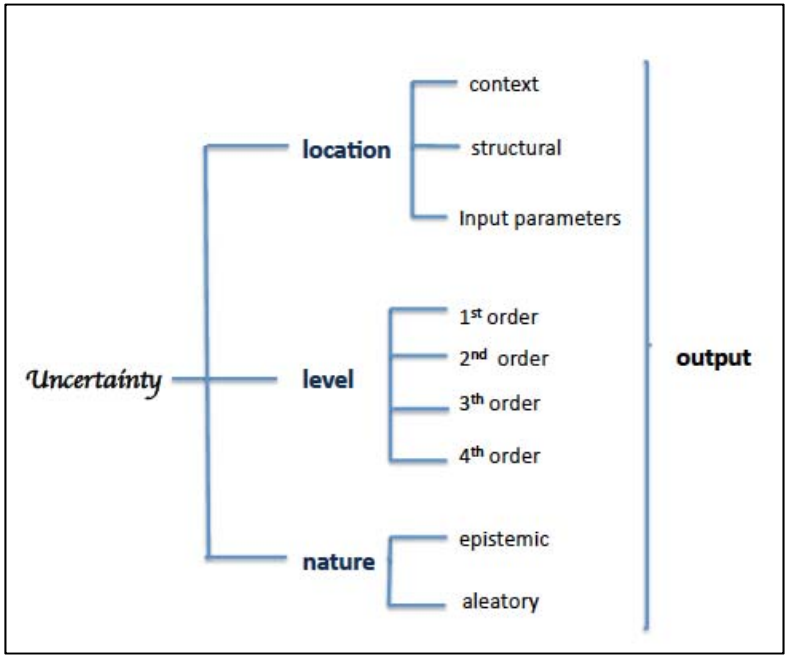


Towards Self-Adaptive Systems

Change of Program Execution
for
Living with Uncertainty

Uncertainty in Self-Adaptive Systems

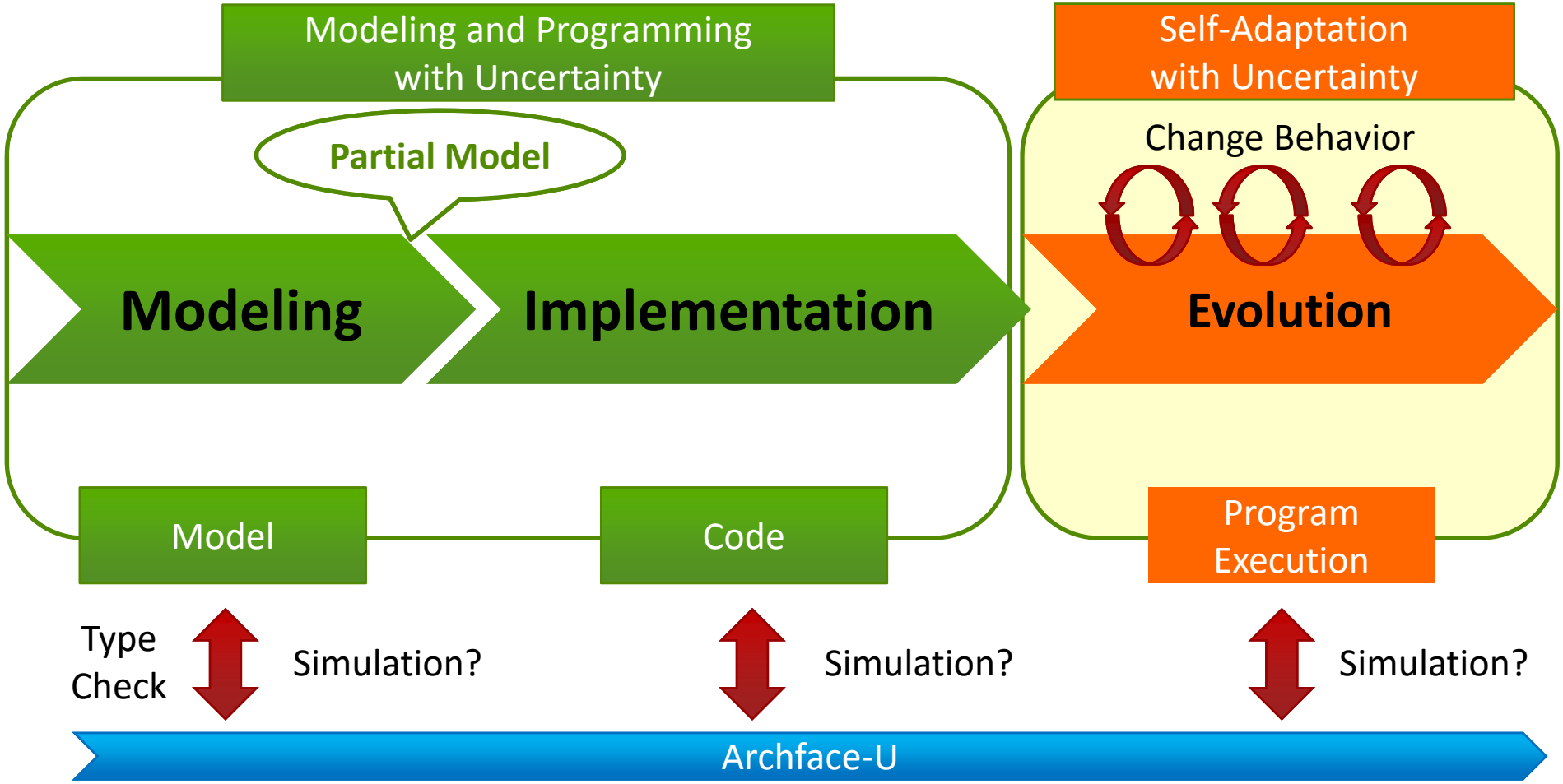
Source of Uncertainty	Classification	
	Location	Nature
Simplifying assumptions [12]	Structural/context	Epistemic
Model drift [12]	Structural	Epistemic
Noise in sensing [12]	Input parameter	Epistemic/ Aleatory
Future parameters value [12]	Input parameter	Epistemic
Human in the loop [12, 17]	Context	Epistemic/ Aleatory
Objectives [12]	Input parameter / context	Epistemic
Decentralization [12]	Context/structural	Epistemic
Execution context/ [12] Mobility [17]	context/ structural/ input parameters	Epistemic
Cyber-phisical system [12] [17]	Context/Structural Input parameter	Epistemic
Automatic learning [17]	Structural Input parameter	Epistemic Aleatory
Rapid evolution [17]	Structural Input parameter	Epistemic
Granularity of models [8]	Context/Structural	Epistemic
Different sources of information [8]	Input parameter	Epistemic/ Aleatory



Three-dimension classification

Diego Perez-Palacin, Raffaela Mirandola: Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation, ICPE 2014.

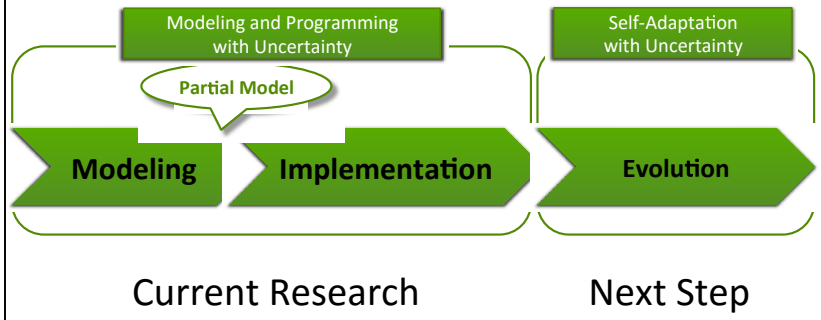
Can Our Approach Deal with Uncertainty in Self-Adaptive Systems?



Summary

7

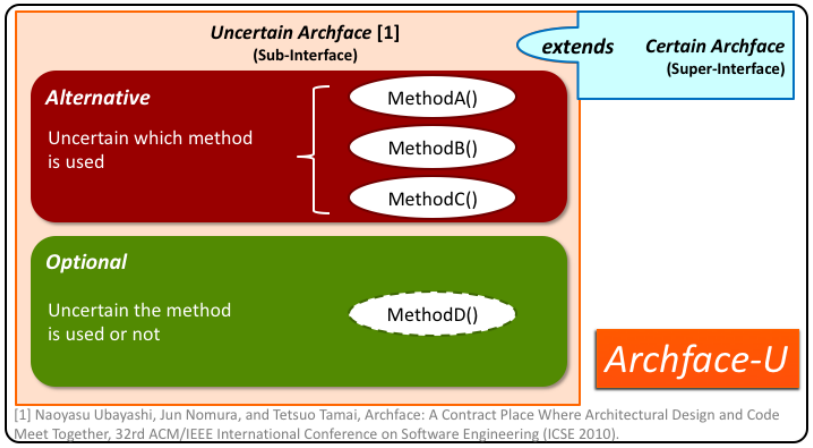
Our Research Goal



Idea 1

13

Uncertainty as Pluggable Interface

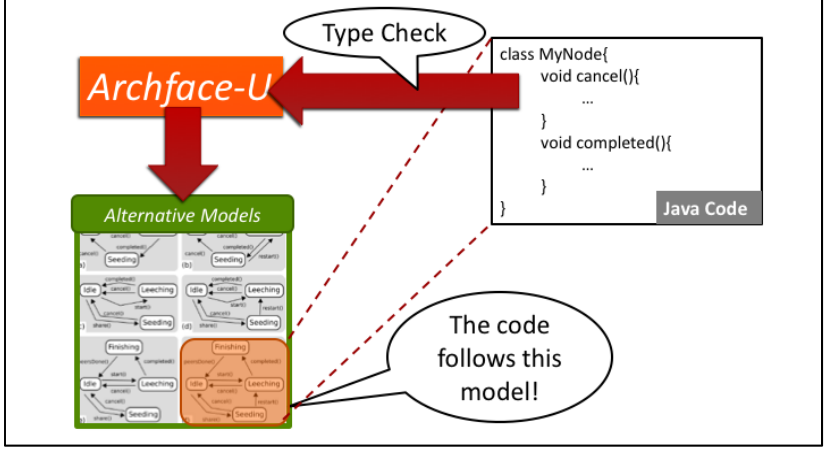


[1] Naoyasu Ubayashi, Jun Nomura, and Tetsuo Tamai, Archface: A Contract Place Where Architectural Design and Code Meet Together, 32rd ACM/IEEE International Conference on Software Engineering (ICSE 2010).

Idea 2

16

Type Check with Archface-U



23

Can Our Approach Deal with Uncertainty in Self-Adaptive Systems?