MiSE 2013 (ICSE Workshop)

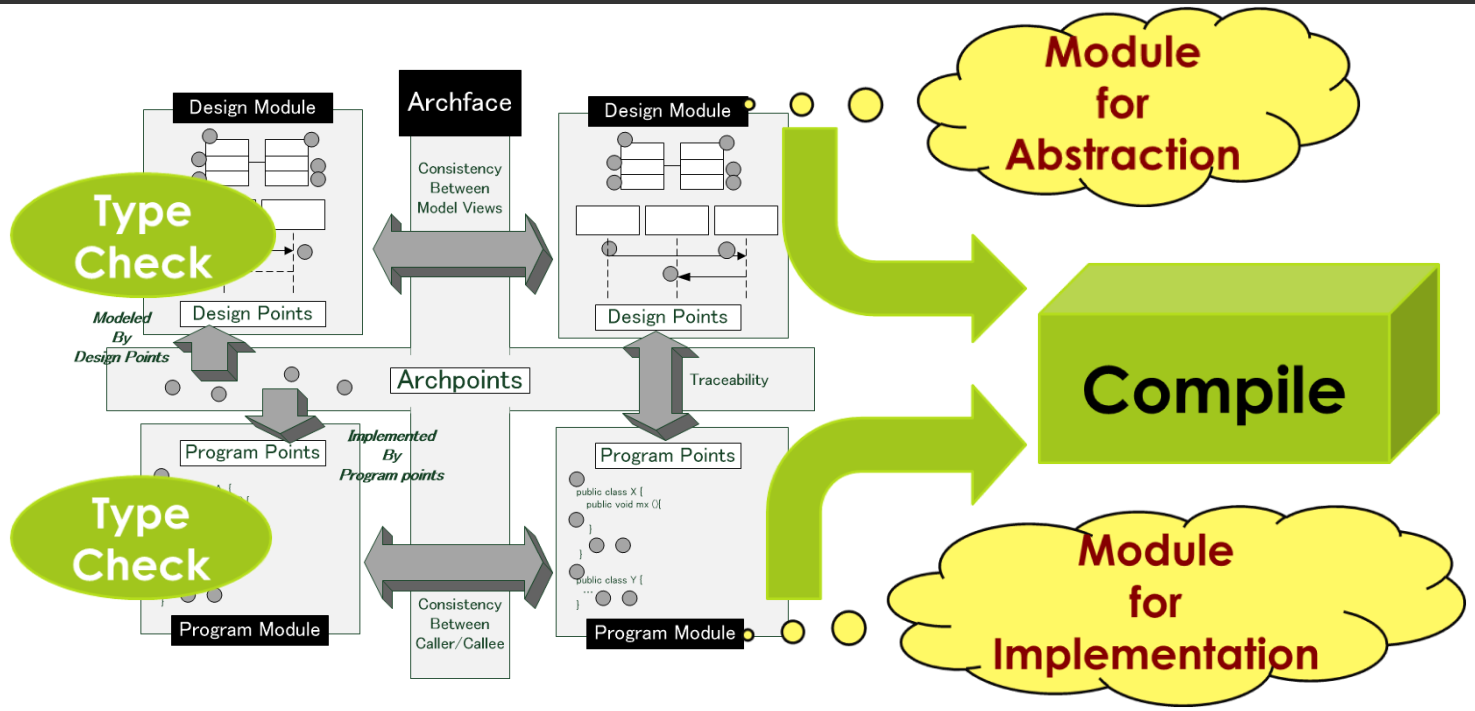
Design Module: A Modularity Vision Beyond Code

- Not only program code but also a design model is a module -

Naoyasu Ubayashi and Yasutaka Kamei
Kyushu University, Japan

May 18, 2013

Overview



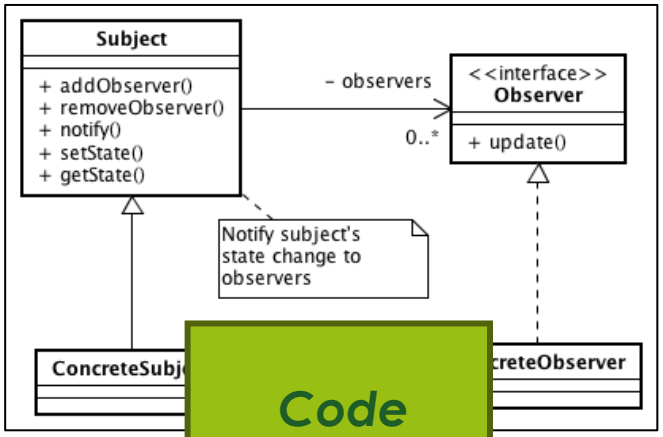
- We provide a novel idea that **treats a design model as a first-class software module.**
- A system cannot be compiled without design modules.
- A developer has to create and modify not only program modules but also design modules.

Outline

- Motivation
- Design Module: New Modularity Vision
- Integration of Design and Program Modules
- Conclusions

Motivation

MDD is fine, but …



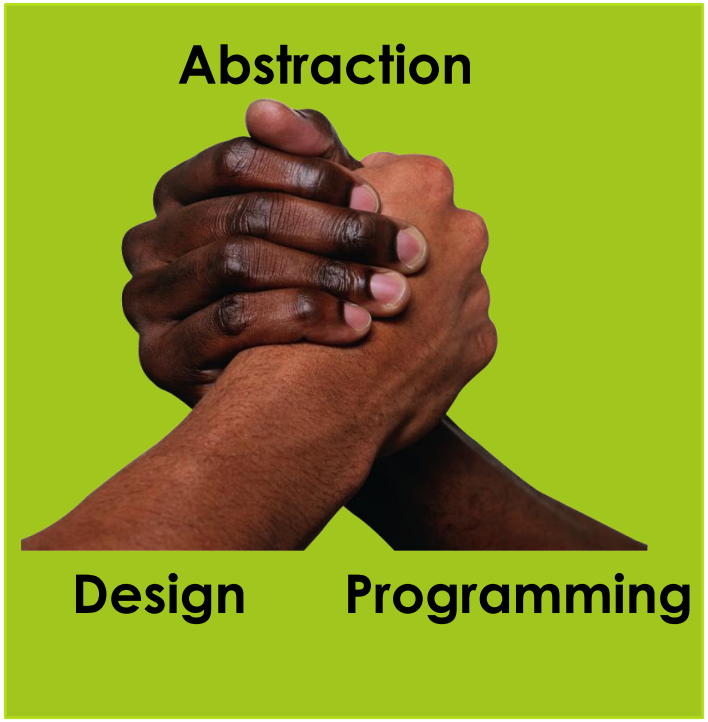
Design

Programming

Our Opinion

Programming should not disappear!

- Programming has not yet disappeared from most software development projects.
- **Both design activities and programming have their own roles.**
- **The key point is Abstraction!**



Abstraction, abstraction, …

- Why is it that some software engineers and computer scientists are able to produce clear, elegant designs and programs, while others cannot?
- Is it possible to improve these skills through education and training?
- Critical to these questions is the notion of **abstraction**.

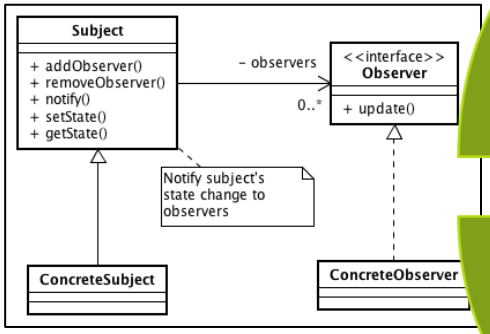
Kramer, J., Is Abstraction the Key to Computing? Communications of the ACM, Vol. 50, Issue 4, pp.36-42, 2007.

What is the purpose of design modeling?

Full code generation

Abstraction?

Detailed Design




```
public class Subject{      private Vector observers
= new Vector();
   private String state = "";            public void
addObserver(Observer o){ … }      public void
removeObserver(Observer o){ … }      public void
notify(){          Iterator i = observers.iterator();
while(i.hasNext() ){              Observers o =
(Observer)i.next();
o.update( this );          }     }
   public void setState(String s){ state = s; }
   public String getState() {return state; } }
```

Abstract Design

Skeleton generation or Hand coding

Traceability?

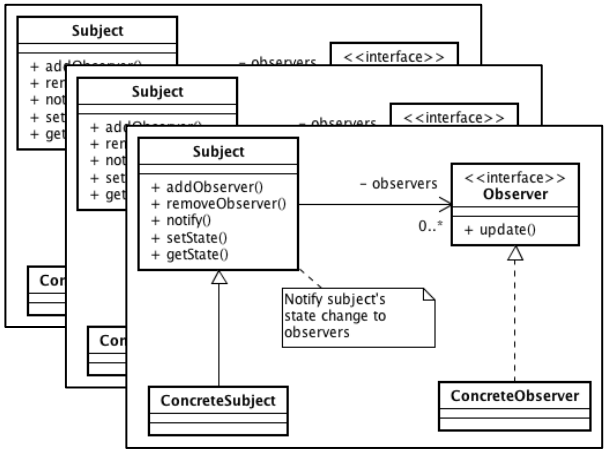
Abstraction and Modularity

- Abstraction is affected by a language mechanism, especially modularity.
- Although a program is composed of program modules, **design models are not regarded as software modules.**

Why?

Not only program code but also a design model is a module

Abstract Design



Code

```
public class Subject{      private Vector observers =
new Vector();
    private String state = "";              public void
addObserver(Observer o){ … }      public void
removeObserver(Observer o){ … }      public void
notify(){          Iterator i = observers.iterator();
while(i.hasNext() ){              Observers o =
(Observer)i.next();
o.update( this );          }     }
    public void setState(String s){ state = s; }
    public String getState() {return state; } }
```

Role Abstraction

Compile

Role Implementation

Our Approach

- A UML design model such as a class diagram and a sequence diagram is regarded as a design module.

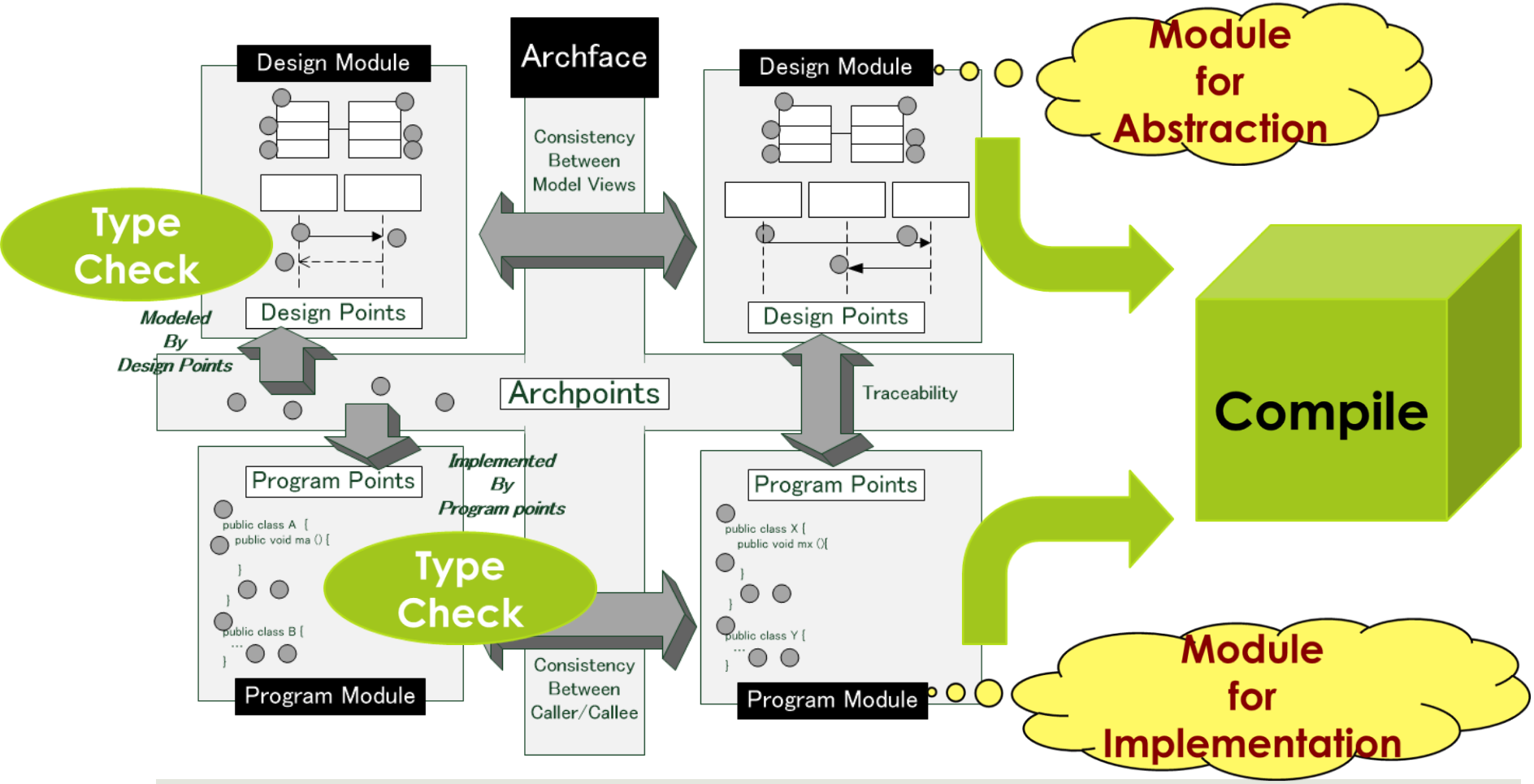
But, How ?

Its Answer is *Archface*!

Ubayashi, N., Nomura, J., and Tamai, T., Archface: A Contract Place Where Architectural Design and Code Meet Together, ICSE 2010.

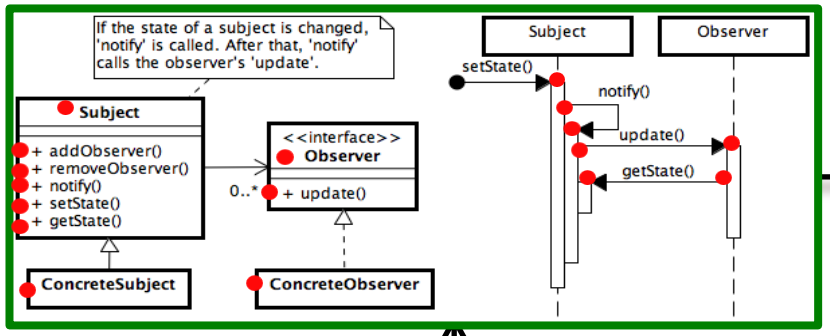
Design Module:
New Modularity Vision

Archface-Centric Modularity



Archface = Design + Program Interface

Abstract Design



Contract between design and code

A set of design points

Archface
(Exposure of archpoints)

```
ublic class Subject{      private
Vector observers = new Vector();
   private String state = ""
…
}
```

Code

A set of program points

Abstraction level can be defined by selecting archpoints!

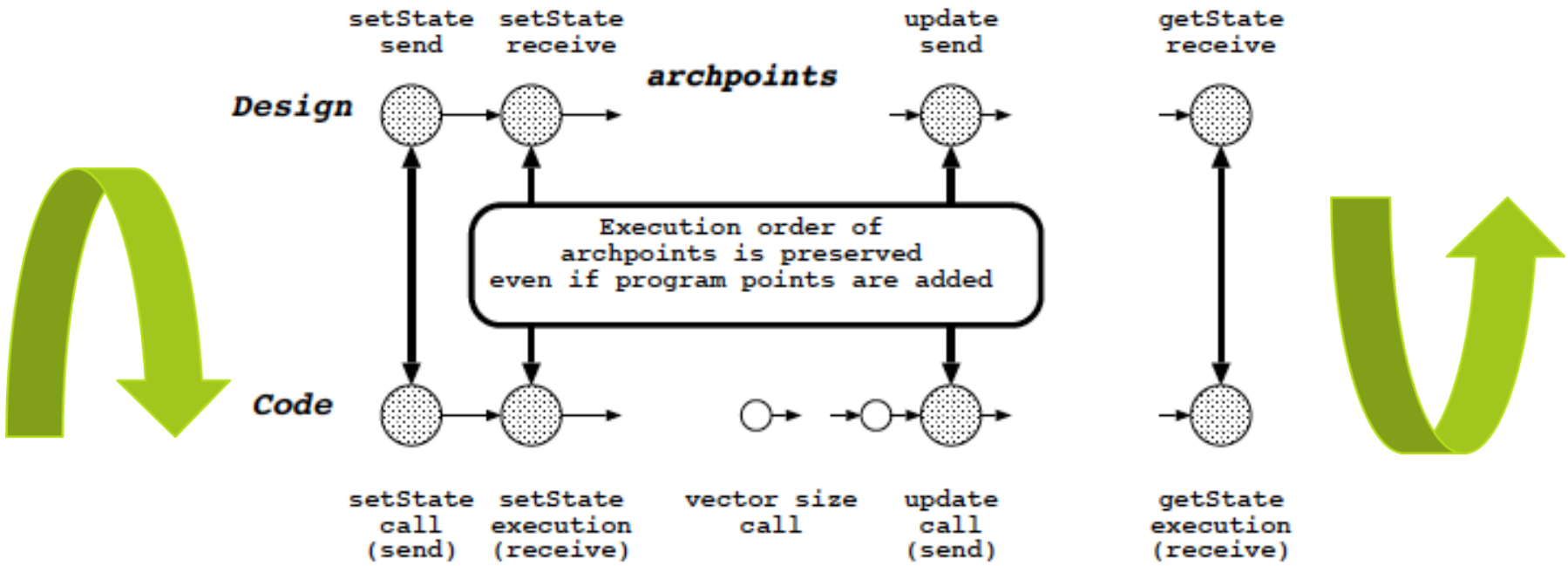
Archpoints, Program points, …

Diagram	Archpoint	Program point (Java)	Pointcut
Class diagram (UML)	class	class definition	`class`
	method	method definition	`method`
	field	variable definition	`field`
Sequence diagram (UML)	message send	method call	`call*`
	message receive	method execution	`execution*`
	(control flow)	(control flow)	`(cflow)*` **
Constraint		**Target**	**Predicate**
Structure	inheritance	class	`inherit`
	association	class	`assoc`
	membership	class, method, field	`memberof`
Behavior	sequence	message send/receive call, execution	`sequence`
	iteration	sequence	`iteration`
	branch	sequence	`alt`

*) AspectJ pointcut, **) Used with `call` or `execution` pointcut

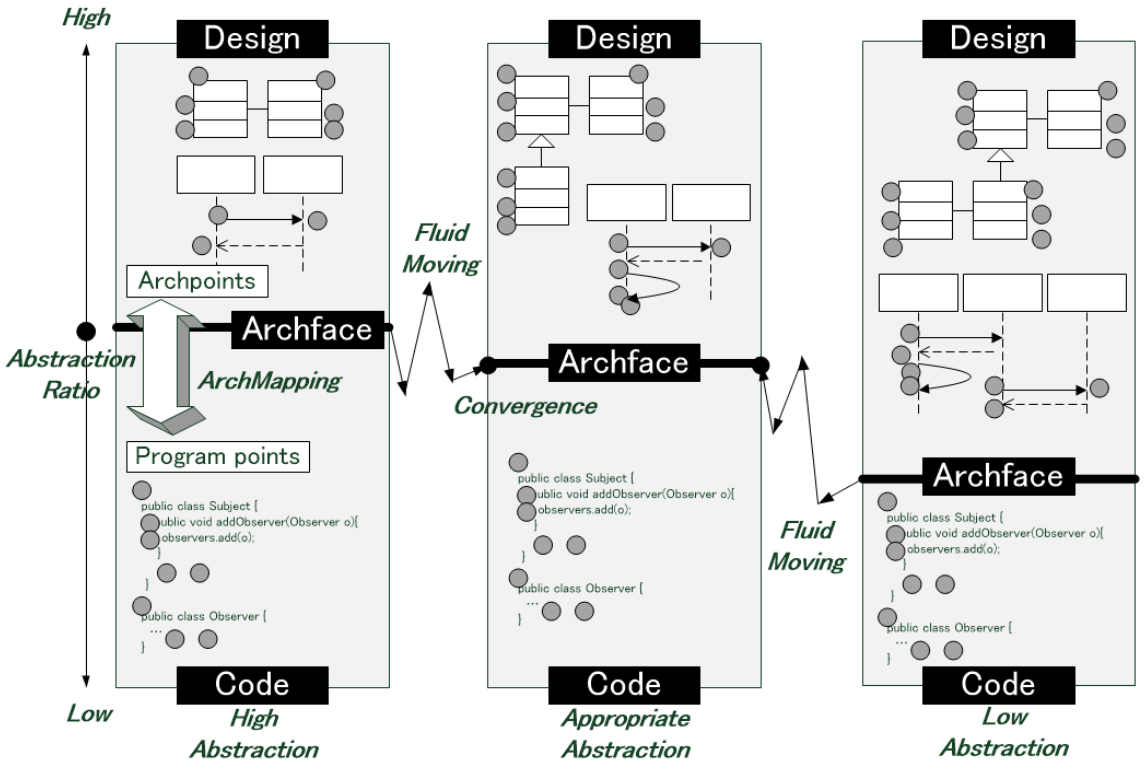
AOP based archpoint selection!

Abstraction = Bisimulation in terms of archpoints



Abstraction level can be defined by selecting archpoints!

Archface Guided Abstraction Refinement



Design and code can co-evolve each other by

- **fluidly moving between them and**
- **seeking an appropriate abstraction level.**

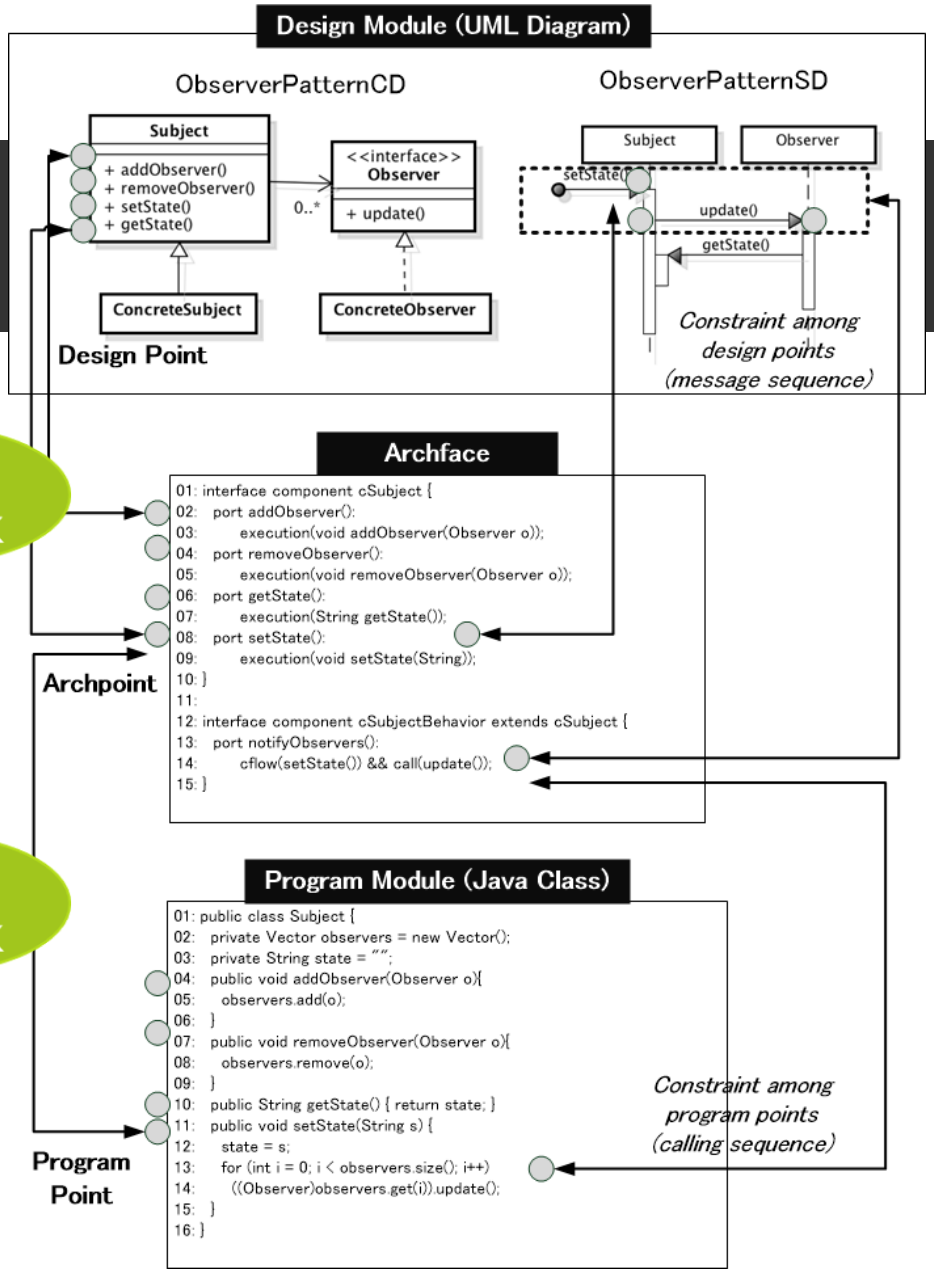
Inspired by CEGAR !

Integration of Design and Program Modules

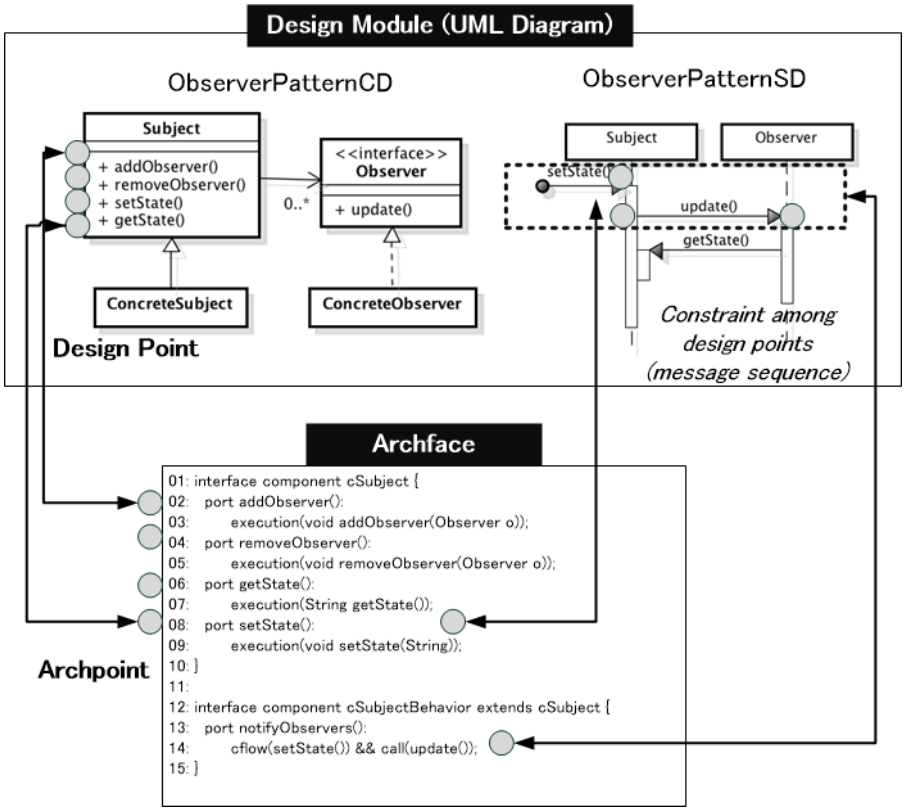
Integration

Does a design module model an archface?

Does a program module implement an archface?



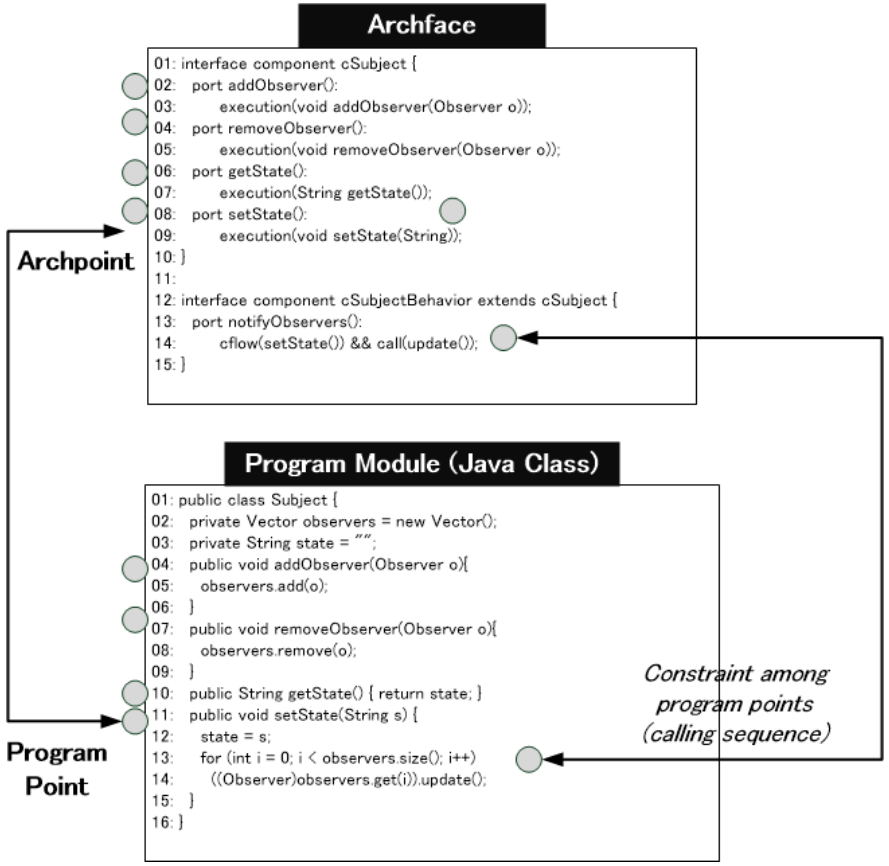
Type Check for a Design Model



Check if an archpoint is modeled as a design point.

Ubayashi, N. and Kamei, Y., Verifiable Architectural Interface for Supporting Model-Driven Development with Adequate Abstraction Level, MiSE 2012.

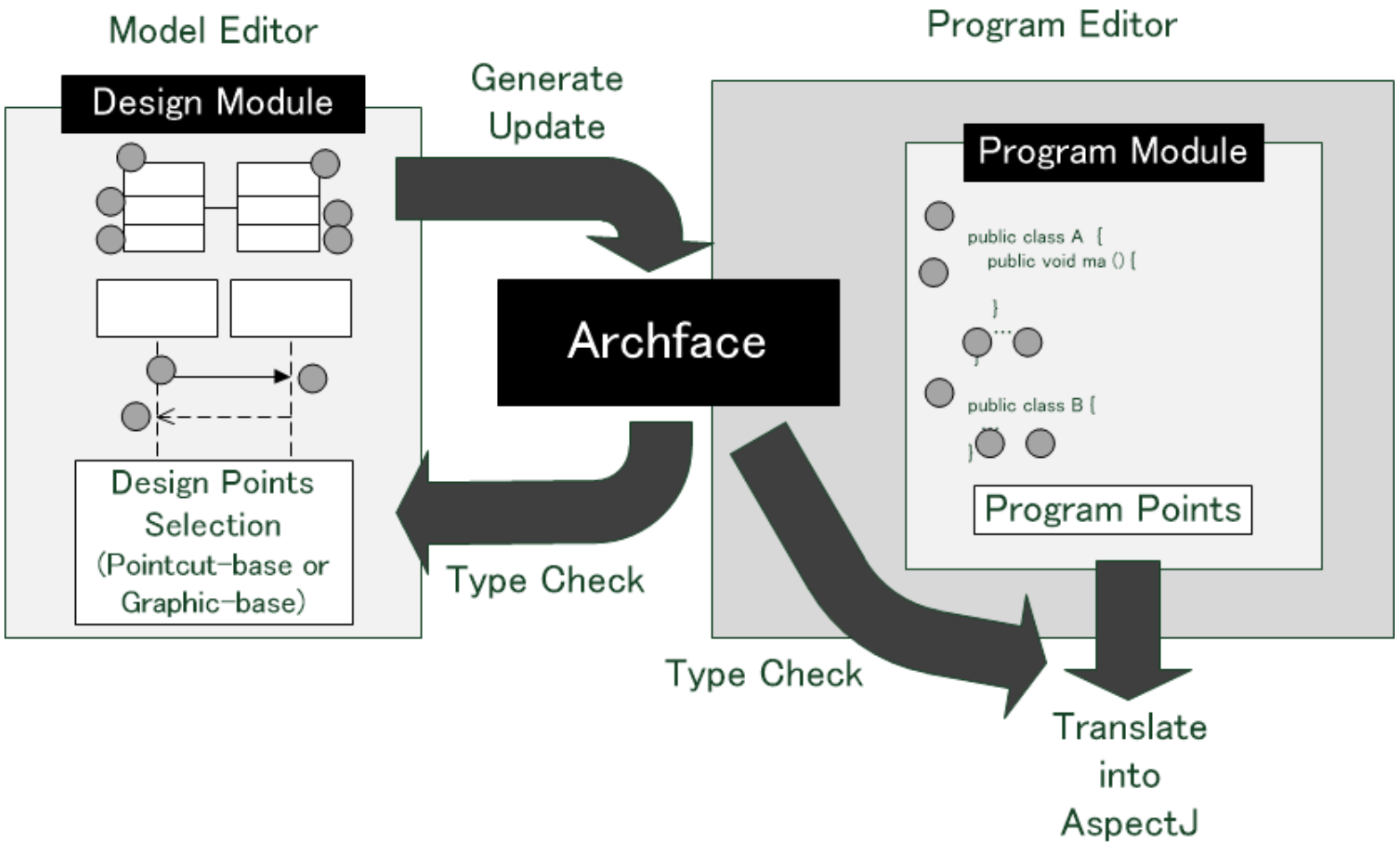
Type Check for Program Code



Check if an archpoint is implemented as a program point.

Archface-Centric IDE

Ongoing Work

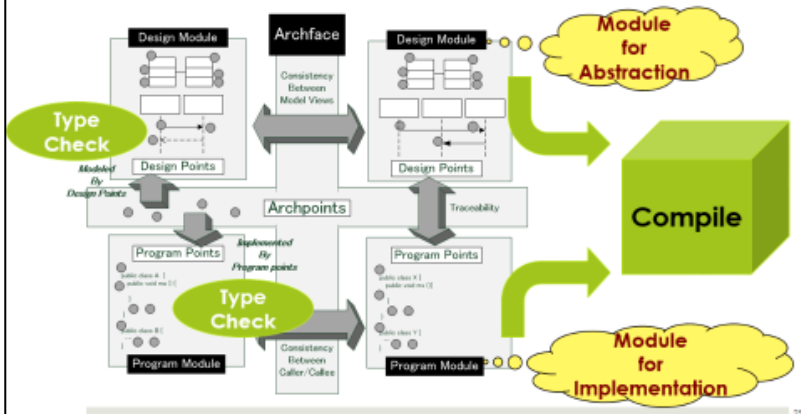


Conclusions

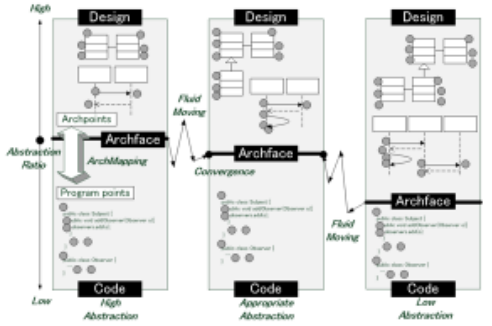
Summary

- We rethought both modularity and compilation **in the light of the abstraction** between design and implementation.
- In our approach, not only program code but also a design model is a module.
- Archface plays an important role in our modularity vision.

Archface-Centric Modularity



Archface Guided Abstraction Refinement

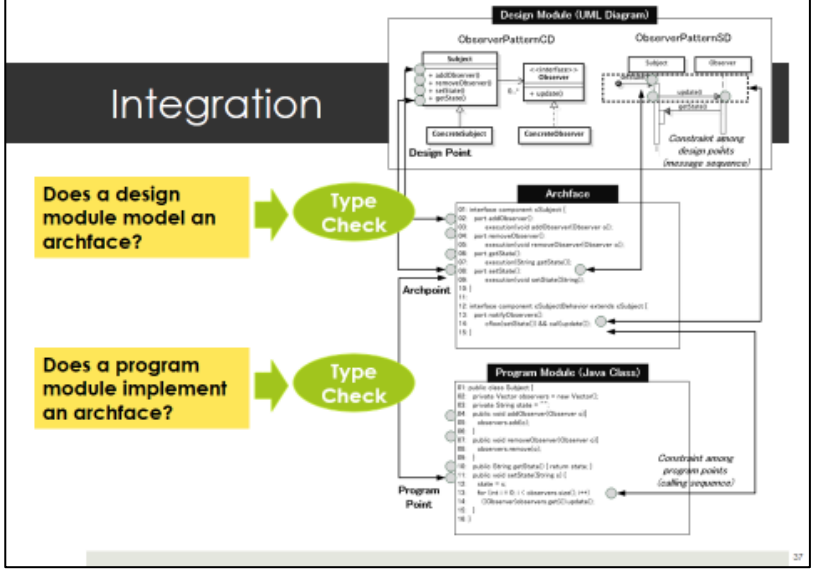


Design and code can co-evolve each other by

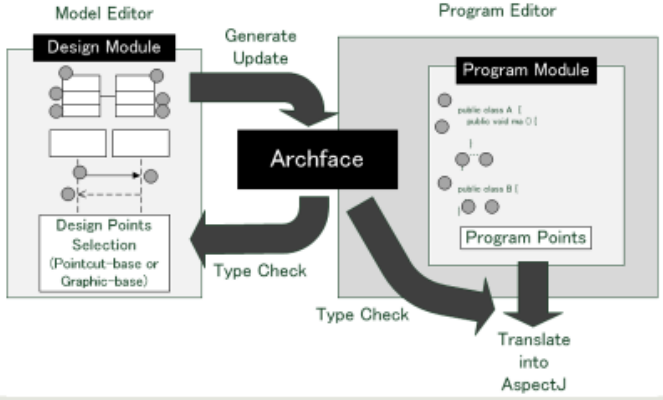
- fluidly moving between them and
- seeking an appropriate abstraction level.

Inspired by CEGAR !

Integration



Archface-Centric IDE **Ongoing Work**



Thank you for your attention.

I love modeling.
I love programming too.