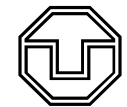


# 11. Frameworks and Patterns - Framework Variation Patterns

1

Prof. Dr. U. Aßmann  
Software Engineering  
Faculty of Informatics  
Dresden University of  
Technology  
Version 12-1.0, 12/1/12

1. Open Role Framework Hooks
2. Framework Hook Patterns
3. Delegation-Based Framework Hook Patterns
4. Recursion-Based Framework Hook Patterns
5. Unification-Based
6. Inheritance-Based
7. T&H in Frameworks



# Literature (To Be Read)

2

- ▶ W. Pree. Framework Development and Reuse Support. In Visual Object-Oriented Programming, Manning Publishing Co., editors M. M. Burnett and A. Goldberg and T. G. Lewis, Pp, 253-268, 1995.  
[www.softwareresearch.net/publications/J003.pdf](http://www.softwareresearch.net/publications/J003.pdf)
  - Or: D. Karlsson. Metapatterns. Paper in Design Pattern seminar, IDA, 2001. Available at home page.
- ▶ D. Bäumer, G. Gryczan, C. Lilienthal, D. Riehle, H. Züllighoven. Framework Development for Large Systems. Communications of the ACM 40(10), Oct. 1997.  
<http://citeseer.ist.pst.edu/bumer97framework.html>

# Secondary Literature

3

- ▶ W. Pree. Design Patterns for Object-oriented Software Development. Addison-Wesley 1995. Unfortunately out of print.
- ▶ M. Fontoura, W. Pree, B. Rumpe. The UML Profile for Framework Architectures. Addison-Wesley, Object Technology Series. 2002.

# Goal

4

- ▶ Studying variabilities of frameworks with the T&H concept
- ▶ Introducing different types of hooks for frameworks and components (TH patterns)
- ▶ Understand framework hook patterns
  - The box-like notation for frameworks and framework hooks patterns
- ▶ More types of dimensional frameworks

# Patterns and Frameworks

5

- ▶ Historically, design patterns were discovered during framework development
  - Smalltalk MVC [Goldberg, Reenskaug]
  - ET++ [Gamma]
  - Interviews [Vlissides]
- ▶ Design patterns are building blocks of frameworks
  - Framework developers vary and extend classes of the framework
- ▶ Design patterns are for the making of the products of a product line architecture
  - Application developers vary and extend classes of the framework
  - Variability design patterns can be used as *framework variation points (framework variation hooks)*
  - Extensibility design patterns can be used as *framework extension points (framework extension hooks)*



# 11.1 Framework Instantiation and Merging With Open Roles

---

---

6

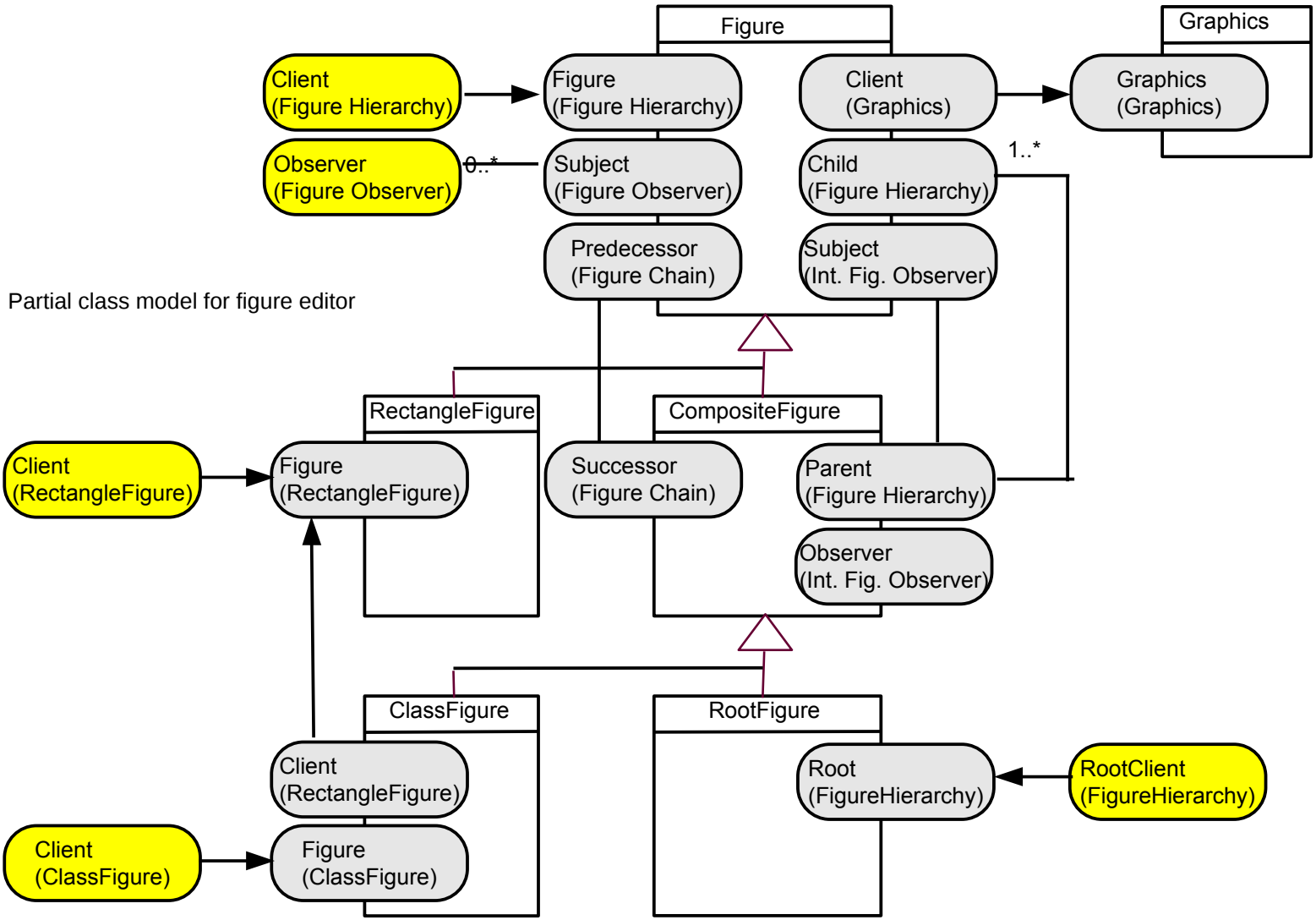


# Framework Instantiation with Open Roles (Role Hot Spots)

7

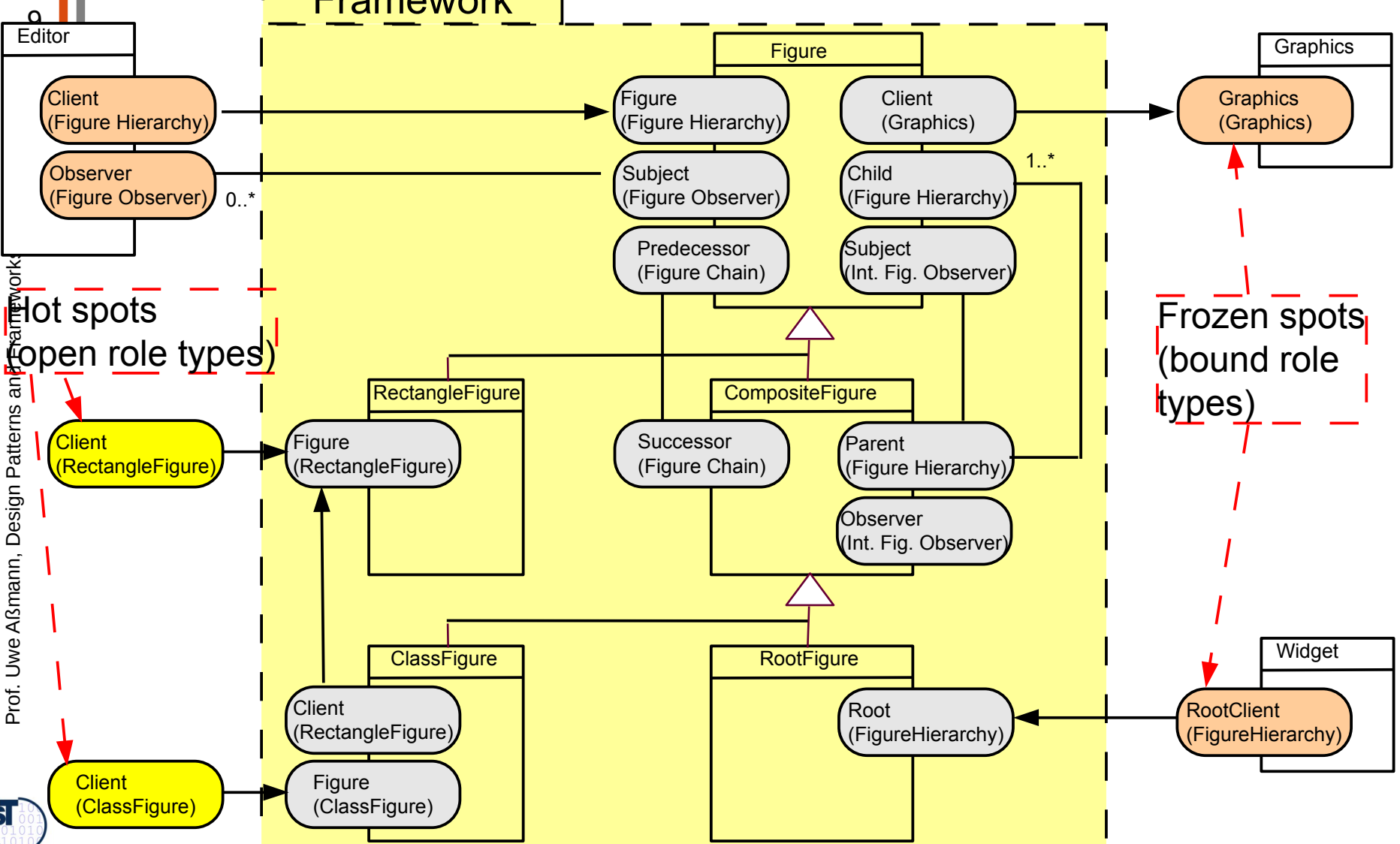
- ▶ The most simple form of framework instantiation is Riehle/Gross' *open role instantiation* based on association
  - Here, frameworks are class models with “open” role hot spots
  - *Open role hooks (free, unbound abilities)* are role types that have not yet been assigned to classes
- ▶ The hot spots form an *integration repertoire (integration role type set)*
  - the set of role types, by which the framework can be integrated into an application
  - *Aka framework hooks, framework variation points*
- ▶ A framework is *instantiated* by binding its integration repertoire to classes
  - The abilities are *bound*, role constraints have to be respected
- ▶ Hence, role models play the bridge between a framework and its clients

# Remember: The Partial Figure Model, a Standard Class-Ability Model



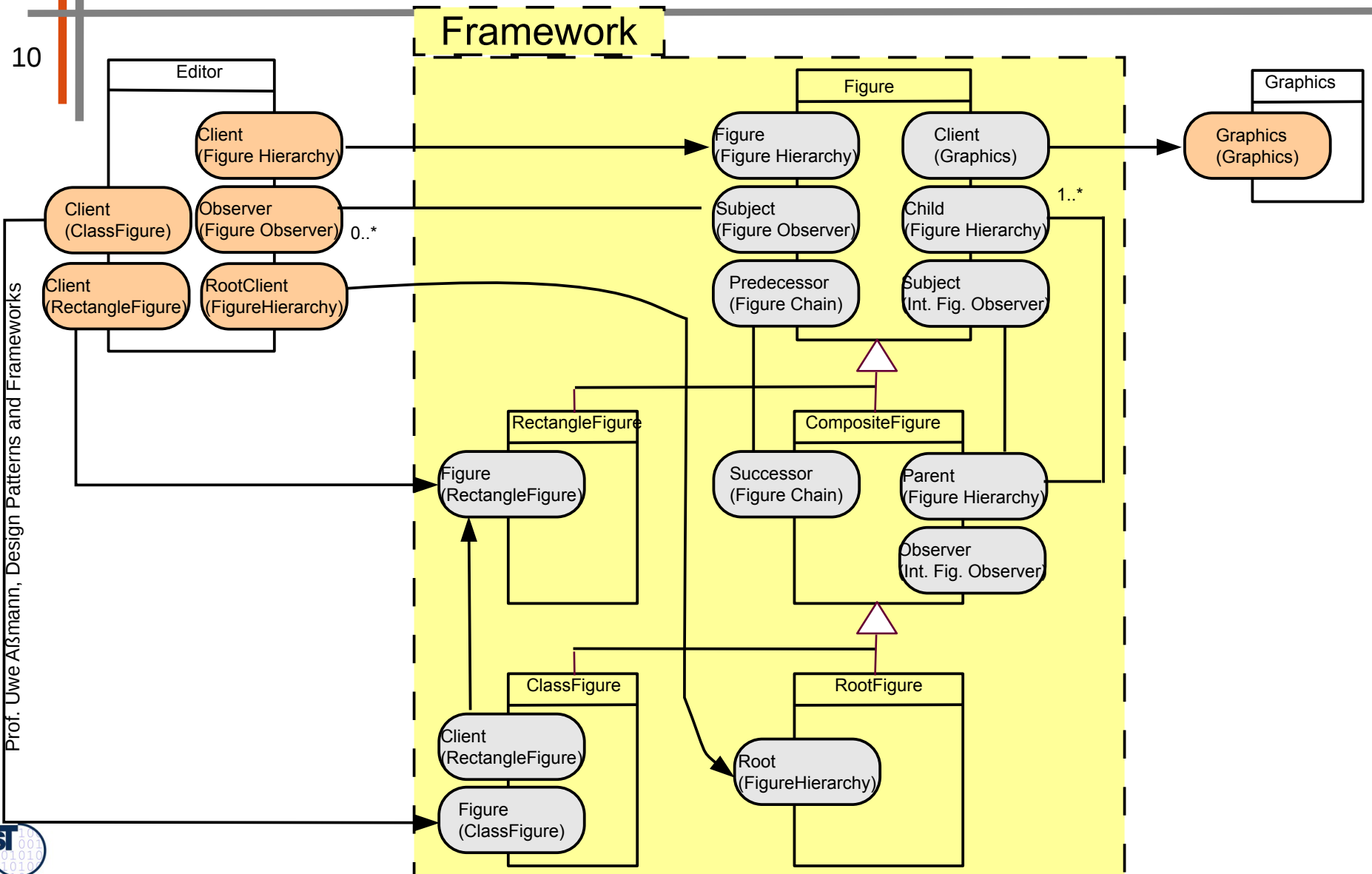


# The Figure Framework, Partially Instantiated

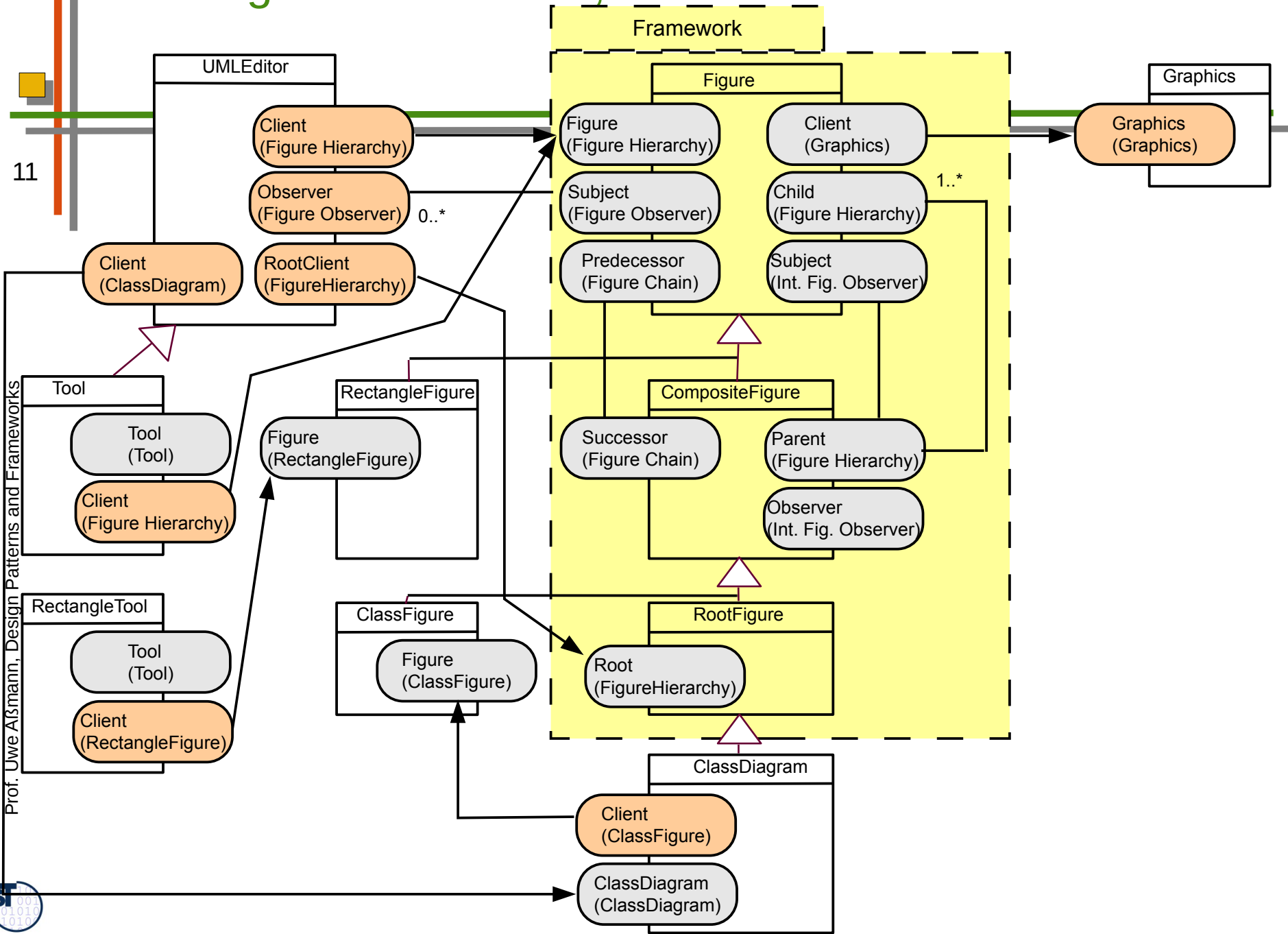


# The Figure Framework, Fully Instantiated to an Editor

10



# The Figure Framework, Instantiated to an UML Editor

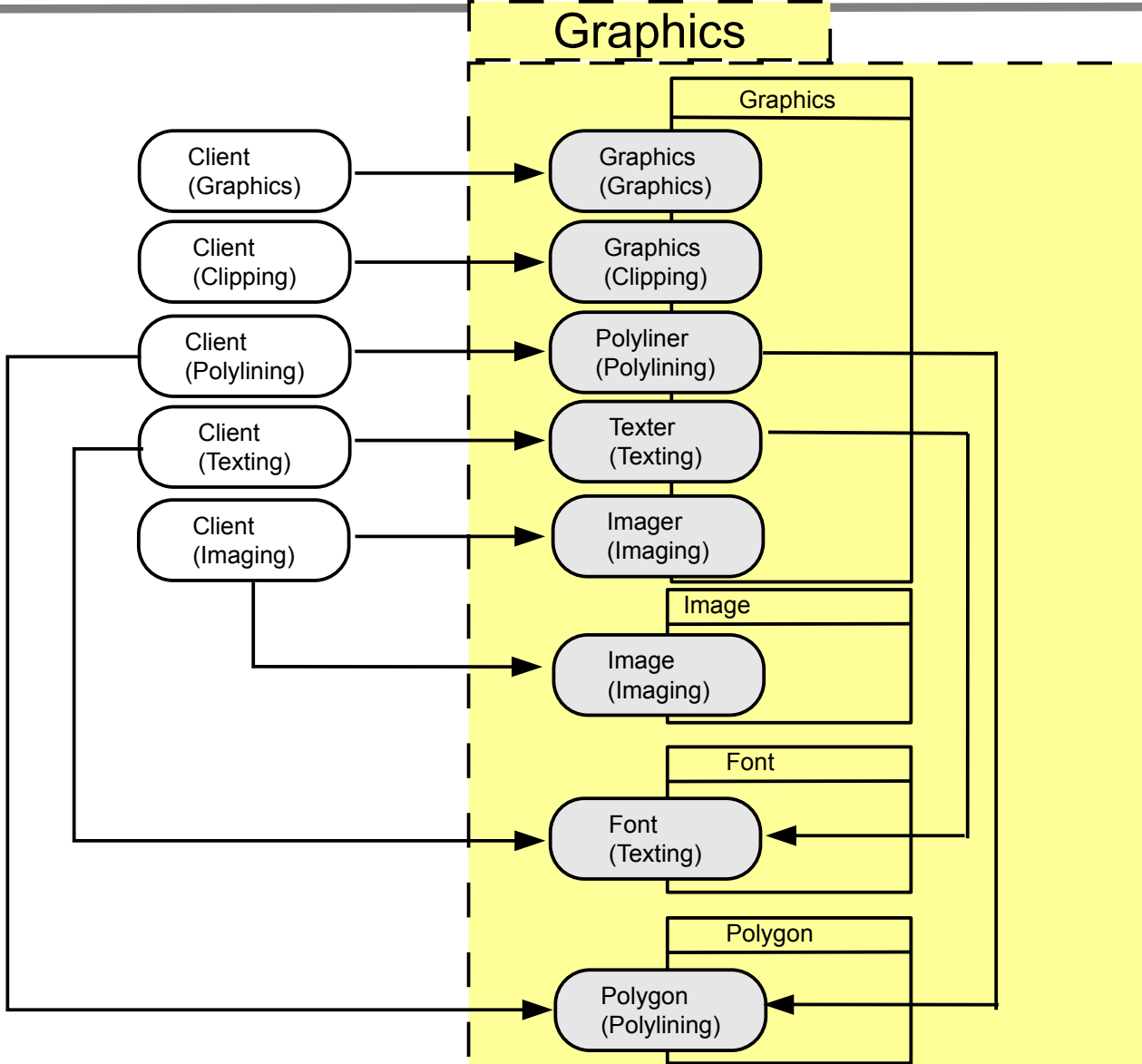


# Merging of Frameworks

12

- ▶ Two frameworks are *merged* by binding the integration abilities of A to classes of B
  - Role constraints have to be respected
- ▶ Hence, role models play the bridge between different frameworks
  - Or layers of frameworks

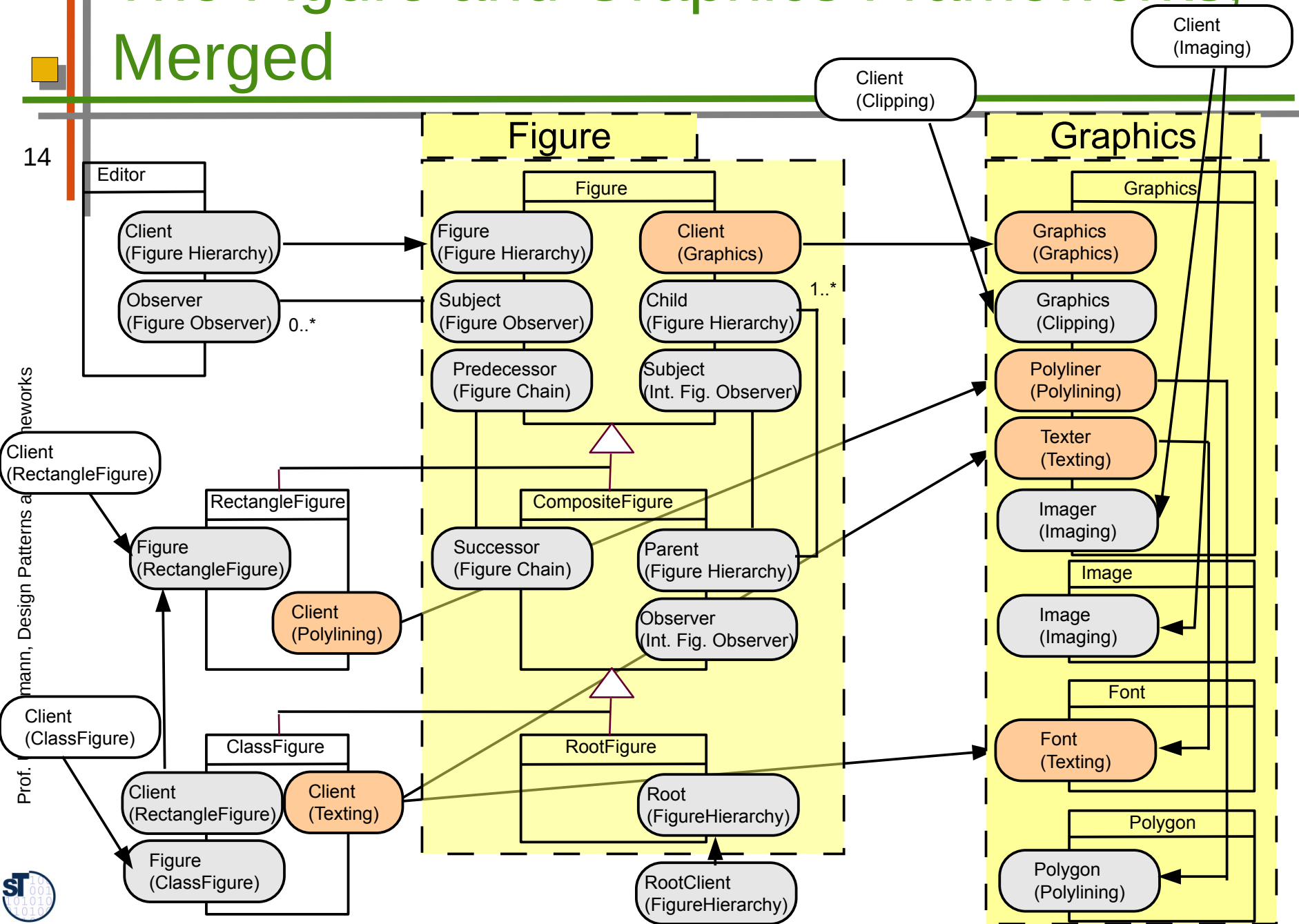
# A Graphics Framework



# The Figure and Graphics Frameworks, Merged

14

Prof. Dr. Ingrid Isenhardt, Design Patterns and Frameworks



# Limitations of Open Role Instantiation

15

- ▶ [Riehle/Gross] role-based framework instantiation relies on simple role binding, with role constraints
- ▶ Role binding for framework instantiation and merging can be even more elaborated



# 11.2 Framework Hook Patterns

---

---

16

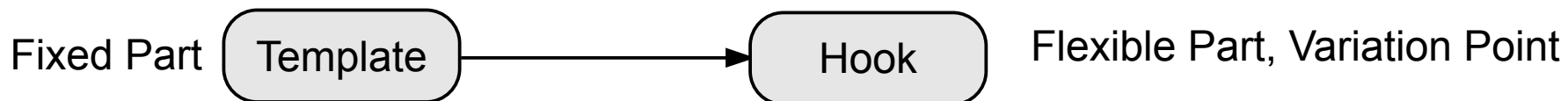




# Pree's Framework Hook Patterns (Template&Hook Role Models)

17

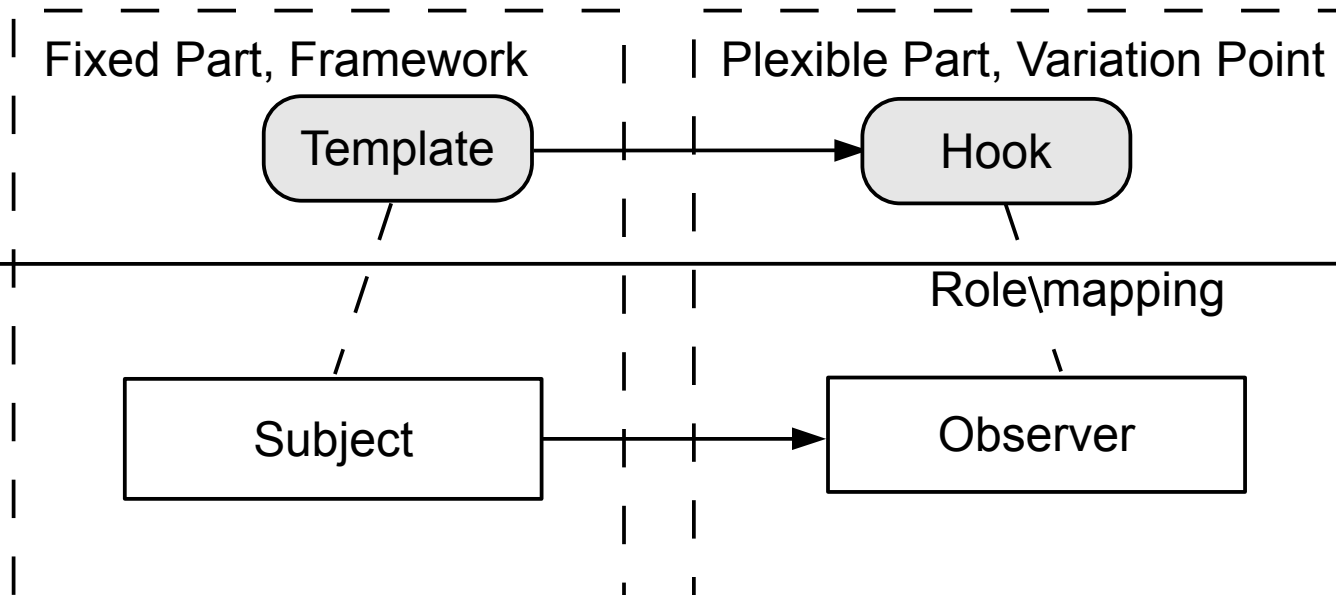
- ▶ In Pree's work, *framework hooks* are characterized by design patterns (*framework hook patterns*)
  - They describe the roles of classes on the *border* of the framework
  - The framework hook pattern determines the way how the classes interact with each other at the border of the framework
- ▶ A framework variation point is characterized with a *Template&Hook conceptual pattern*
  - Pree called this a *T&H metapattern*, we call this a *T&H role model*
- ▶ A T&H role model has 2 parts:
  - A template class (or *template role type*), which gives the skeleton algorithm of the framework: Fix, grasps commonalities
  - A hook class, which can be exchanged (or: a *hook role type* which can be bound to a client class): Variable, even extensible, grasps variability and extension



# T&H Patterns and Standard Patterns

18

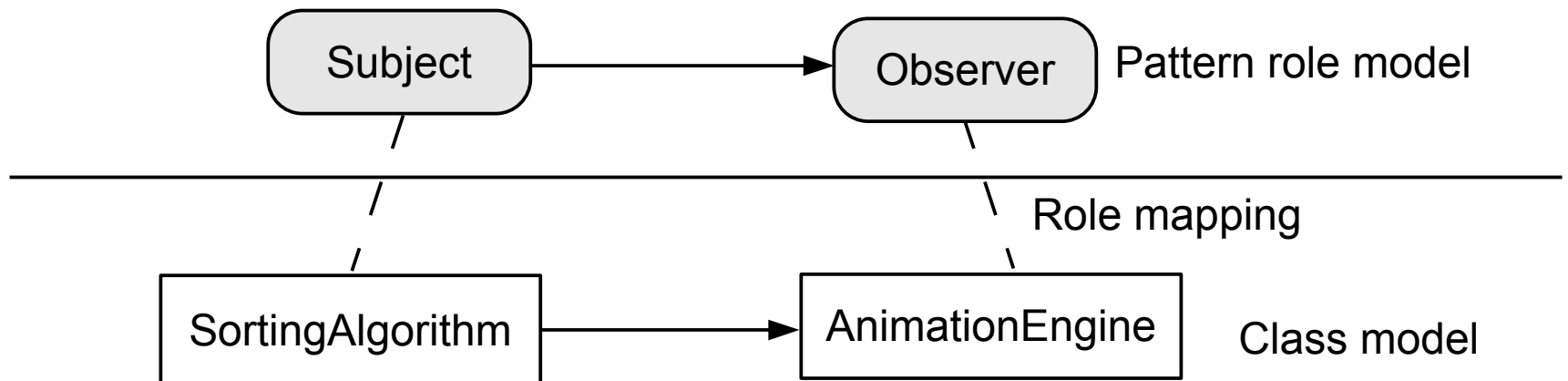
- ▶ A TH-role model *overlays* another pattern (hence Pree called it a *metapattern*)
  - The template part fixes parts of the pattern
  - The hook part keeps parts of the pattern variable, i.e., open for binding.



# T&H in Standard Design Patterns

19

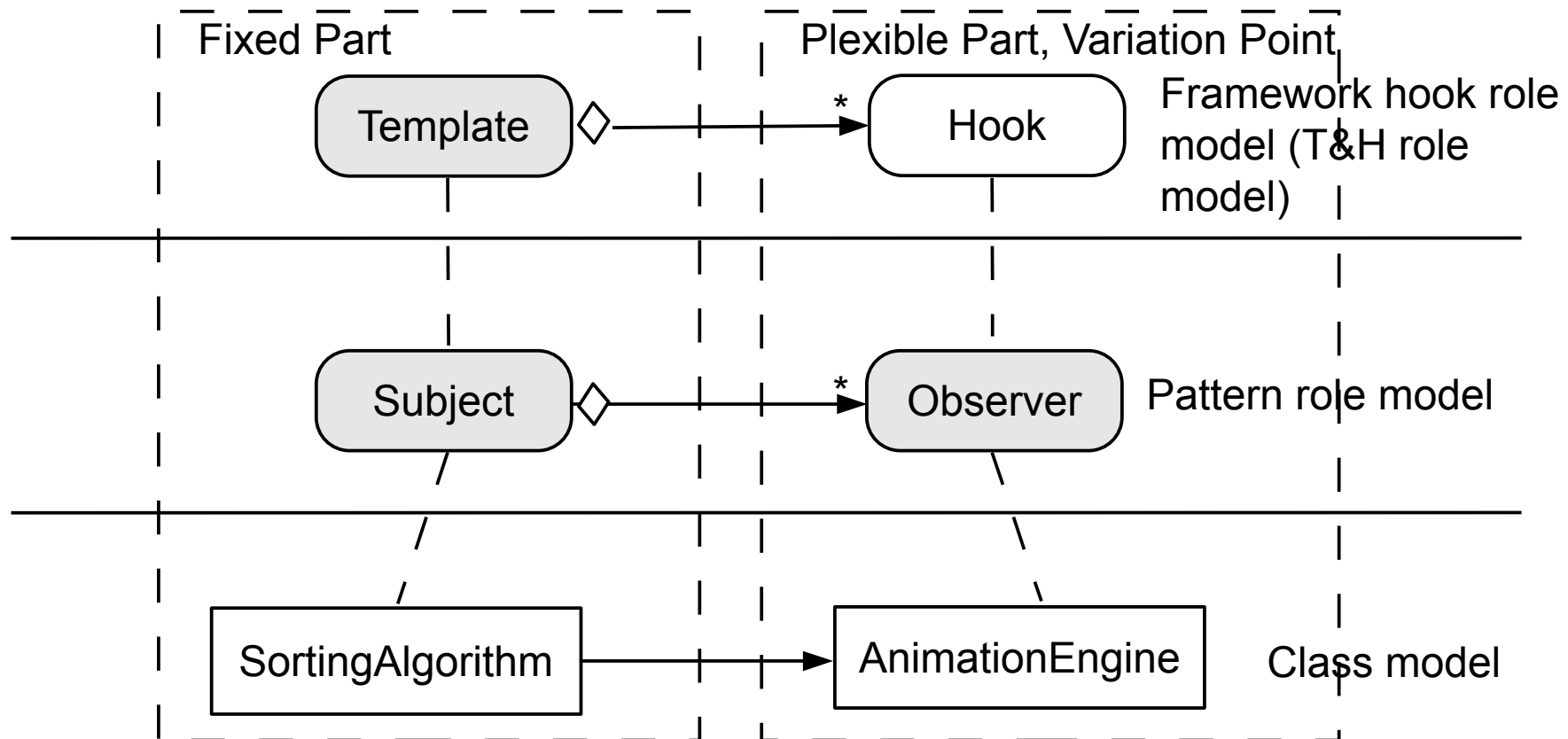
- ▶ Subject and Observer can vary; nothing is fixed
  - SortingAlgorithm and AnimationEngine can be exchanged



# T&H in Framework Hook Patterns

20

- ▶ Subject can no longer vary; it is fixed
  - SortingAlgorithm cannot be exchanged (expection: DimensionalClassHierarchies)



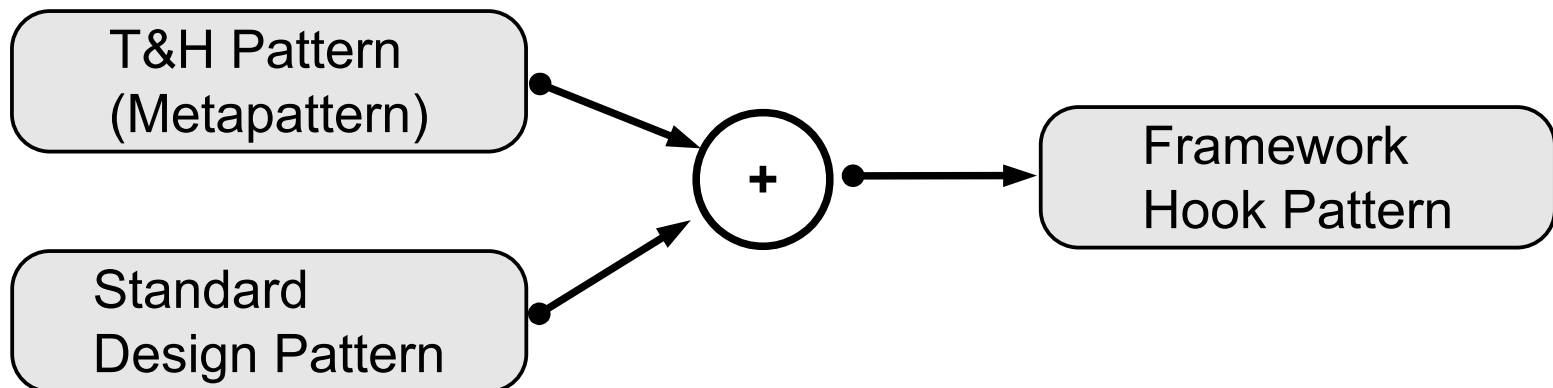
# Why T&H Patterns Add More to Standard Patterns

21

- ▶ Due to the Riehle-Gross Law, we know that metapatterns are role models that overlay the role models of design patterns
  - Metapatterns are very general role models that can be mixed into every design pattern
  - As design patterns describe application models, metapatterns describe design patterns
- ▶ In [Pree], roles are not considered. Pree has only hook classes and hook methods. Here, we combine [Pree] and [Riehle/Gross]

If a metapattern is overlayed to a role model of a design pattern, it adds commonality/variability knowledge, describing a *framework variation point*

- The template part characterizes the framework's fixed parts
- The hook part characterizes the framework's variation point
- ▶ Hence we call a design pattern with metapattern information **framework hook pattern**



# Framework Hook Patterns

22

- ▶ The template-hook role model
  - adds more pragmatics to a standard design pattern, information about commonality and variability. Hence, framework variation points are described
  - The template-hook role model adds more *constraints* to a standard design pattern. Some things can no longer be exchanged
- ▶ Pree discovered 7 framework hook patterns, i.e., 7 template-hook role models for framework hooks
  - The template-hook role models describe the parameterization of the framework by *open role hooks*
  - They include Riehle's open role hooks, but add more variants
  - There are even other ones (see next chapter)

# Remark

23

- ▶ Note: we mean in the following:
  - with the role *Template*, that the class of the role type belongs to the framework
  - with the role *Hook*, that the class of the role type belongs to the application
  - with the role *TemplateM(ethod)* that the role defines a template method, *calling* a hook method *HookM(ethod)*
- ▶ Problem: Pree uses *TemplateM/HookM*, but calls them *Template/Hook*
  - and varies *HookM* classes, which is misleading because the variation is actually in the framework and the fixed part in the application

# Differences between Standard Patterns and Framework Hook Patterns

24

## ▶ Standard design pattern

- Often, no template parts; everything flows (exception: TemplateClass and -Method)
- Rich pattern and role model
- Applicable everywhere in the framework
- No T&H metapattern overlayed

## ▶ Framework hook pattern

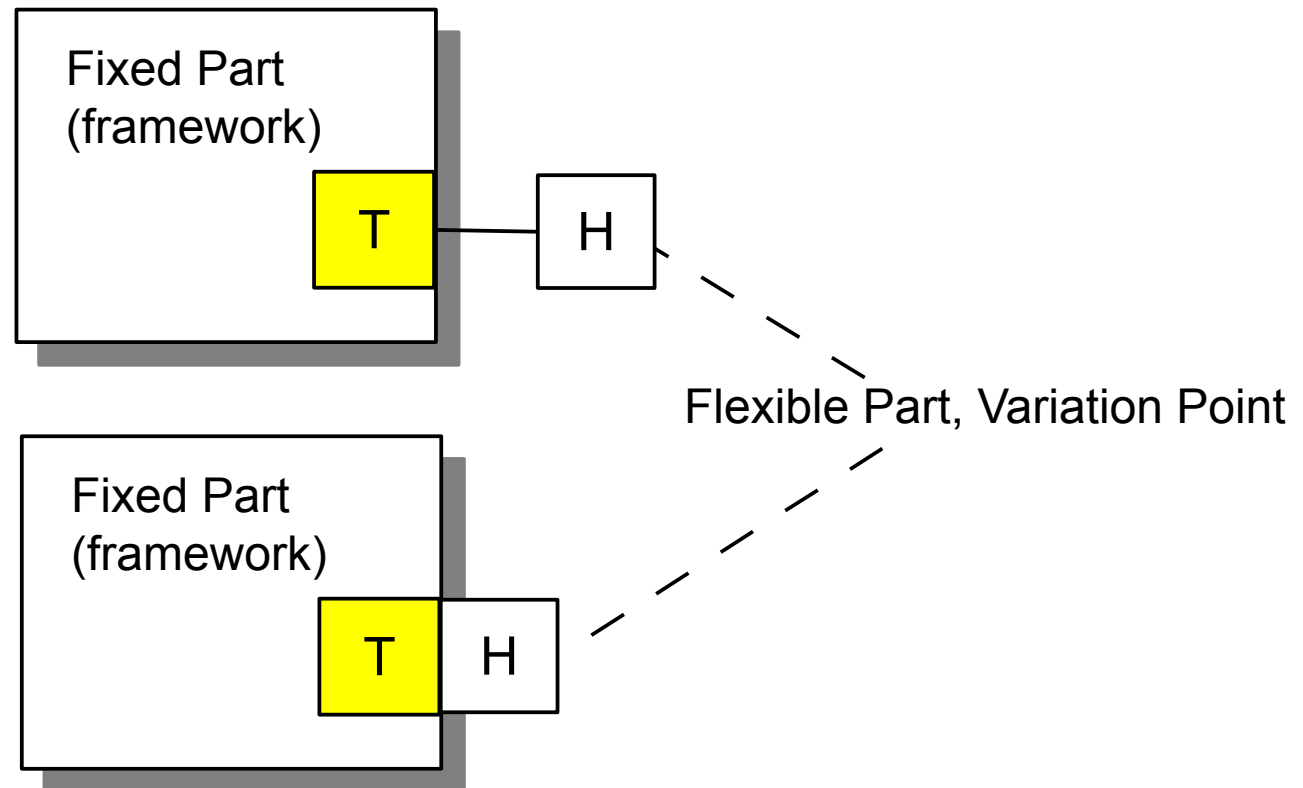
- Fixed and variable part
- Elementary pattern and role model
- Applicable only *at the border* of the framework,
- or at the border of a component, i.e., in an “interface”
- One T&H metapattern overlayed



# A Simple Notation for Framework Hook Patterns

25

- ▶ Insight: A framework hook pattern does something like this
  - It provides a design pattern at the border of a framework
  - It combines a T&H role model with standard role models





# 11.3 Delegation-Based Framework Hook Patterns

---

---

26



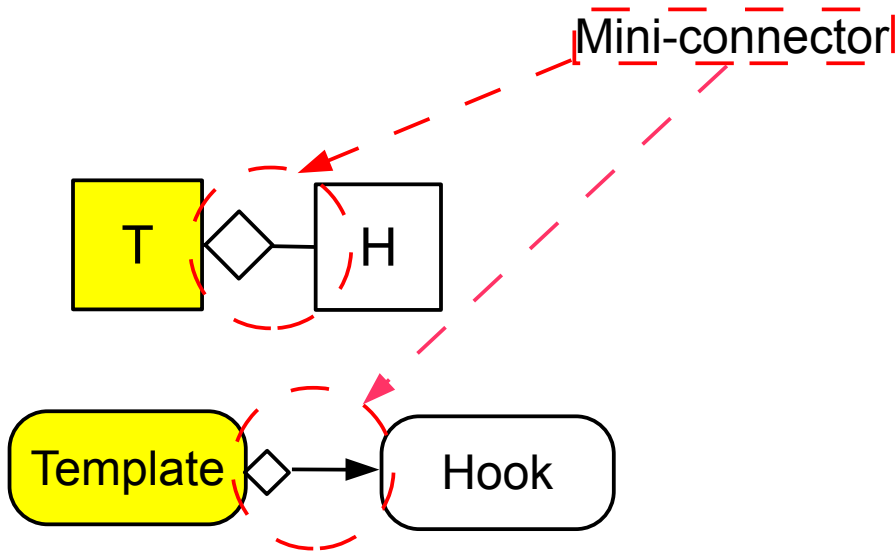
# T—H Connection Pattern

27

- ▶ T&H *connection pattern* (T--H framework hook)
  - Similar to Riehle/Gross open role type, but with aggregation instead of association
  - T and H classes are coupled by a template-hook role model, the hook is a delegatee (the relation is called a *mini-connector*)
  - “Whole” is in the framework, “Part” is in the plugin

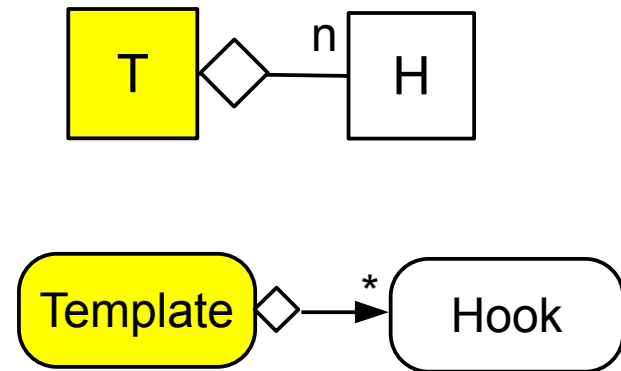
## 1-T—H (aggregated open role hook)

H part of T



## n-T—H (flat extension)

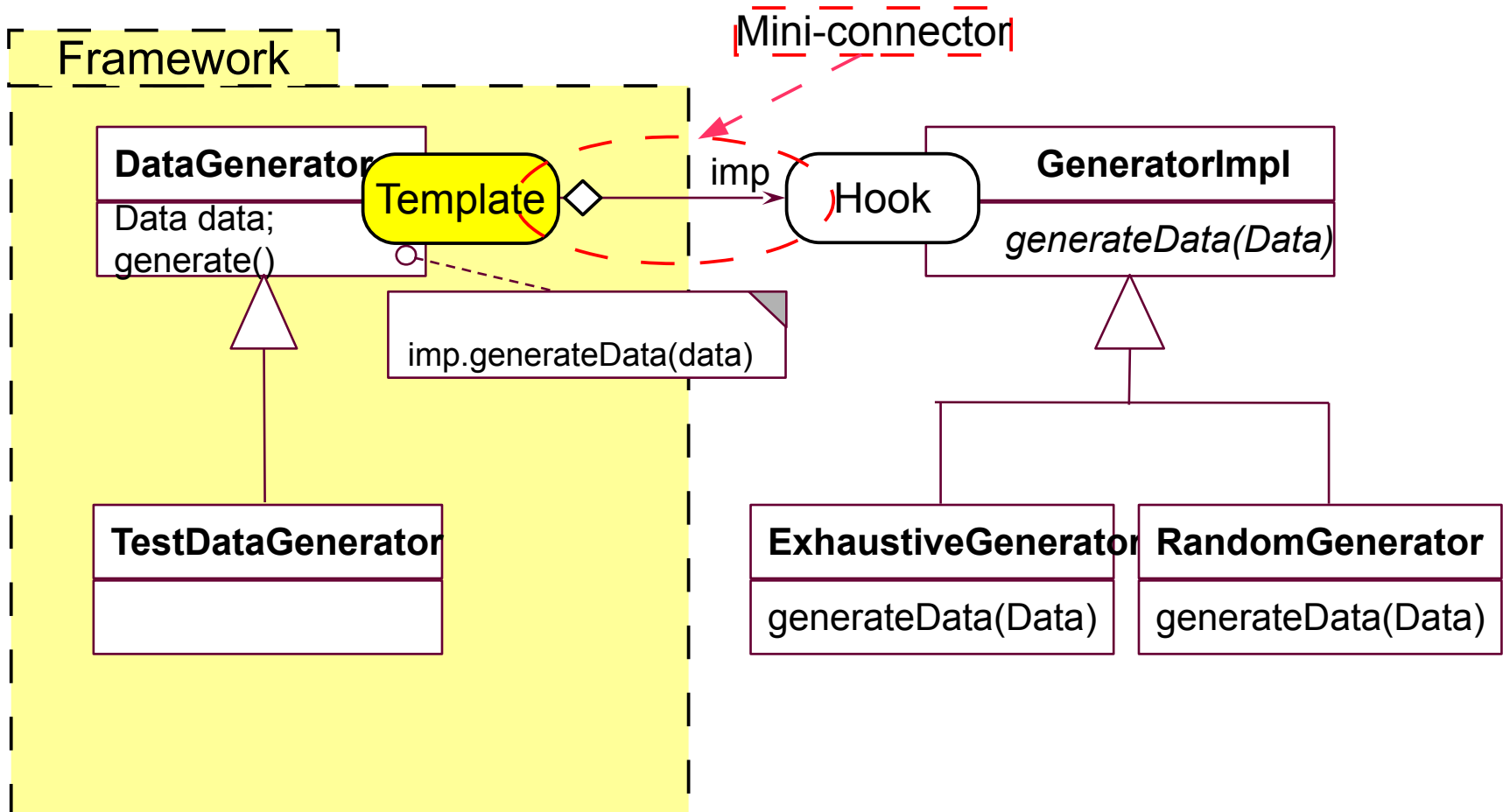
T has n H parts, n is dynamic



# TemplateClass with 1-T--H

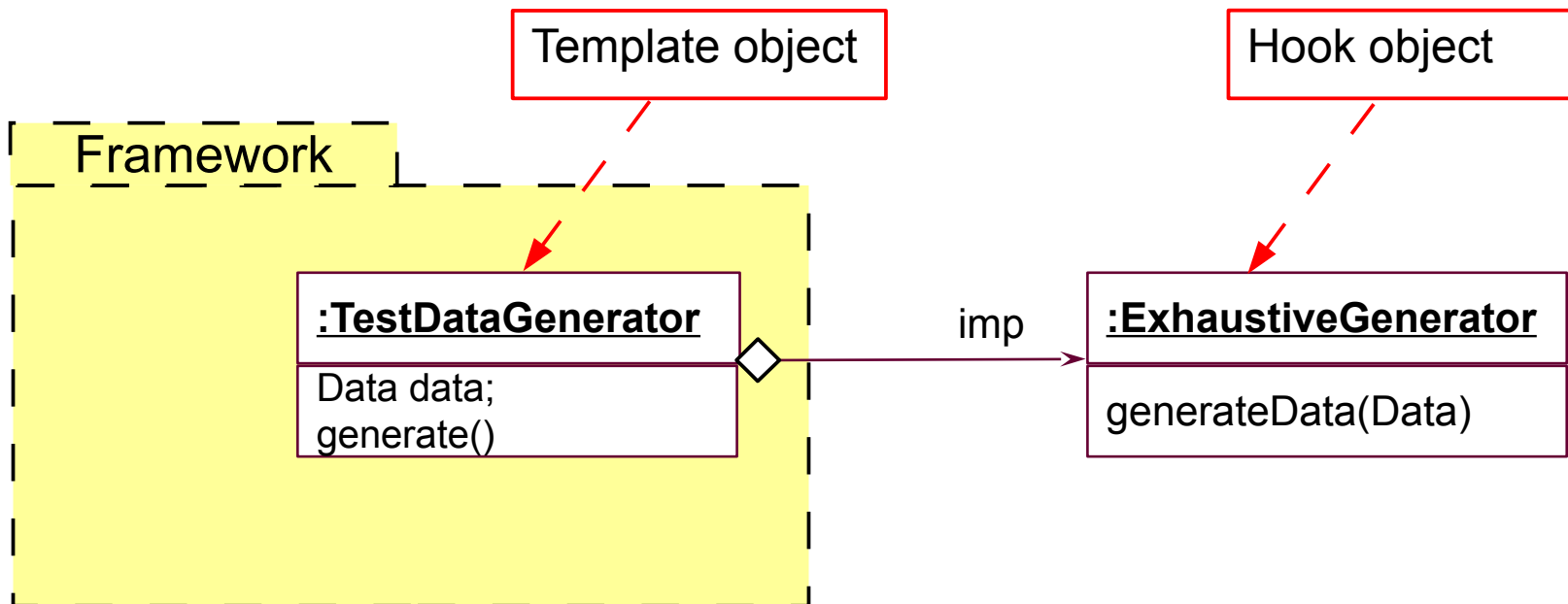
28

- Attention: in this case, the Template role also carries the TemplateM role (framework has template method, application has hook method)



# TemplateClass Runtime Scenario

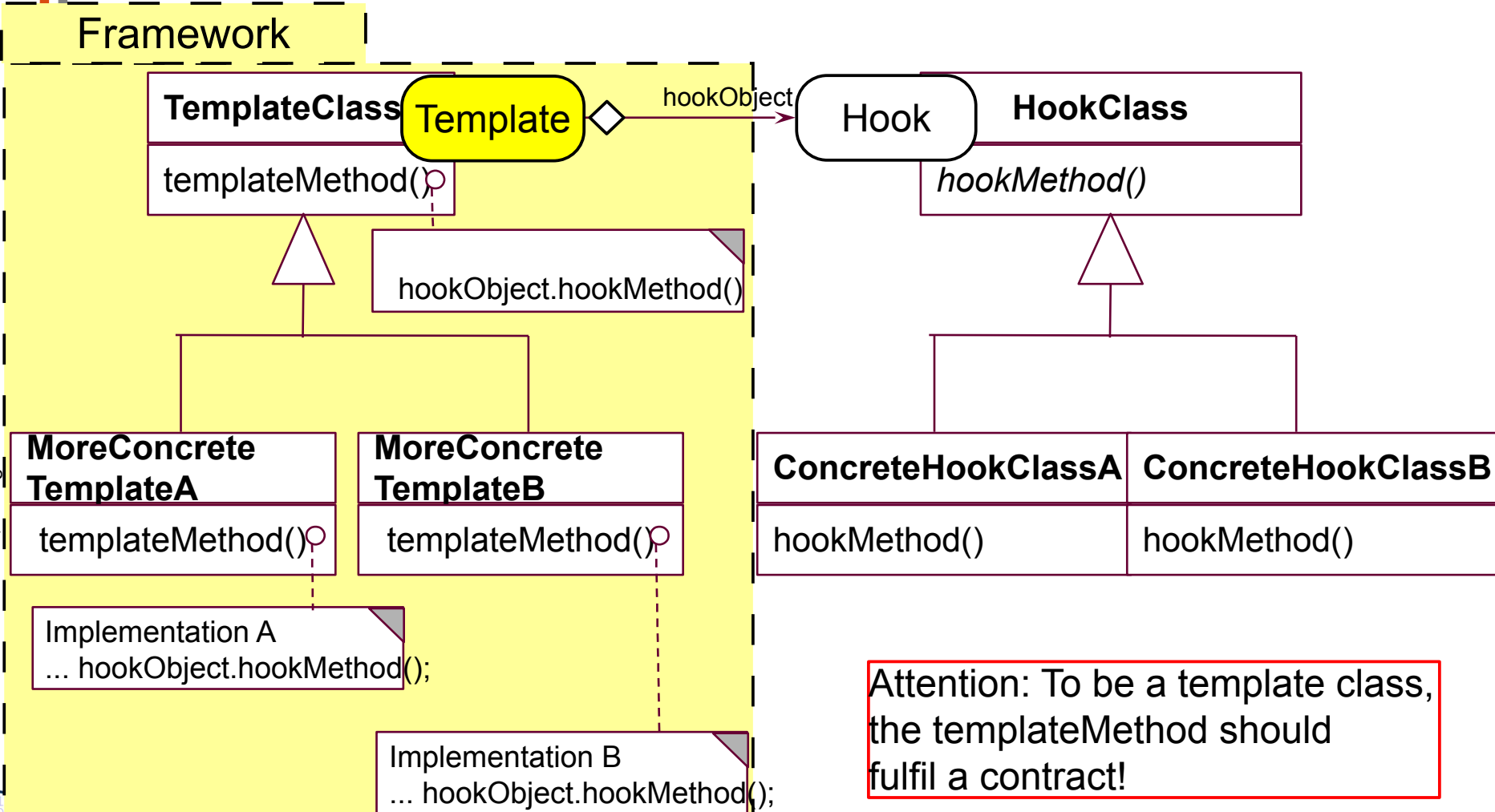
29



# Dimensional Hierarchies with 1-T--H (Bridge with Template/Hook Constraint)

30

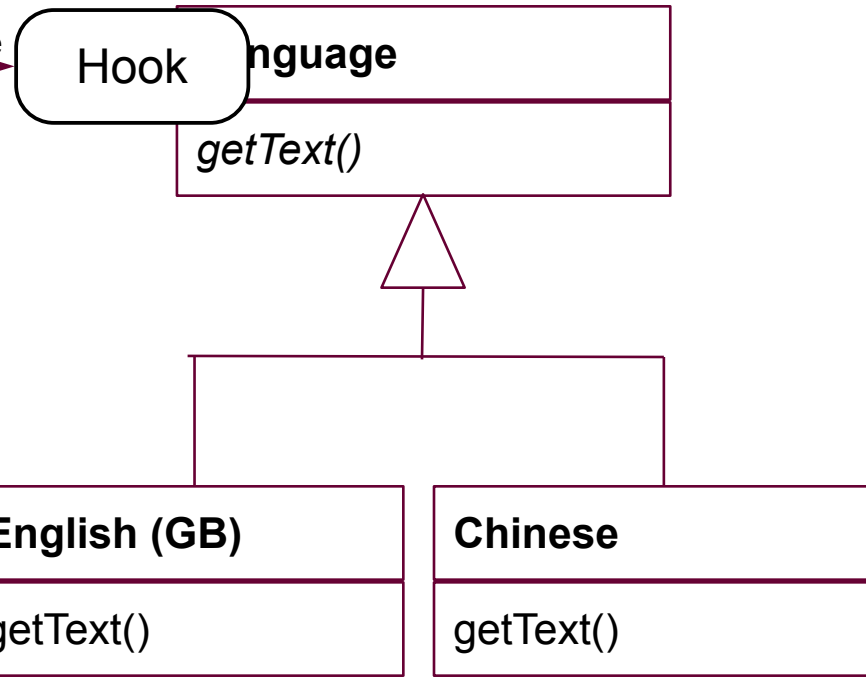
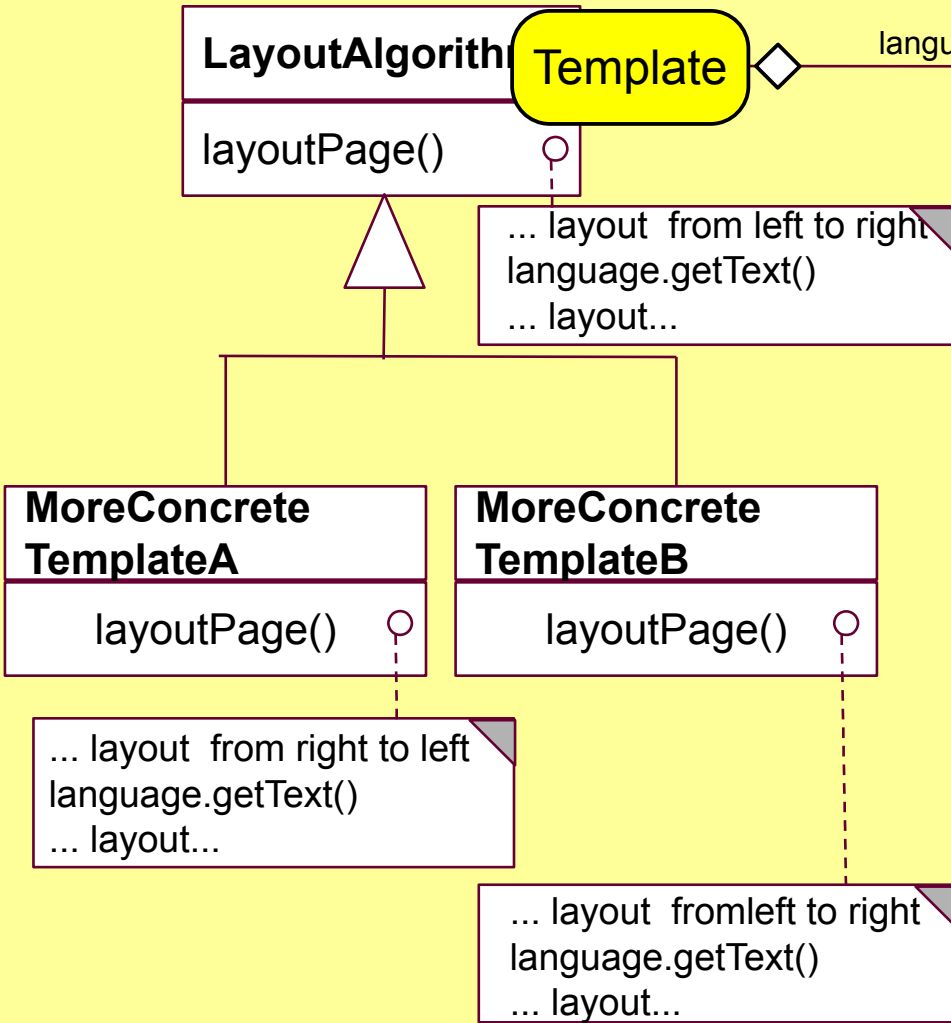
- ▶ Template classes cannot be varied by user, but by the hook subclass



Attention: To be a template class, the templateMethod should fulfil a contract!

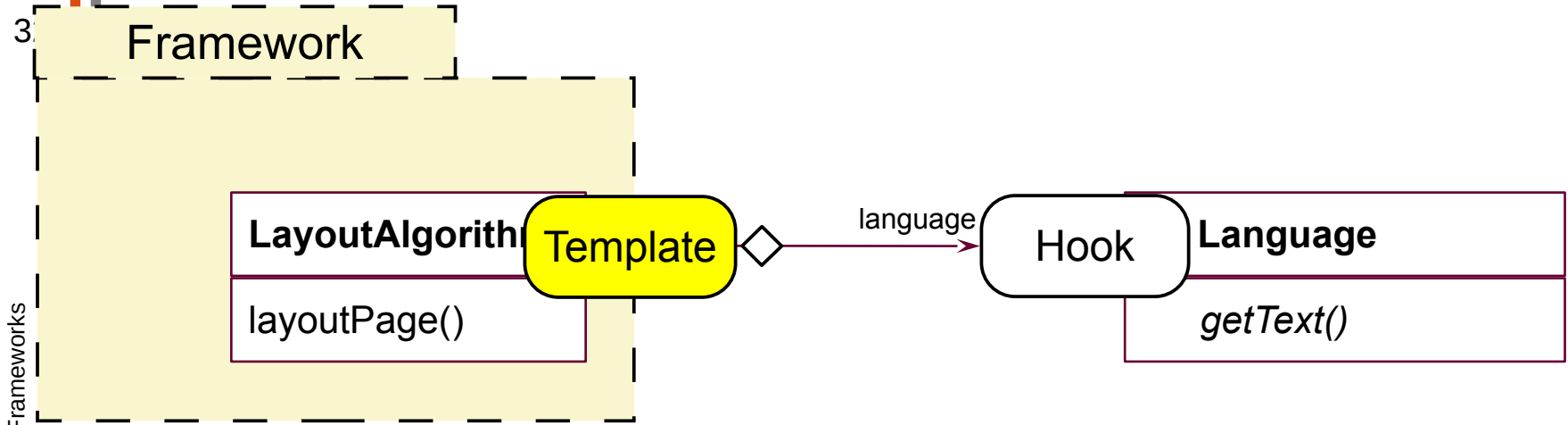
# Internationalization as Dimensional Class Hierarchy with 1-T--H

## Framework

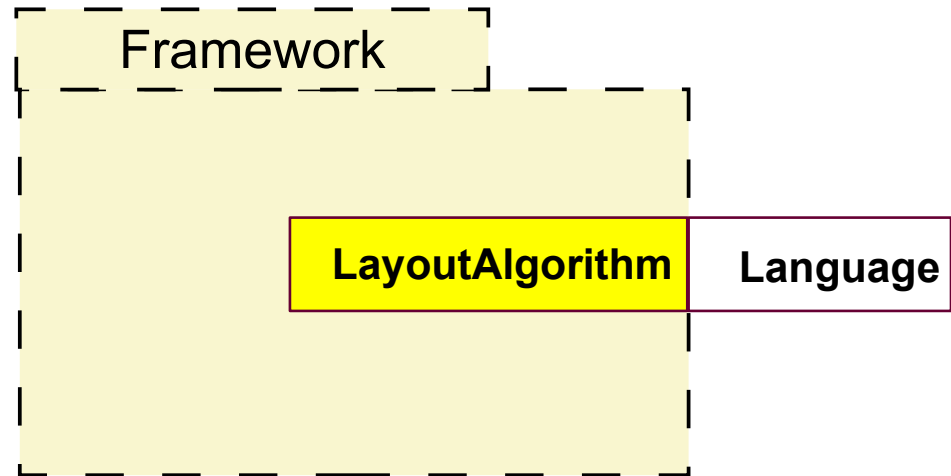


In the template class, the templateMethod fulfills the contract that all content of the page has been layouted.

# Internationalization as Dimensional Class Hierarchy with 1-T--H



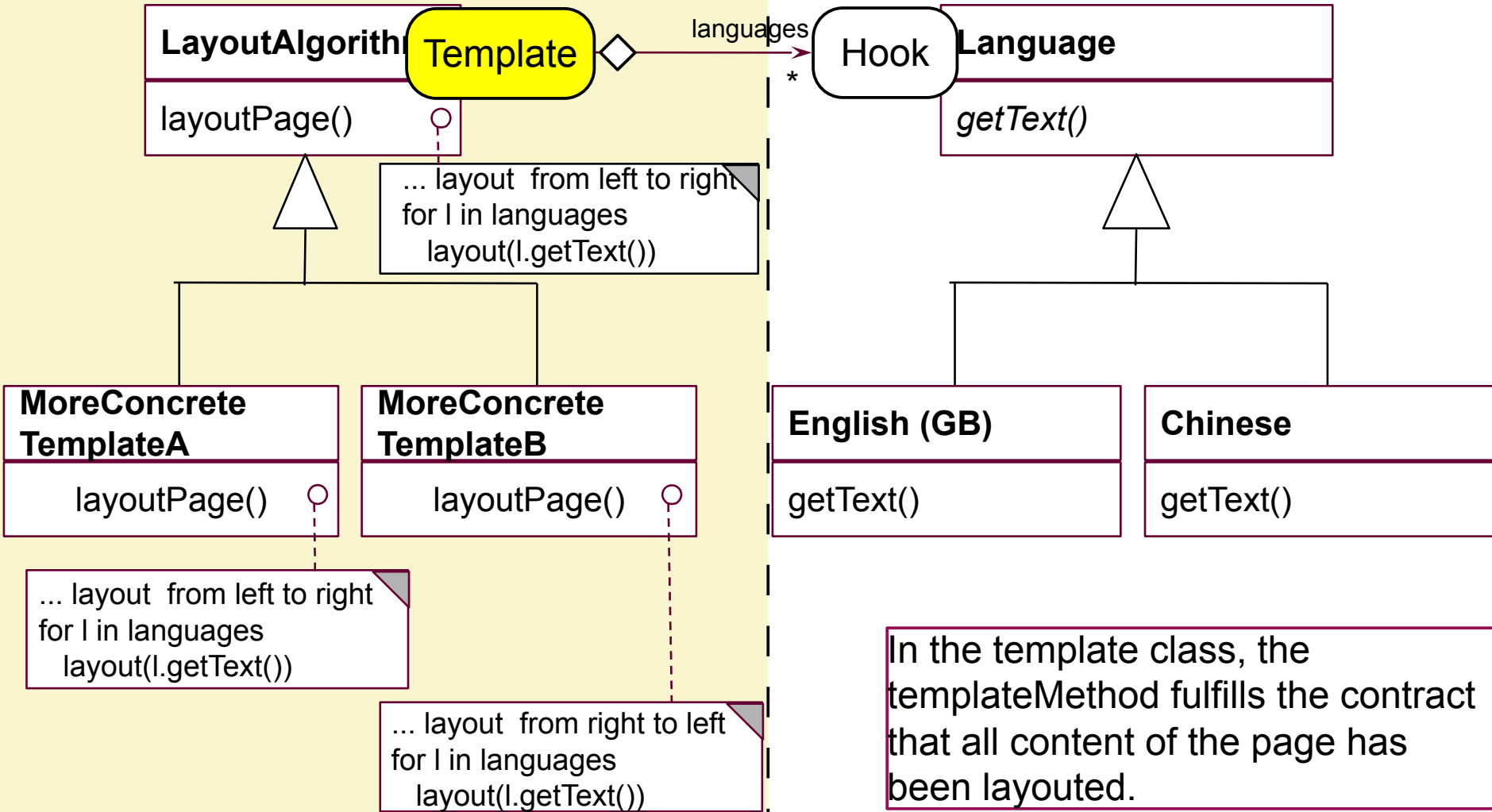
▶ may be abbreviated to:





# Multiple Internationalization as Dimensional Class Hierarchy with n-T--H

## Framework



# Multiple Internationalization as Dimensional Class Hierarchy with n-T--H

34

- ▶ n-T—H is based on \*-Bridge pattern
- ▶ This framework hook allows for multiple internationalized texts
  - An application can layout several languages at the same time
- ▶ The layout algorithm can be coupled with different languages that use the same layout (multiple internationalization)
- ▶ However, mixin of different layout languages freely with languages is impossible!
- ▶ Here, you can see the power of the T—H concept:
  - 1-T--H: dynamic variability
  - n-T—H: dynamic extension (flat, non-recursive)

# Multiple Internationalization as n-T—H Dimensional Hierarchy

Framework

**LayoutAlgorithm**

layoutPage()

Template

languages

Hook

**Language**

*getText()*

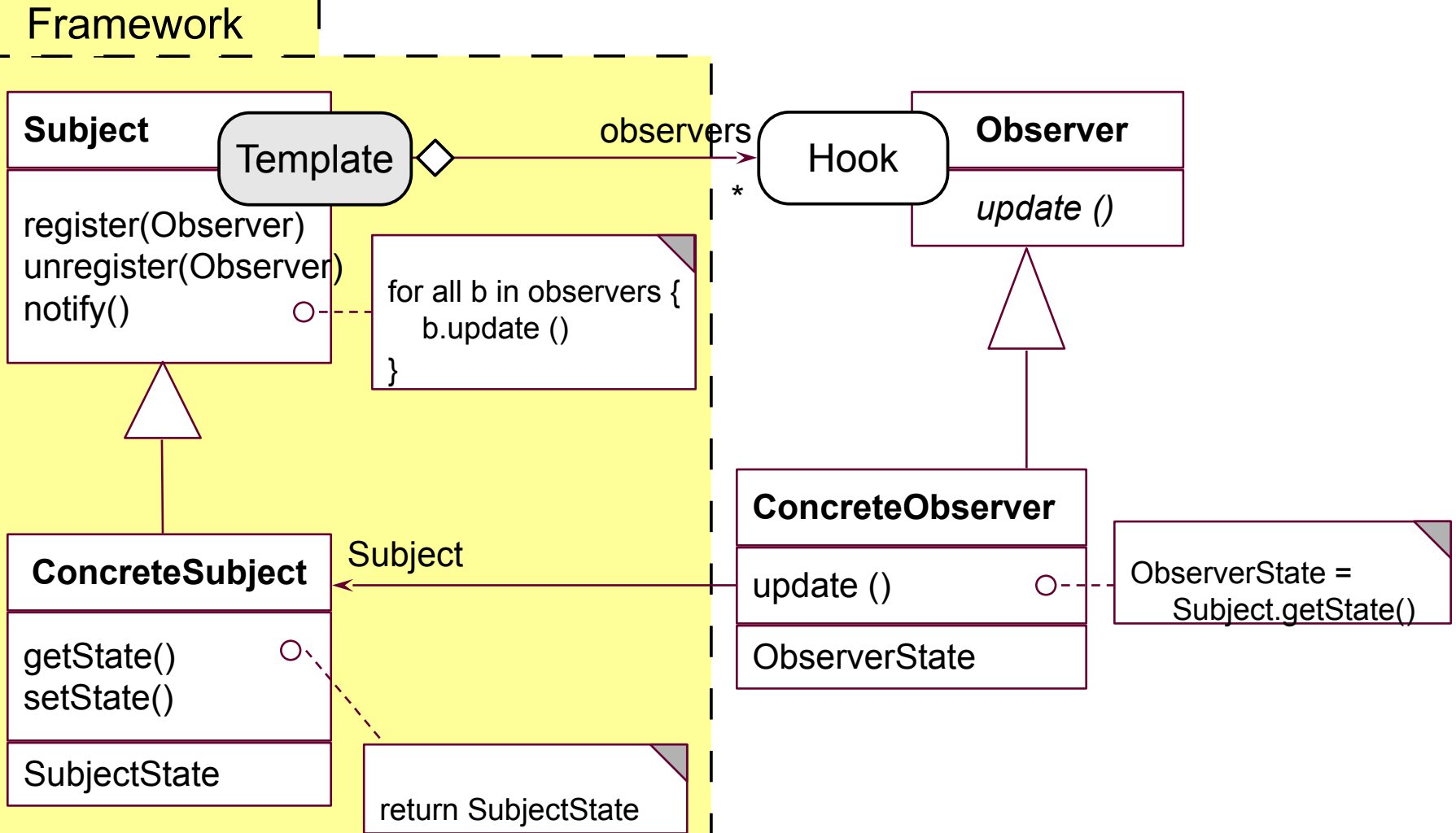
\*

Framework

**LayoutAlgorithm**

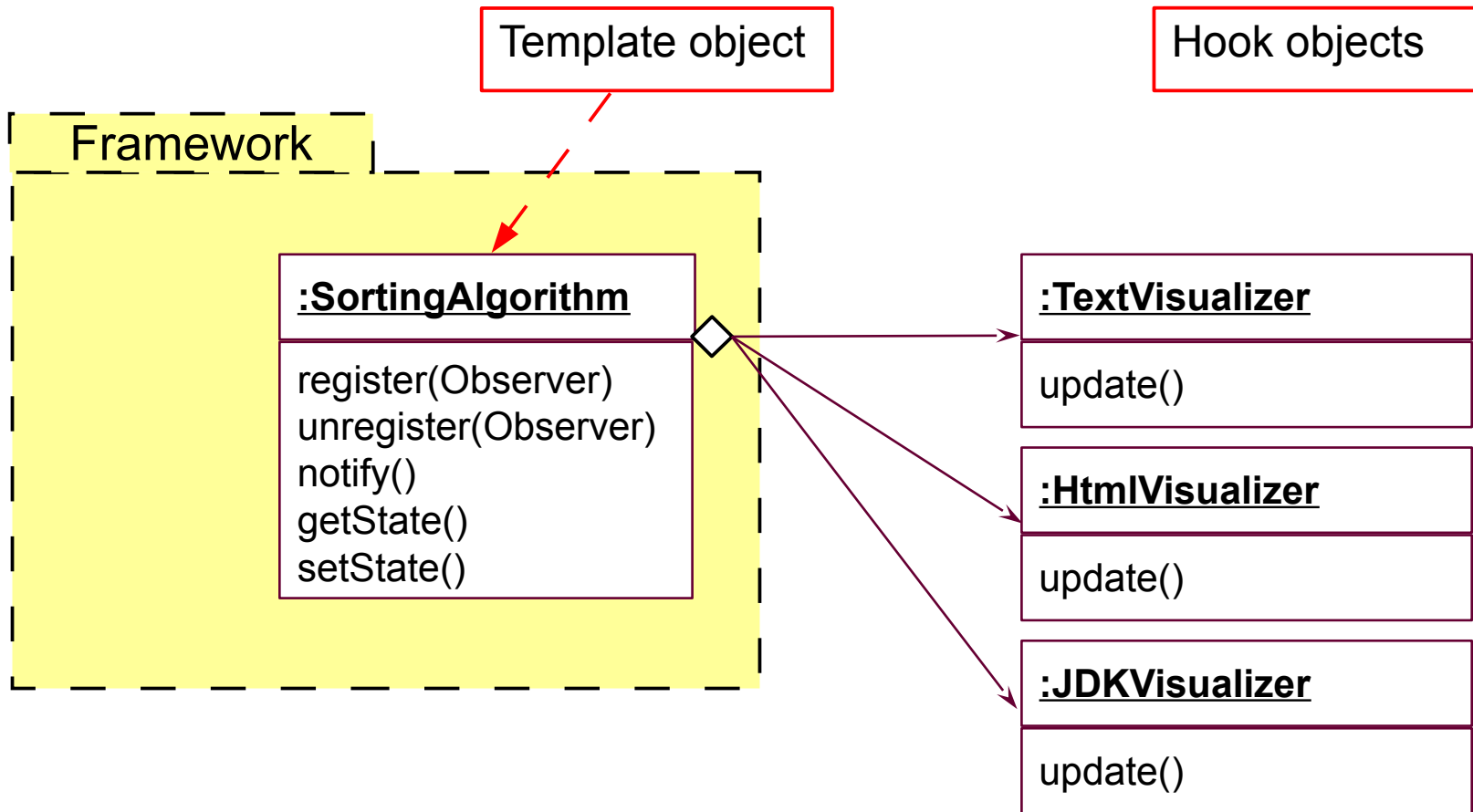
\* **Language**

# Observer as n-T—H of a Framework

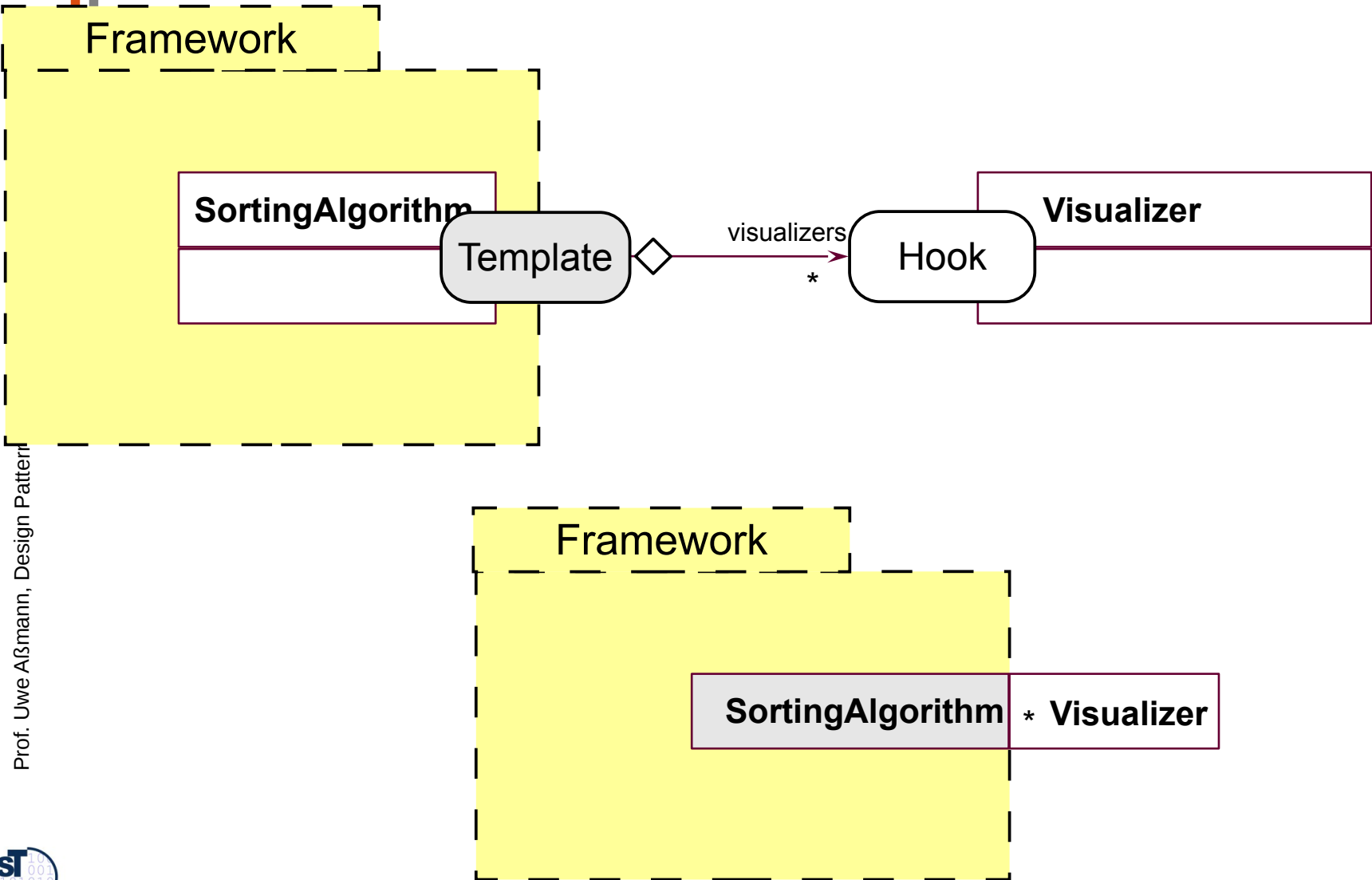


# Observer Runtime Scenario: Several Visualizers in Parallel

37



# Observer-based Extensible Frameworks



# Observer

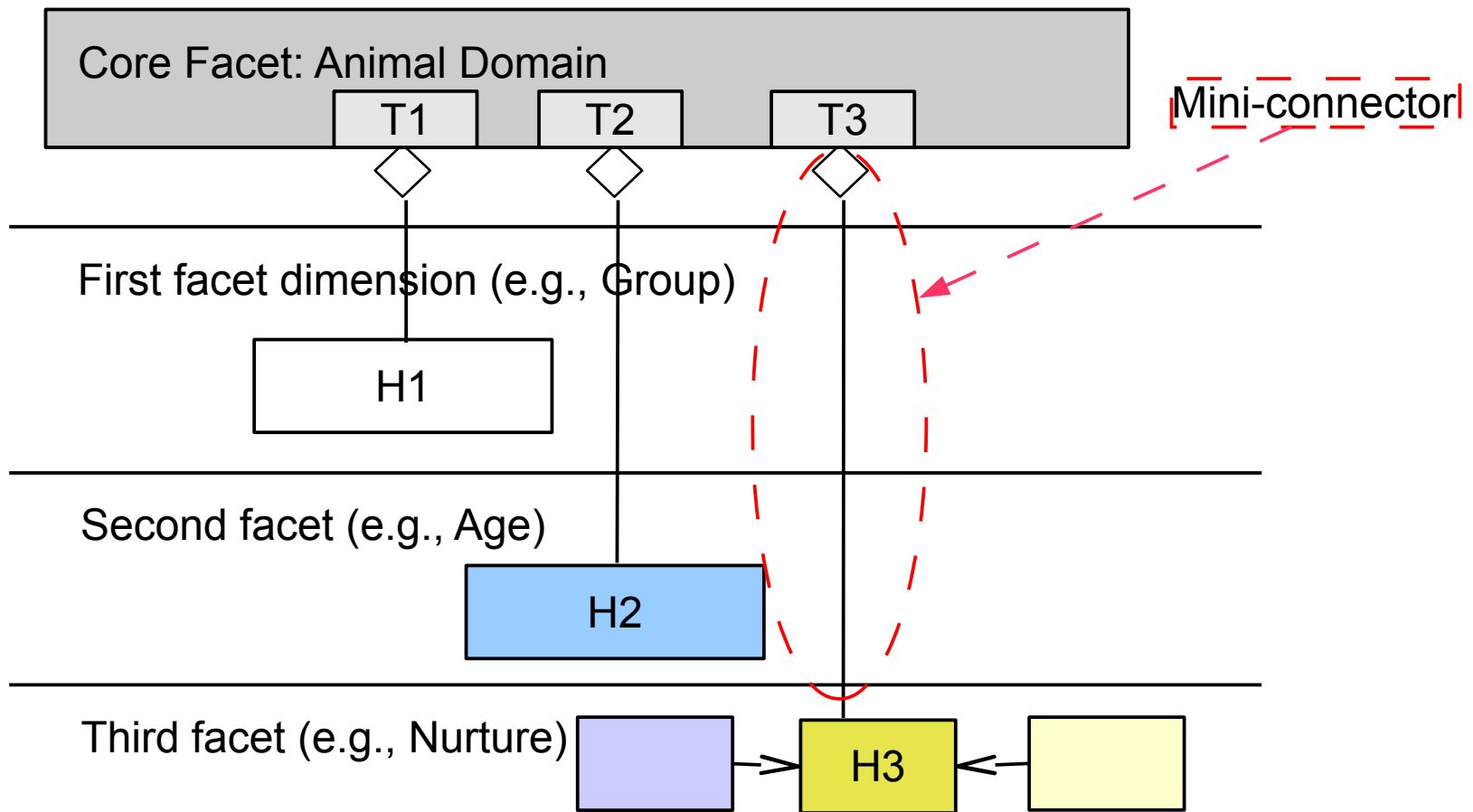
39

- ▶ The Observer pattern is used for extensibility
- ▶ With T&H, it becomes clear that Observers are a perfect way to achieve product lines with new feature extensions:
  - Model a critical template algorithm as Subject (template of the n-T--H)
  - Model an extension as a new Observer (hook of the n-T--H)

# Bridge Frameworks Have T—H Hooks

40

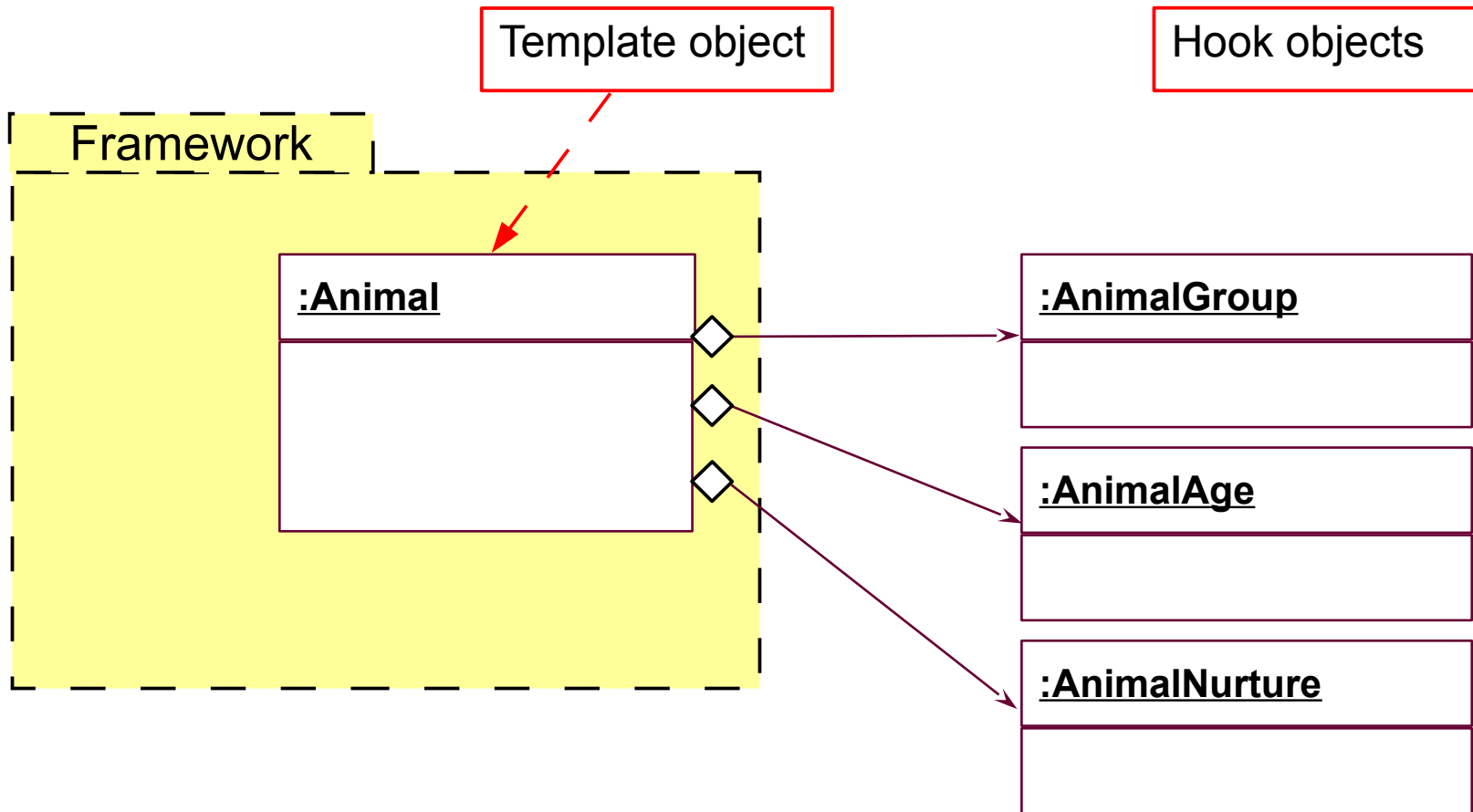
- ▶ Every dimension corresponds to a T—H hook
- ▶ Bridges, Strategy, Adapter can be used as mini-connectors





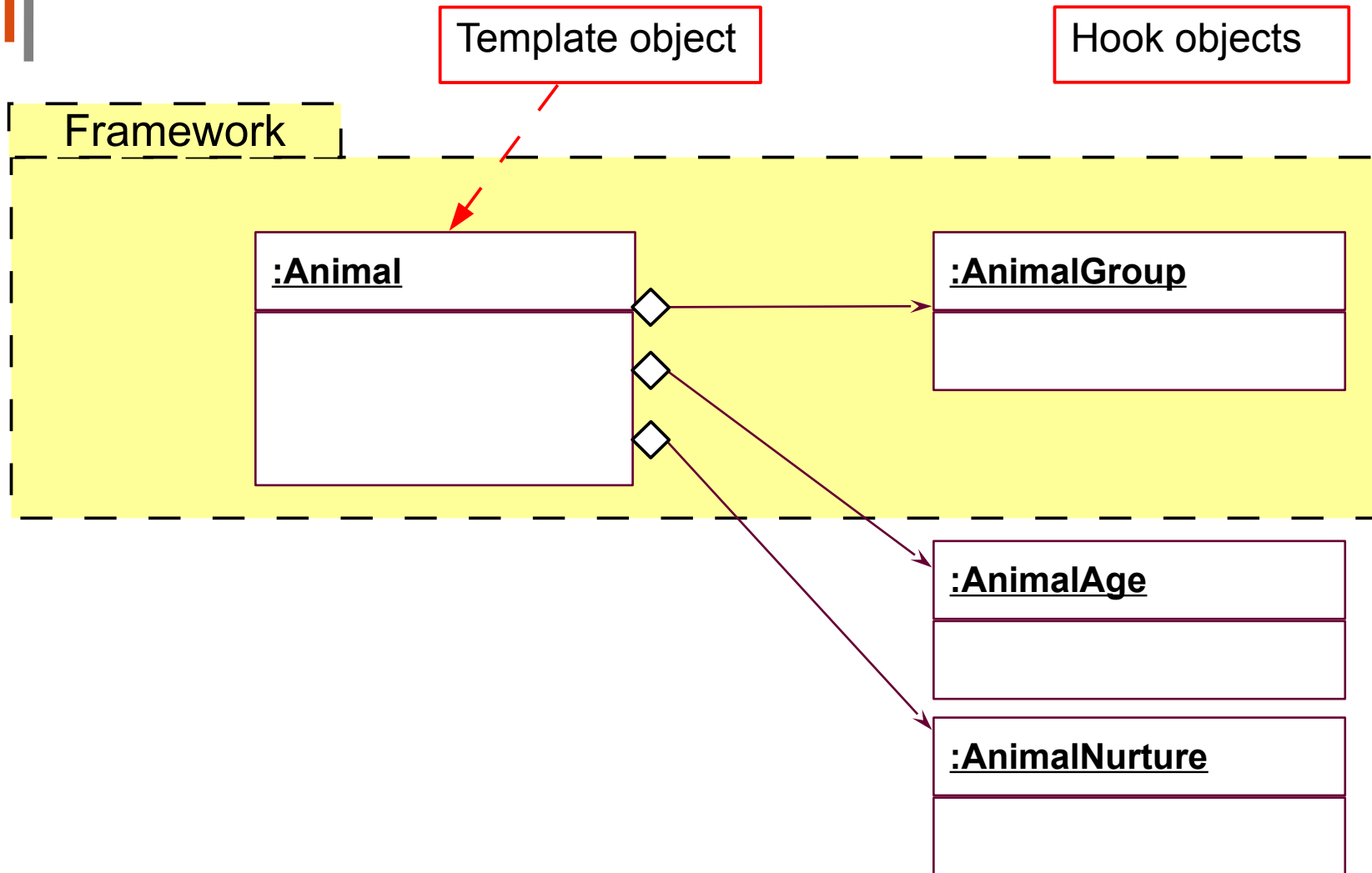
# Bridge Framework Runtime Scenario

41



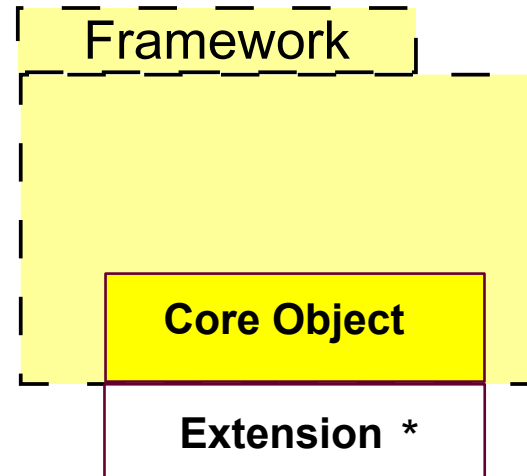
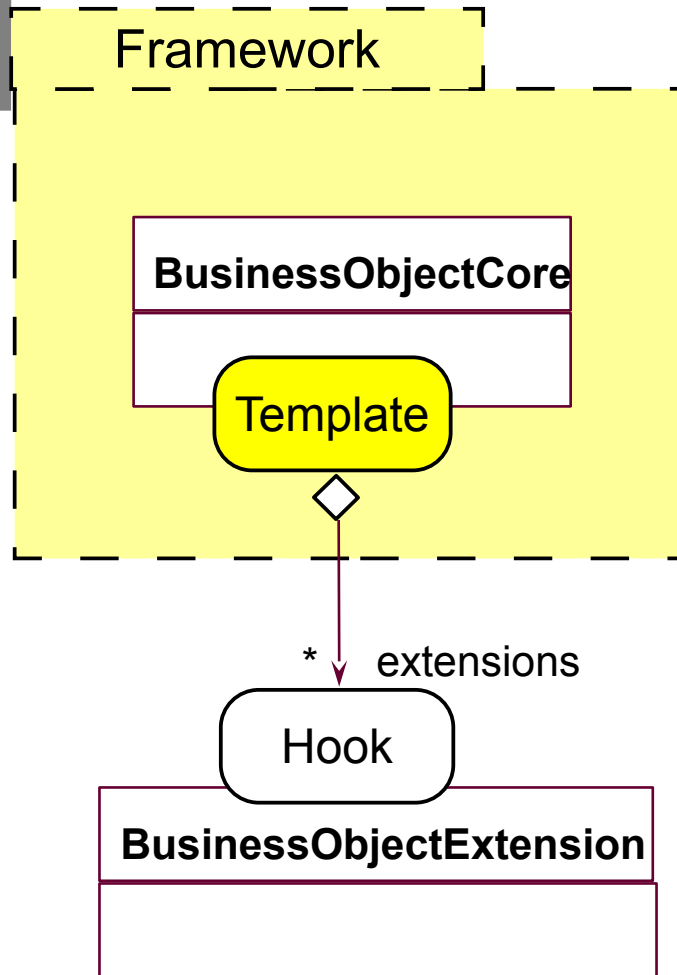
# Bridge Framework Runtime Scenario, with dimension 1 in Framework

42



# Extensible Bridge Framework with n-T--H

43



# n-T—H Makes Bridge Frameworks Extensible


44

- ▶ An n-T—H framework hook makes dimensional bridge frameworks extensible with new dimensions *at run time*
- ▶ New extensions in new dimensions can be added and removed on-the-fly
- ▶ Applications
  - Business applications
  - System software
  - 3- and n-tier architectures

# T—H Patterns Result in Blackbox Frameworks

45

- ▶ The main relation between T and H is *delegation*.
- ▶ Hence, when overriding and instantiating H, the framework is untouched (*blackbox framework*)
- ▶ 1-T—H gives variability
- ▶ n-T—H gives extensibility



# 11.4 The $H \leq T$ Recursion Metapattern

---

---

46



# H<=T Recursive Connection

47

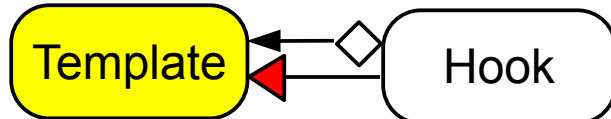
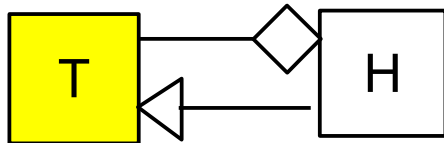
- ▶ T&H *recursive connection pattern* (H<=T framework hook, *deep extension pattern*)
  - with 1- or n-ObjectRecursion
  - H-class inherits from T; T is part of H
  - H is decorator of T (1:1) or a composed class in a composite pattern (1:n)

## H<=T (deep list extension)

T part of H

H inherit from T

1-ObjectRecursion/Decorator

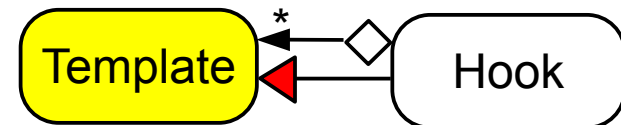
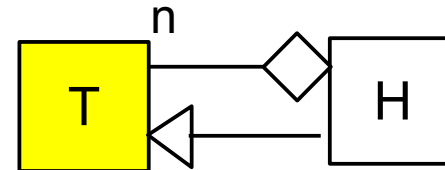


## n-H<=T (deep graph extension)

H has n T parts

T inherit from H

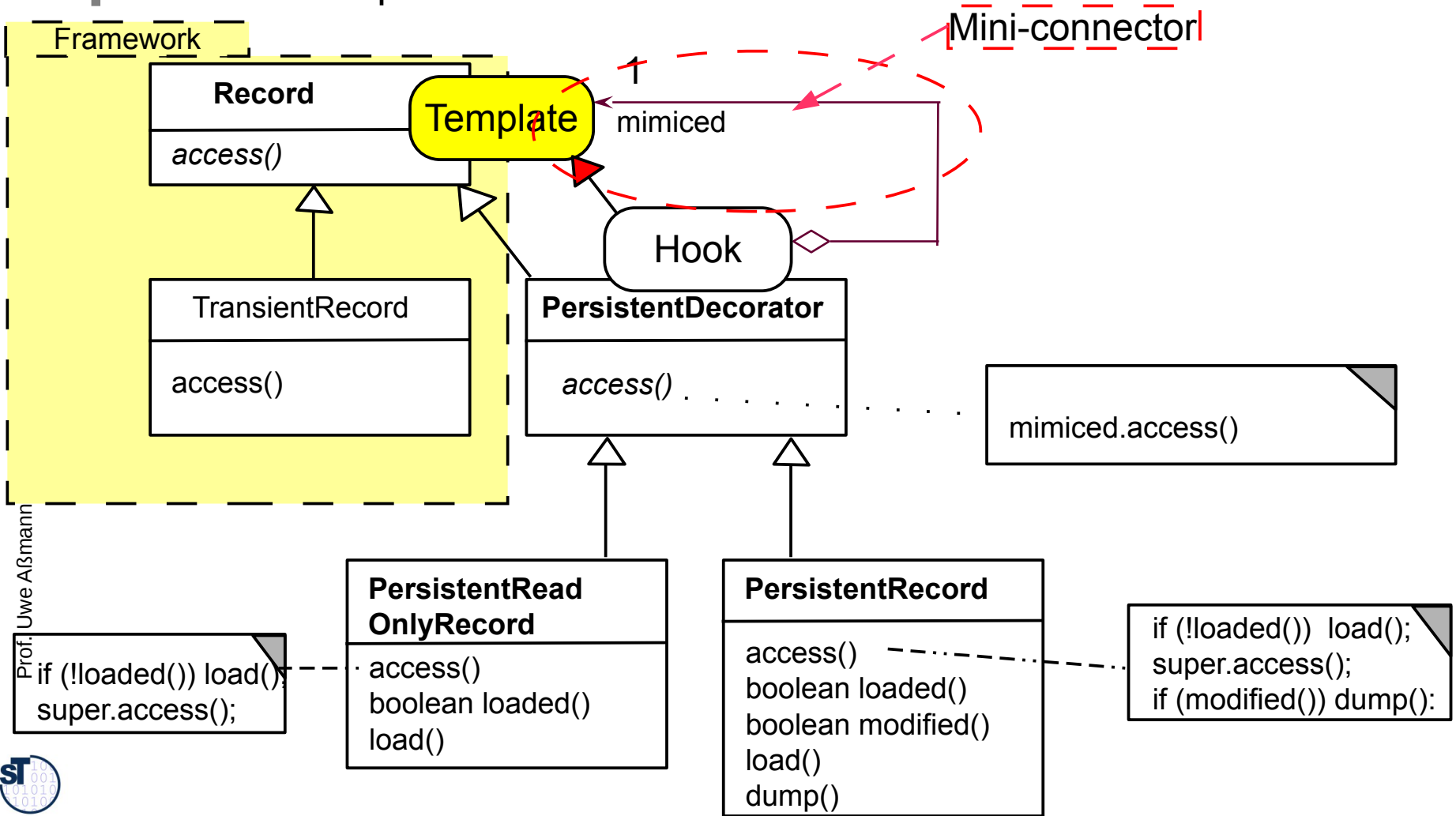
n-ObjectRecursion/Composite



# Decorator as 1-H<=T

48

- ▶ All decorator objects have to conform to the template class of the Decorator pattern

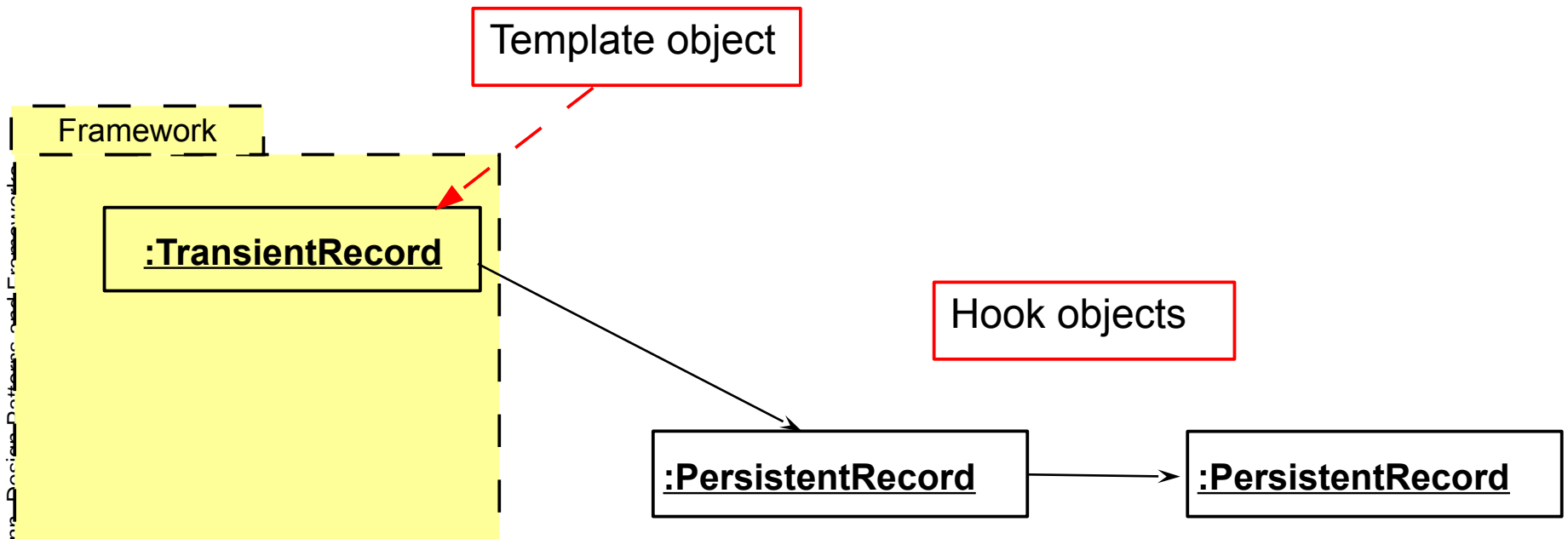




# Decorator as Framework Hook Pattern

49

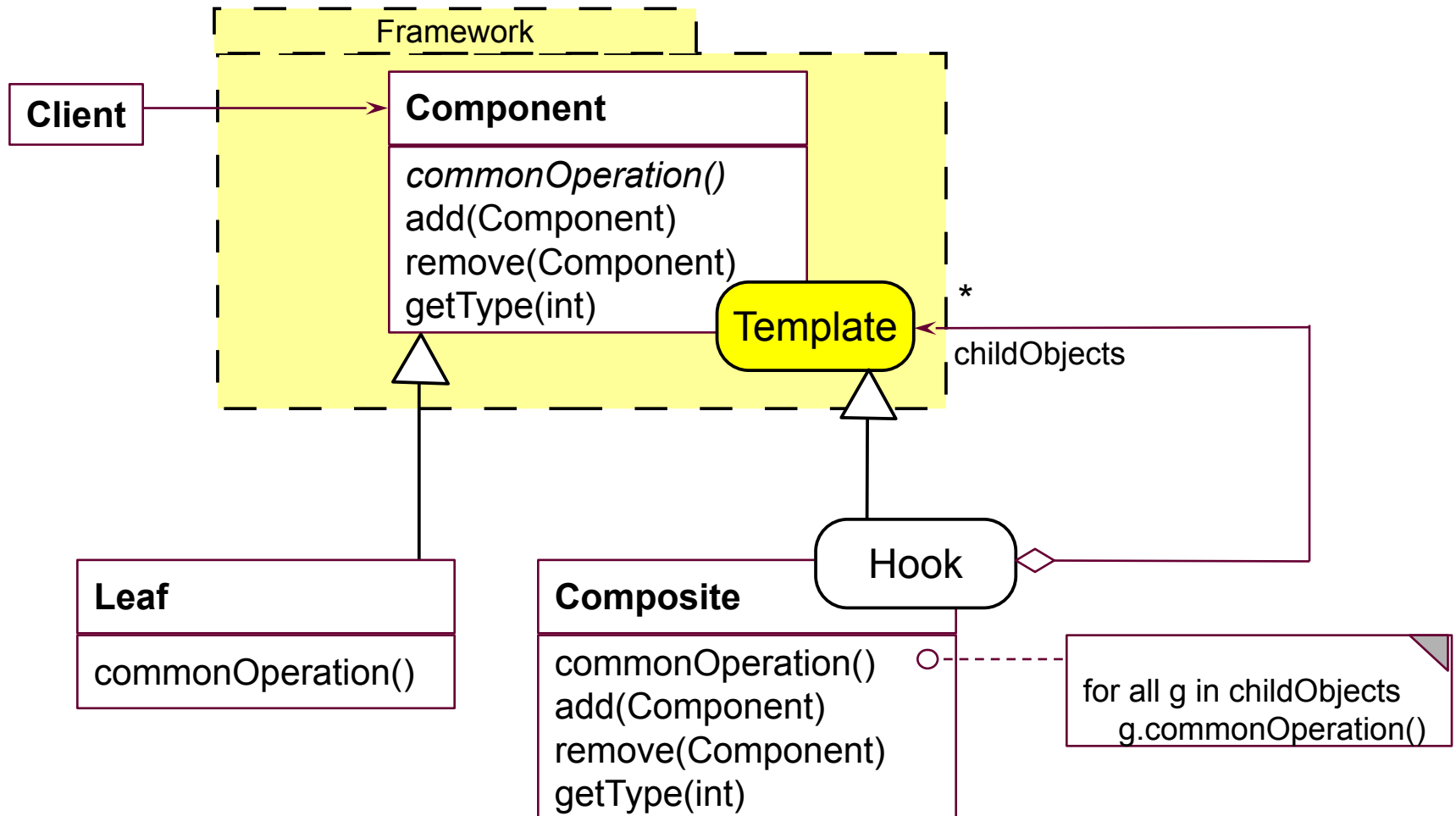
- ▶ Lists extend the framework



# Composite as $n-H \leq T$

50

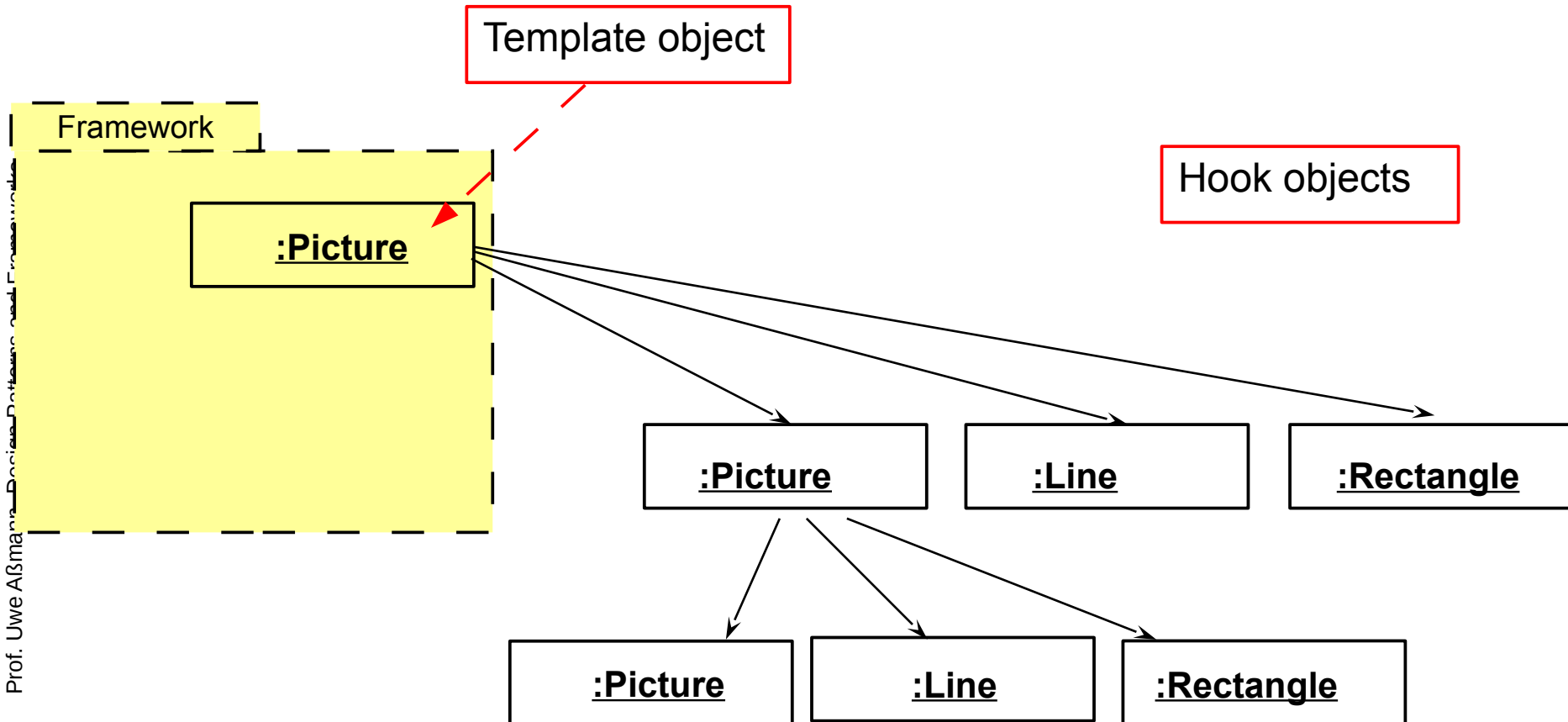
- ▶ Composite is as instance of n-ObjectRecursion and  $n-H \leq T$



# Composite as Framework Hook Pattern

51

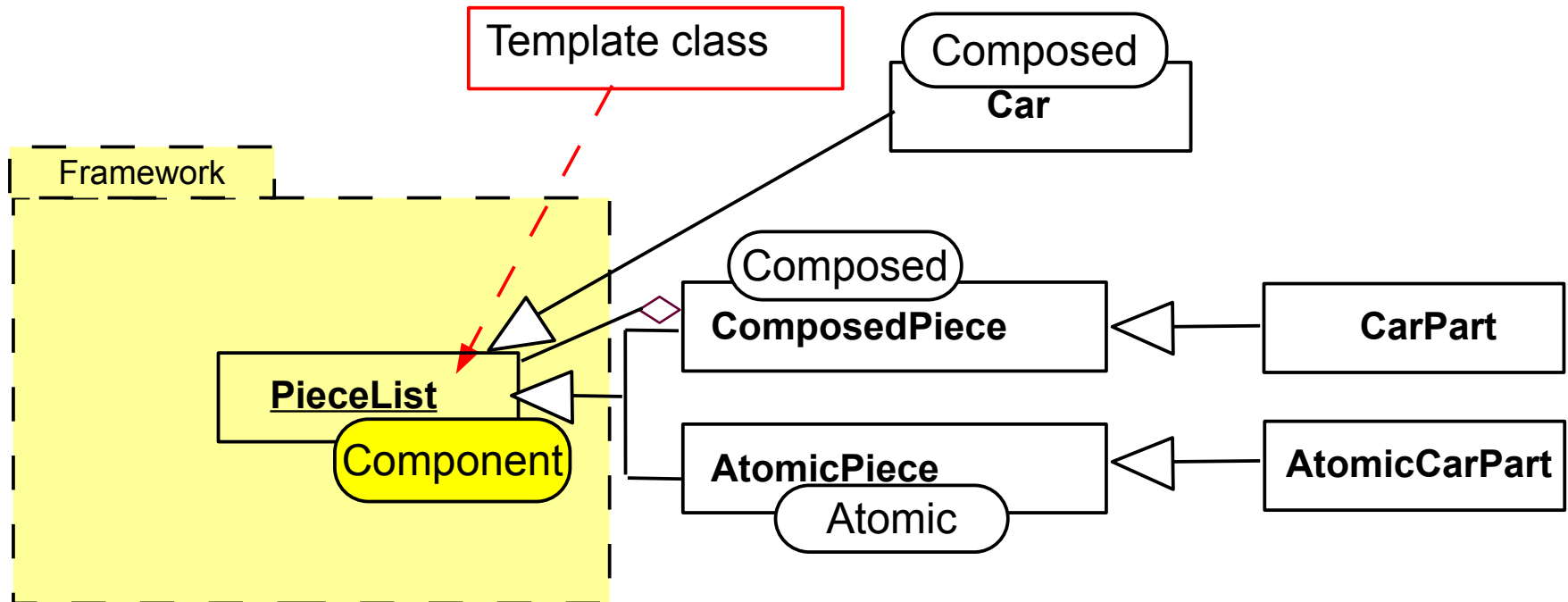
- ▶ Part/Whole hierarchies extend the framework



# Production Data Systems

52

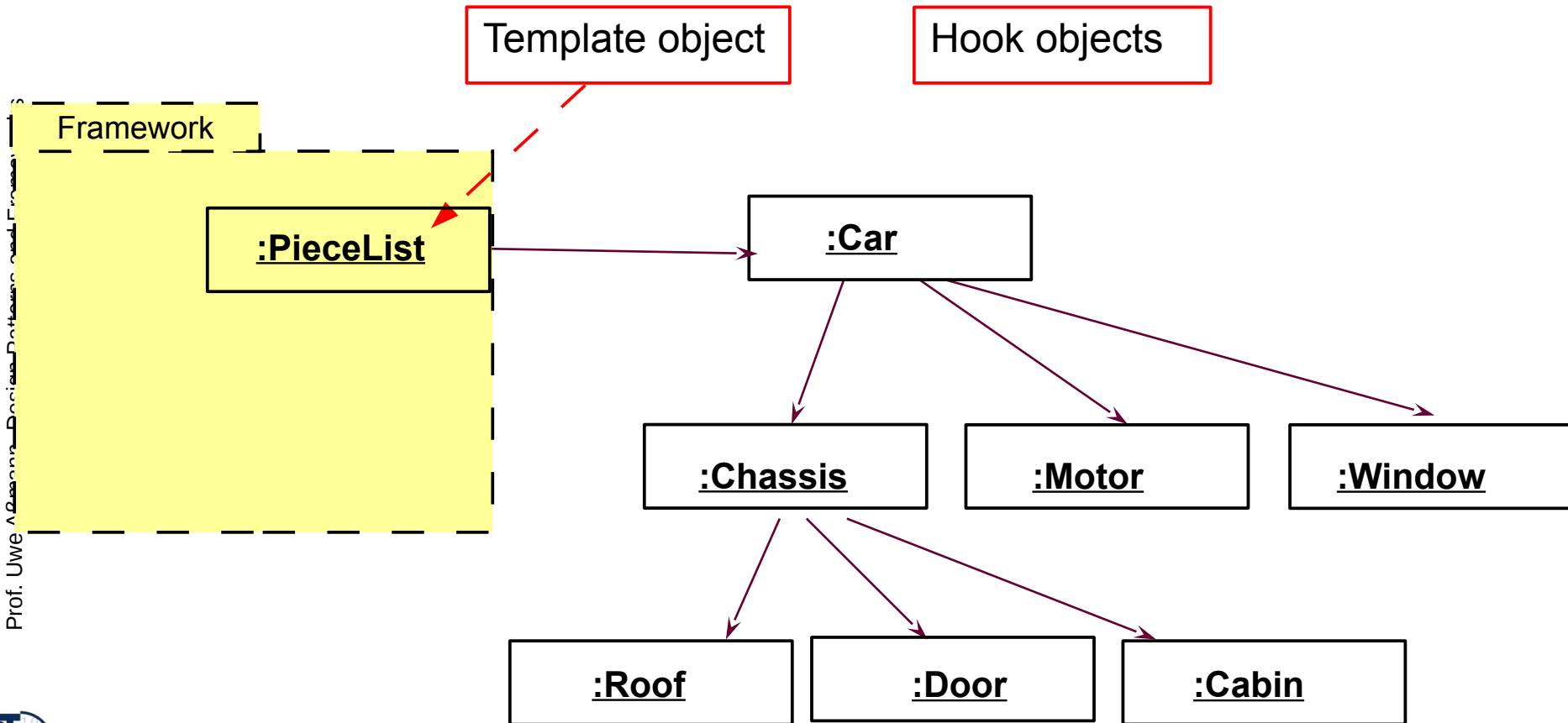
- ▶ Piece lists are part/whole hierarchies of technical artefacts in production
- ▶ The roles of a composite form the hook of the framework



# Production Data Systems

53

- ▶ Piece lists are part/whole hierarchies of technical artefacts in production
- ▶ Example: SAP PDM module, IBM San Francisco



Prof. Uwe Albrecht, Business Process and Process



# H<=T

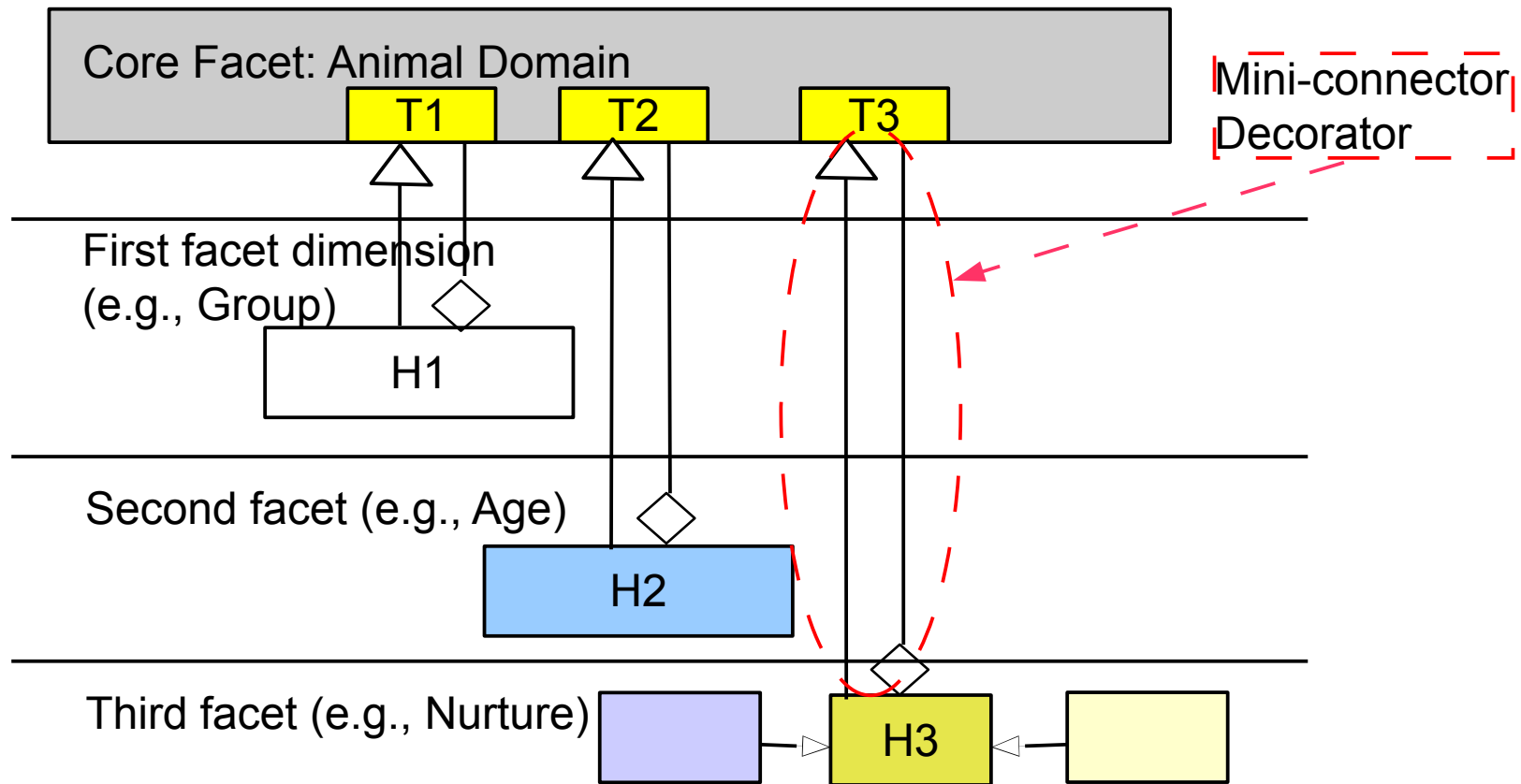
54

- ▶ H<=T framework hooks result in frameworks between black-box and white-box
- ▶ Mini-connector H<=T is used
- ▶ Attention: The class with the Template role carries the HookM role, the class with the Hook role carries TemplateM role
  - The template (fixed) class in the framework is called from the hook class in the application (which carries the template method role)
  - Pree calls the pattern T<=H, but means TemplateM <= HookM !!

# Bridge Frameworks Can Be Done with $H \leq T$ (Bridge $H \leq T$ Framework)

55

- ▶ A dimension may correspond to a  $H \leq T$  hook of the core framework
- ▶ Composite, Decorator, Bureaucracy can be used as mini-connectors

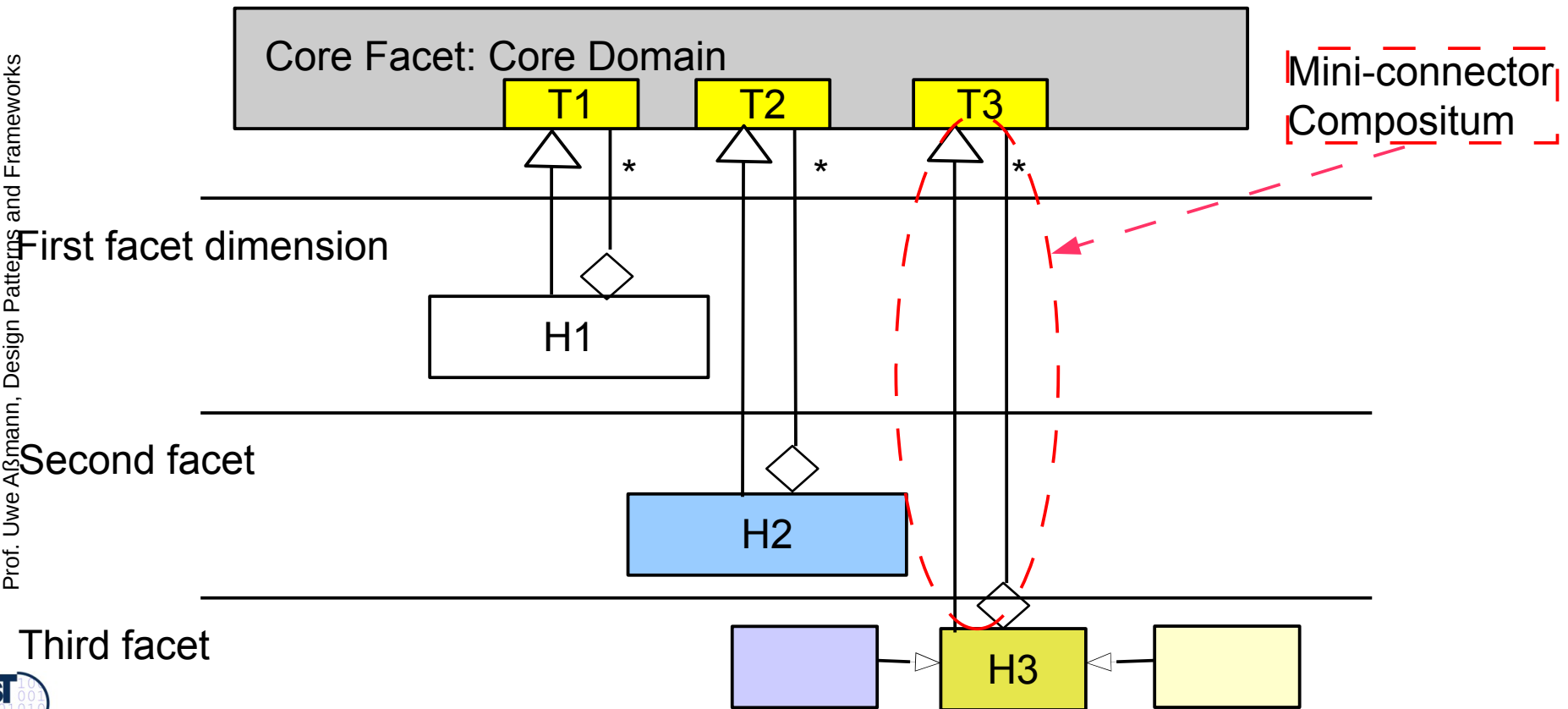


# Bridge Frameworks Can Be Done with $H \leq T$ (Bridge $H \leq T$ Framework)

56

- ▶ Composite as mini-connector

Prof. Uwe Alßmann, Design Patterns and Frameworks







# 11.5 The TH Unification Metapattern

57

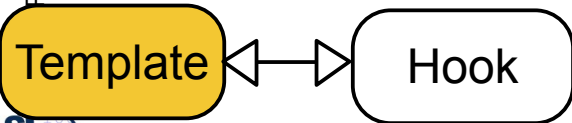


# TH

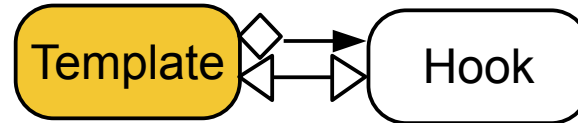
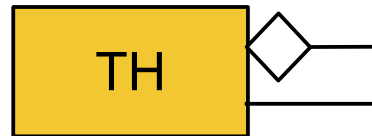
58

- ▶ Unified T&H pattern (TH framework hook)
  - T-class == H-class

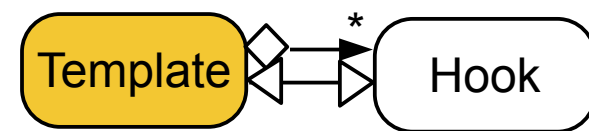
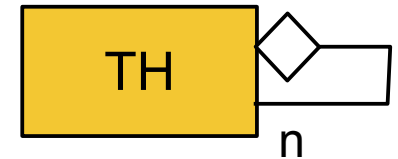
**TH**  
T == H  
TH part of TH  
“funny” Decorator



**1-TH (deep list extension)**  
T == H  
TH part of TH  
“funny” Decorator



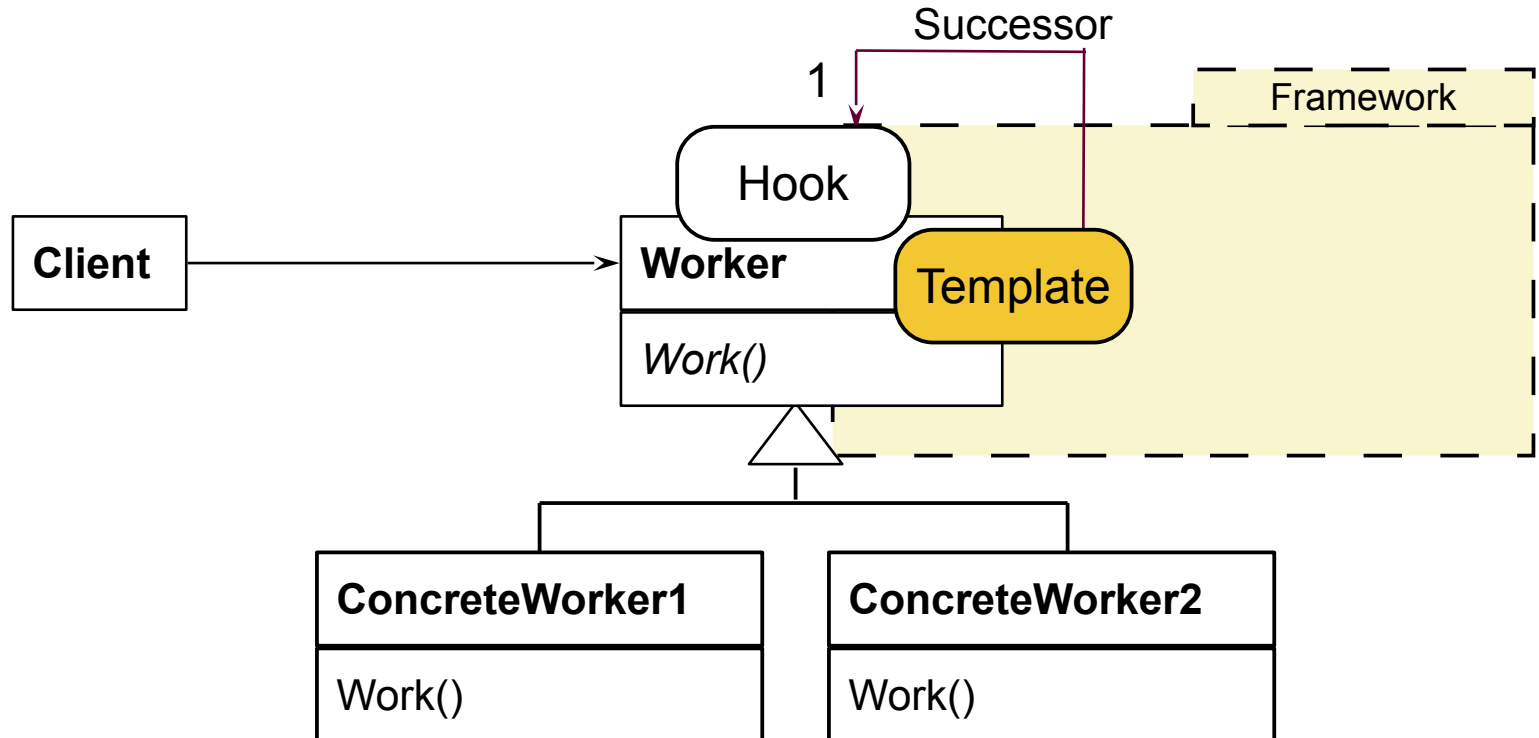
**n-TH (deep tree extension)**  
T == H  
TH has n TH parts  
“funny” 1:n-Composite



# ChainOfResponsibility as 1-TH

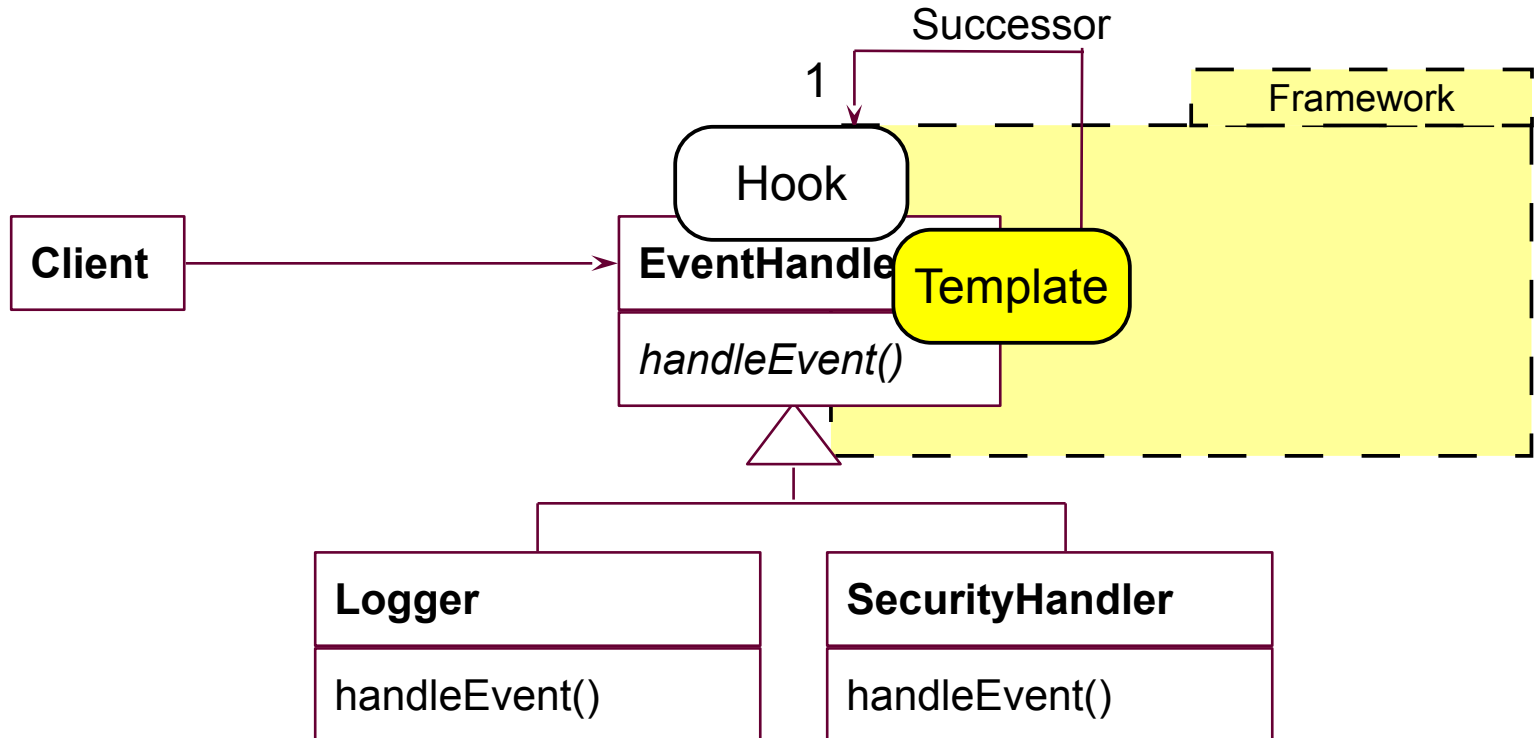
59

- ▶ A Chain is recursing on the abstract super class, i.e.,
  - All classes in the inheritance tree know they hide some other class (unlike the ObjectRecursion)



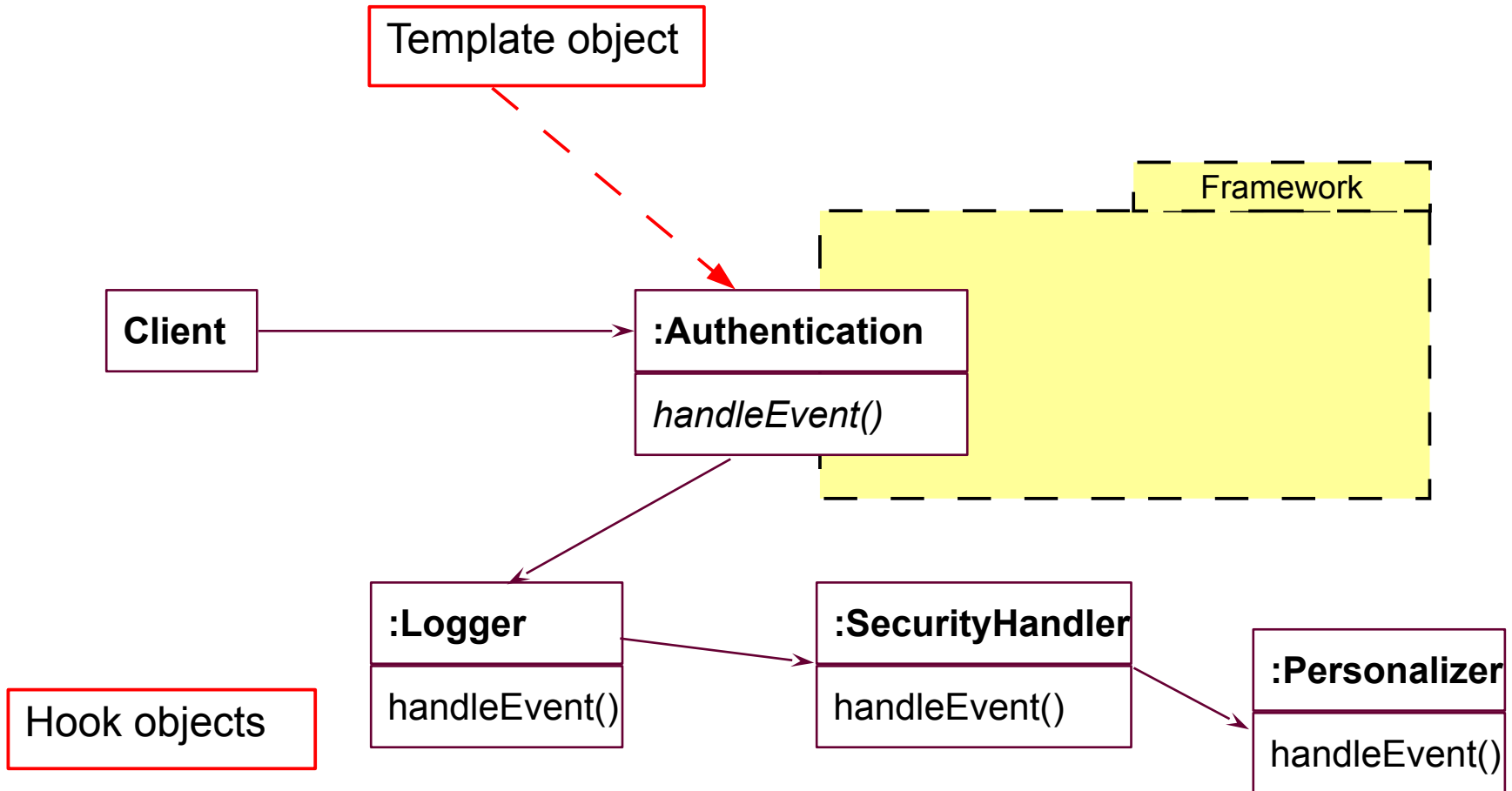
# Event Handlers

60



# Event Handlers: Object Diagram

61



# Why TH Unification Makes Sense

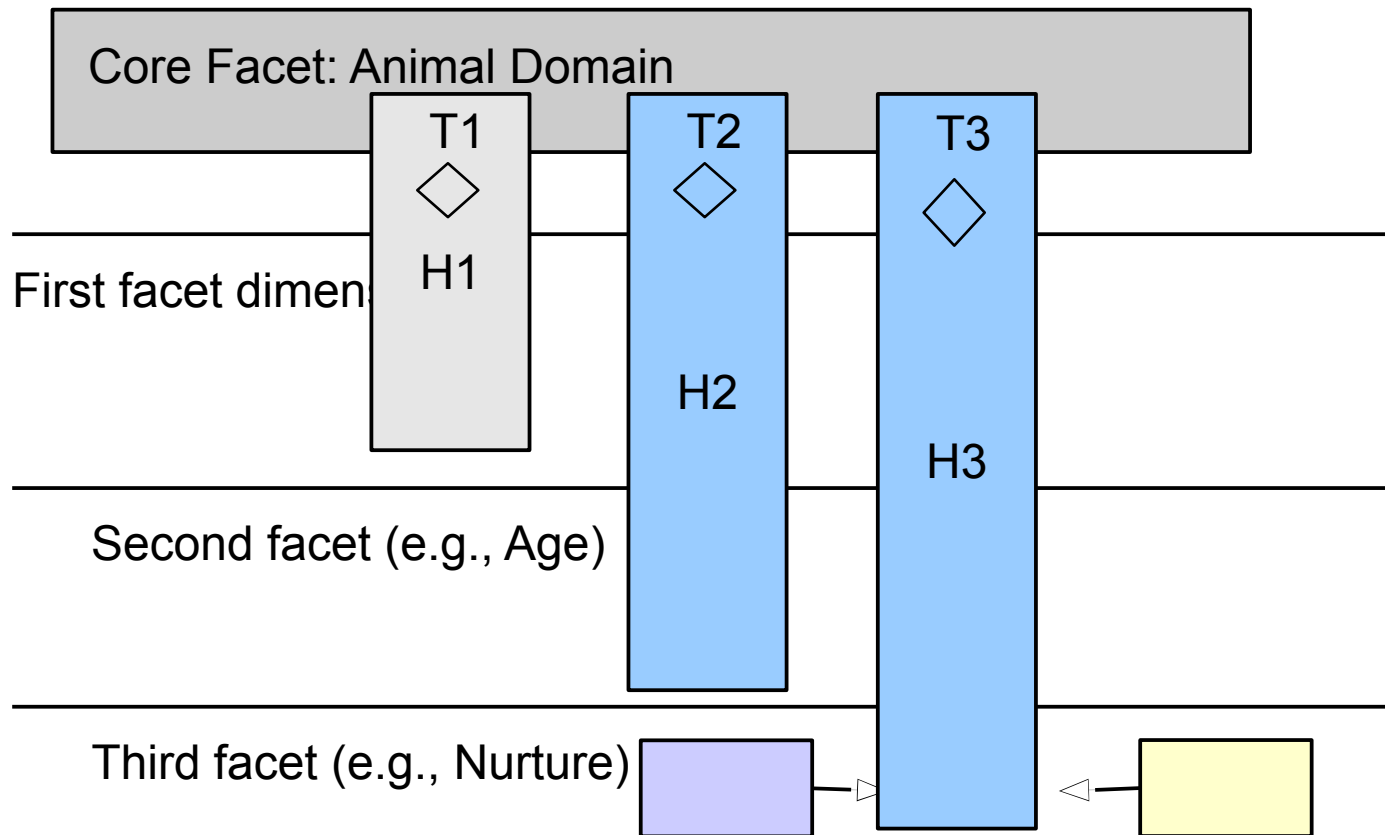
62

- ▶ If a hook class is the same as the template class,
  - Some methods are template methods, others are hook methods
  - Together with the template, the hooks can be exchanged
- ▶ Template methods in the template class are not abstract, but concrete
  - They are build from referencing hook methods of the hook class
- ▶ As we saw in the last chapter, merging role types in one class can make an application faster, but less flexible

# Bridge Frameworks Can Be Done with TH (Bridge TH Framework)

63

- ▶ A dimension may correspond to a  $H \leq T$  hook
- ▶ Chain can be used as mini-connector





# 11.6 The H<T Whitebox Inheritance Metapattern

---

---

64



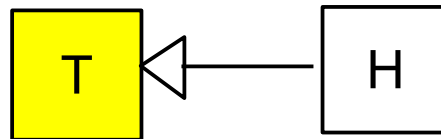


# H<T

65

- ▶ If H inherits from T, H<T framework port (whitebox framework pattern)
  - Whitebox reuse of T in the framework, while deriving H in the application
  - (not of Pree, earlier known)
- ▶ If a hook class inherits from a template class, it inherits the skeleton algorithm
  - Template methods in the template class are not abstract, but concrete
  - They are build from referencing hook methods of the hook class
- ▶ A H<T framework hook means whitebox framework

H<T

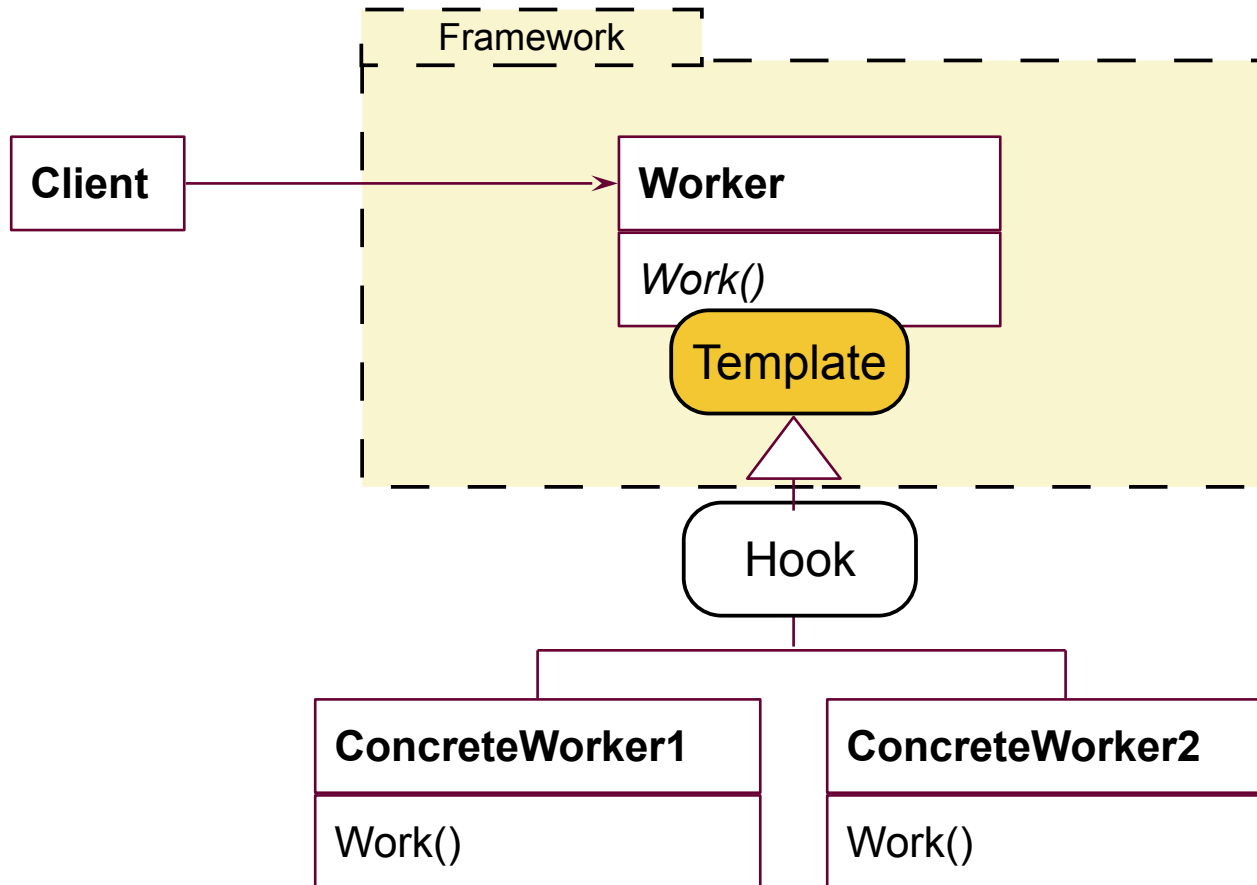


# Whitebox Framework with $H < T$

## Framework Hook

66

- ▶ Also TemplateMethod can be applied ( $\text{HookM} \leq \text{TemplateM}$ )



# Summary of T&H Patterns and Framework Hooks

67



# Cardinalities and Extensibility of Framework Hooks

68

- ▶ 1:1 – T and H correspond 1:1
  - T has 1 H part
  - Hooks are not extensible at runtime
  - 1:1 T&H framework hooks should be used when the behavior of the framework should be varied, but not extended at the variation point
    - Because variability patterns form the mini-connector between T and H, derived from 1-ObjectRecursion
- ▶ 1:n – T and H correspond 1:n
  - T has n H parts
  - Hooks are extensible, also dynamically
  - 1:n T&H framework hooks should be used when the behavior of the framework should not only be varied, but also *extended* dynamically at the variation point
    - Because extensibility patterns form the mini-connector between T and H, derived from n-ObjectRecursion

# Framework Hook Patterns

69

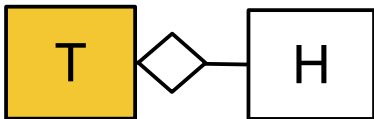
## Inheritance

## Unification

## Aggregation/Association

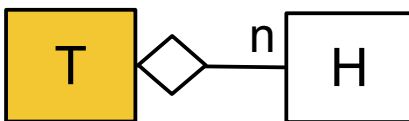
### T--H

H part of T  
T is core class of complex object



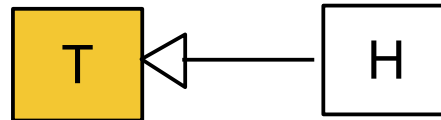
### n-T--H

T has n H parts  
T is core class of complex object



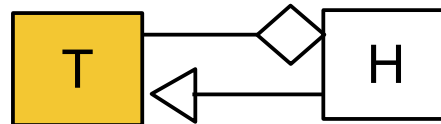
### H<T

H inherit from T  
whitebox



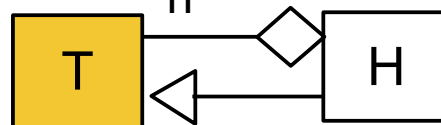
### H<=T

T part of H  
H inherit from T  
Decorator



### n-H<=T

H has n T parts  
H inherit from T  
1:n-Decorator



## Recursion

### TH

T == H



### 1-TH

T == H  
TH part of TH  
“funny” Decorator



### n-TH

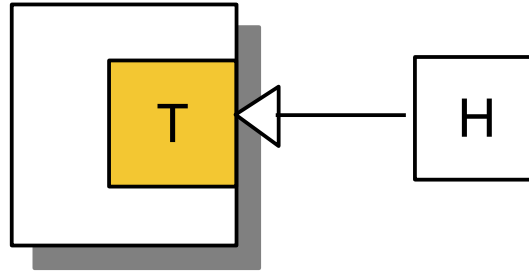
T == H  
TH has n TH parts  
“funny” 1:n-Composite



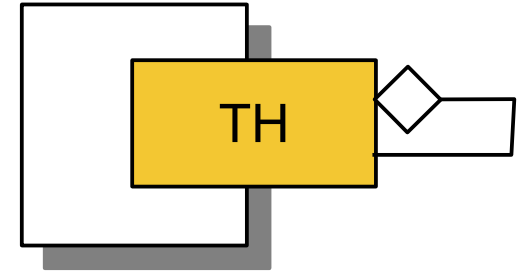
# Deriving a Simple Notation for Framework Hooks

70

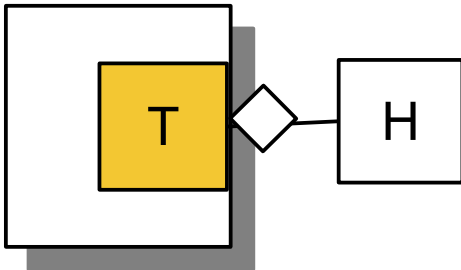
H<T



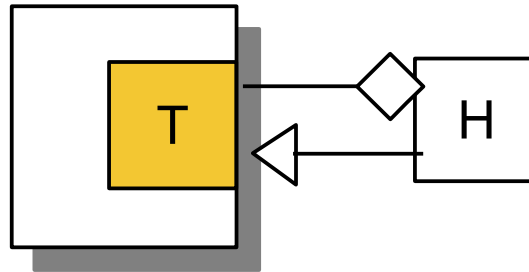
TH



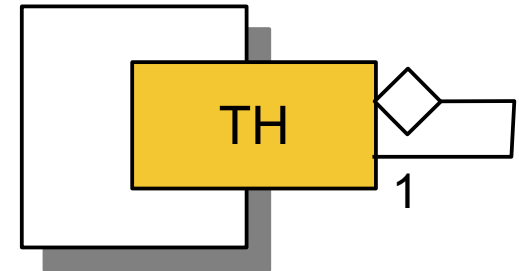
1-T--H



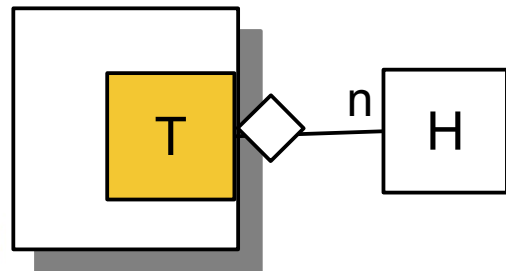
H<=T



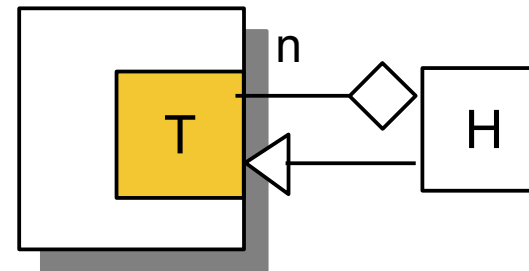
1-TH



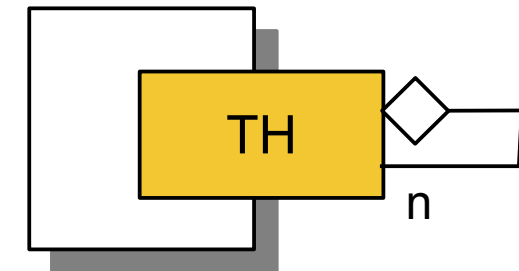
n-T--H



n-H<=T



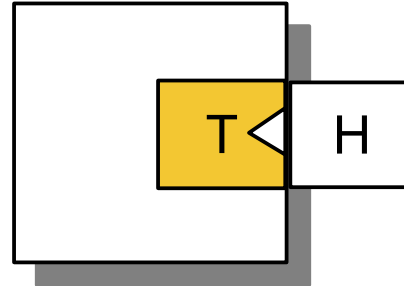
n-TH



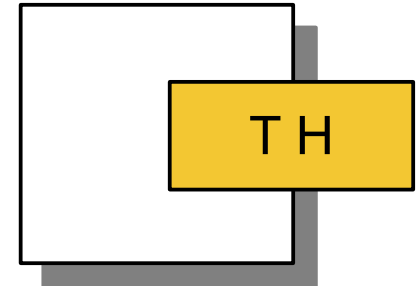
# Short-Hand Notation for Framework Hooks

71

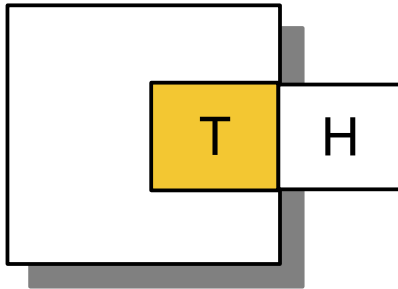
**H<T**



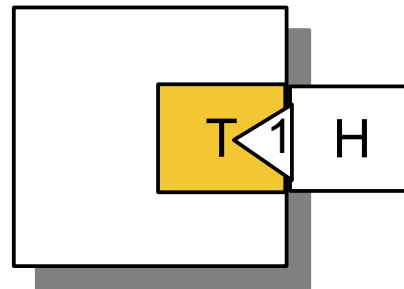
**TH**



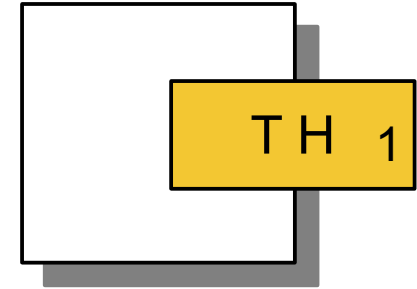
**1-T--H**



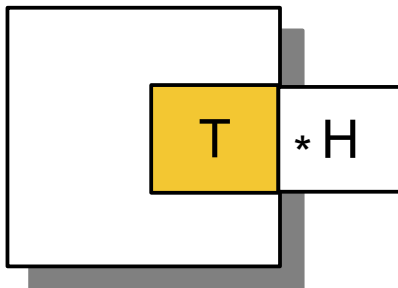
**H<=T**



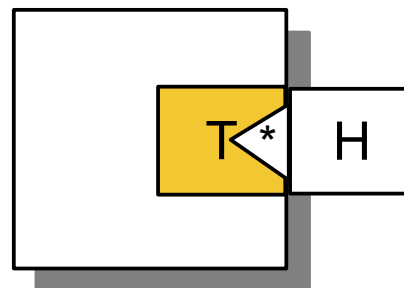
**1-TH**



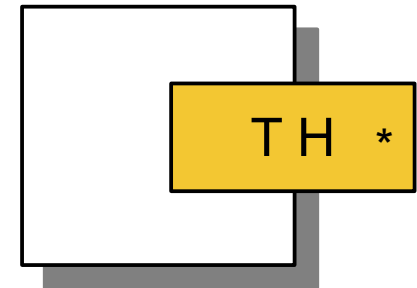
**n-T--H**



**n-H<=T**



**n-TH**





# 11.7 T&H in Frameworks

---

---

72





# Advantages of T&H Framework Hook Patterns

73

- ▶ One big mess with frameworks is the *trustworthy framework instantiation problem*:
  - If a framework is instantiated by inheritance (whitebox) or delegation (blackbox), illegal combinations of parameters appear
  - Applications may not run stabel
- ▶ Framework Hook Patterns describe much more precise *how* the variation points of a framework should be instantiated
  - They allow for determining whether the framework is *varied* or *extended* in a product line

# Pree's First Law of Framework Instantiation

74

- ▶ Variability-based framework hooks define *framework variation points*
  - If you want to constrain the uses of a framework to a fixed set of variations, use variability patterns for framework hooks (1-TH patterns)

**If a framework hook is based on a variability pattern,  
the framework is varied, but NOT extended**

# Pree's Second Law of Framework Instantiation

75

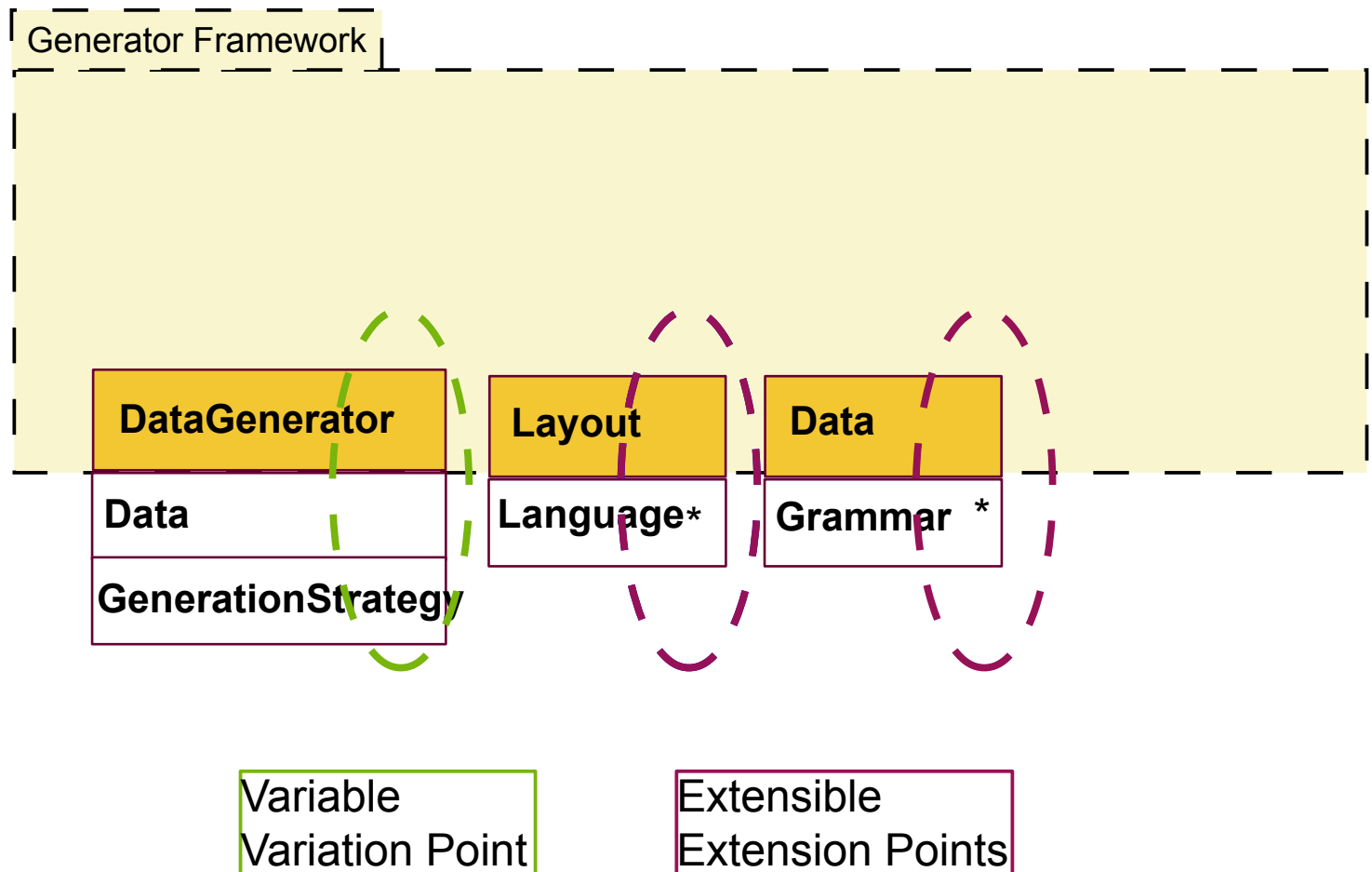
- ▶ Extensibility-based framework hooks define *framework extension points*
  - If you *do not want to constrain* the uses of a framework to a fixed set of variations, use extensibility patterns for framework hooks (n-TH patterns)

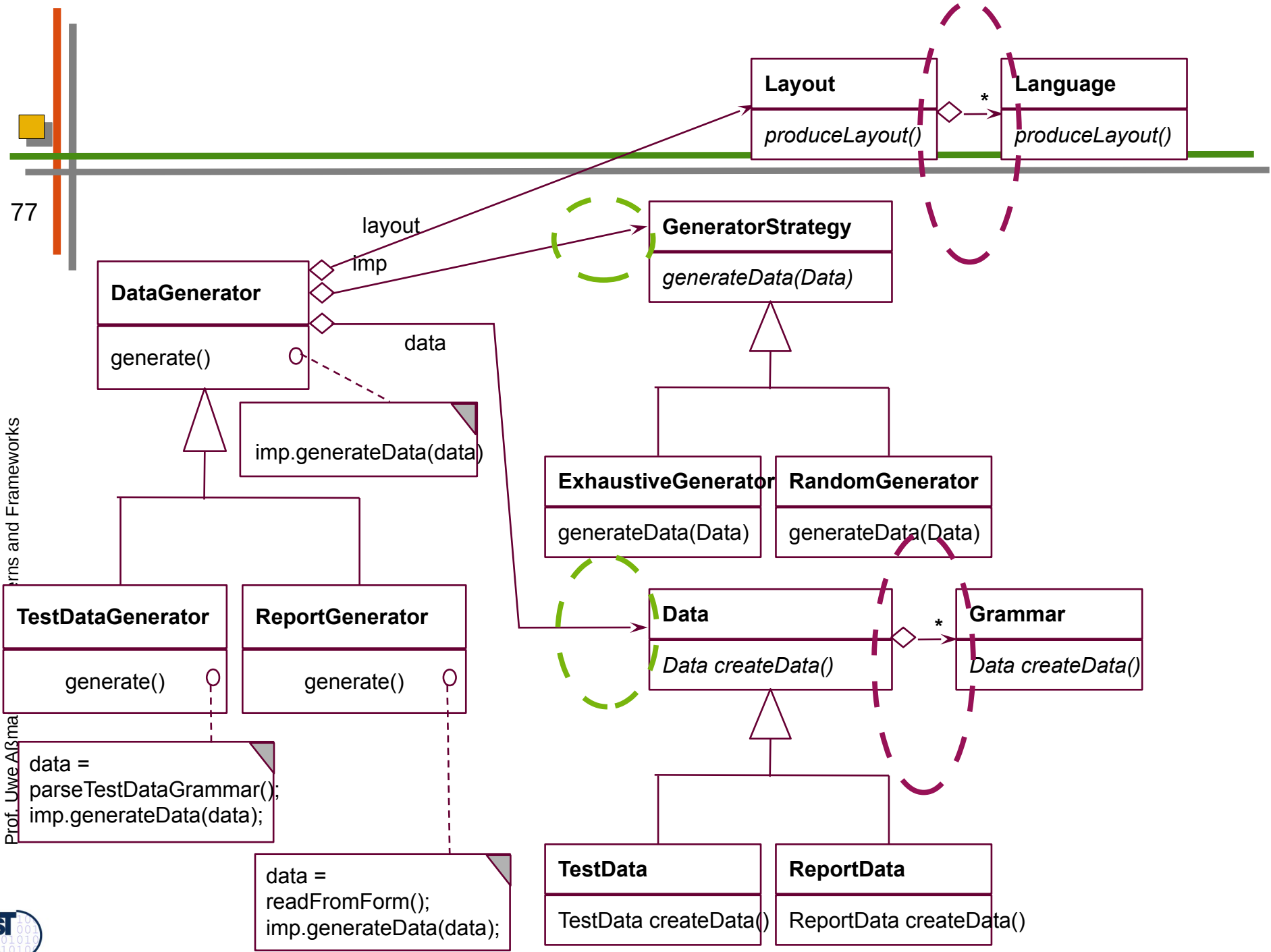
**If a framework hook is based on an extensibility pattern,  
the framework is extended, but not varied**

# A Multi-lingual dimensional Data Generator

76

- ▶ One framework hook may have several bridge dimensions





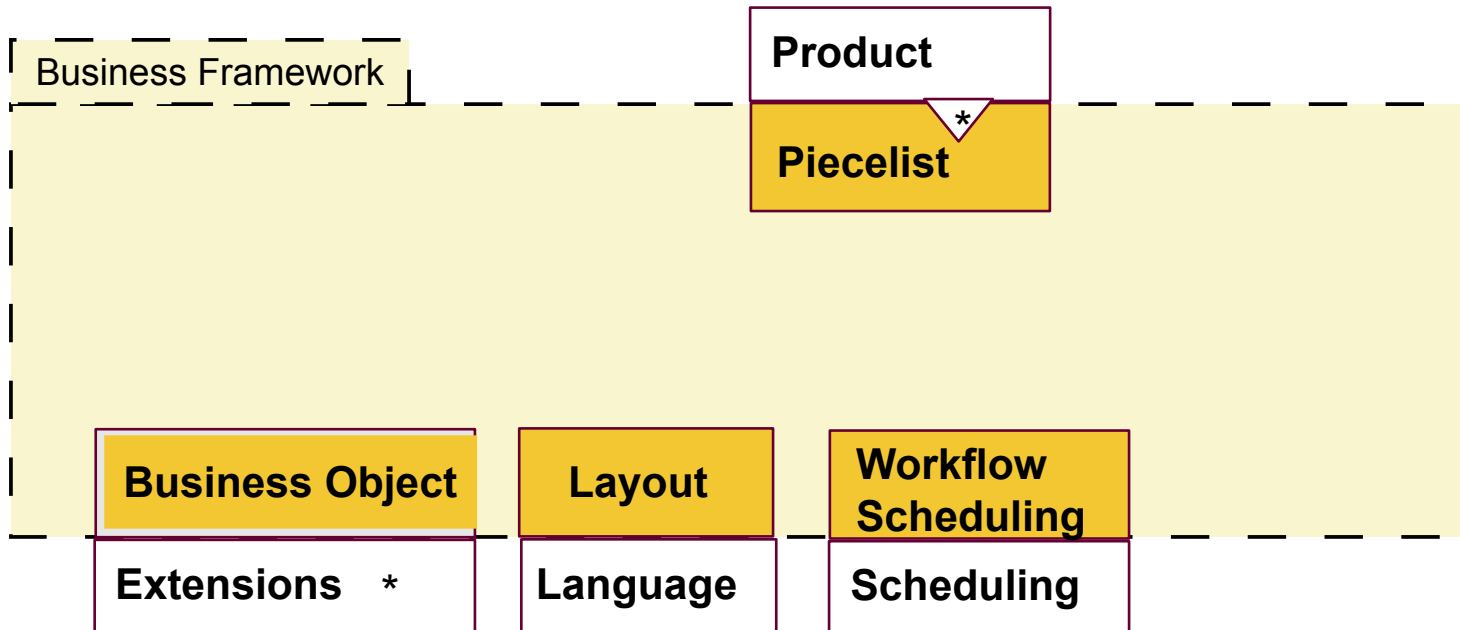
# Framework Instantiation Market

78

- ▶ Today, frameworks are the most important software technology for product lines in large companies
- ▶ Instantiating big frameworks is very hard
  - Requires special *instantiation consultancy*, which is a big market
  - SAP Germany has a marker for instantiation companies of their framework!
  - If you go to a big company, teach them framework instantiation patterns!

# A Multi-lingual Business Framework

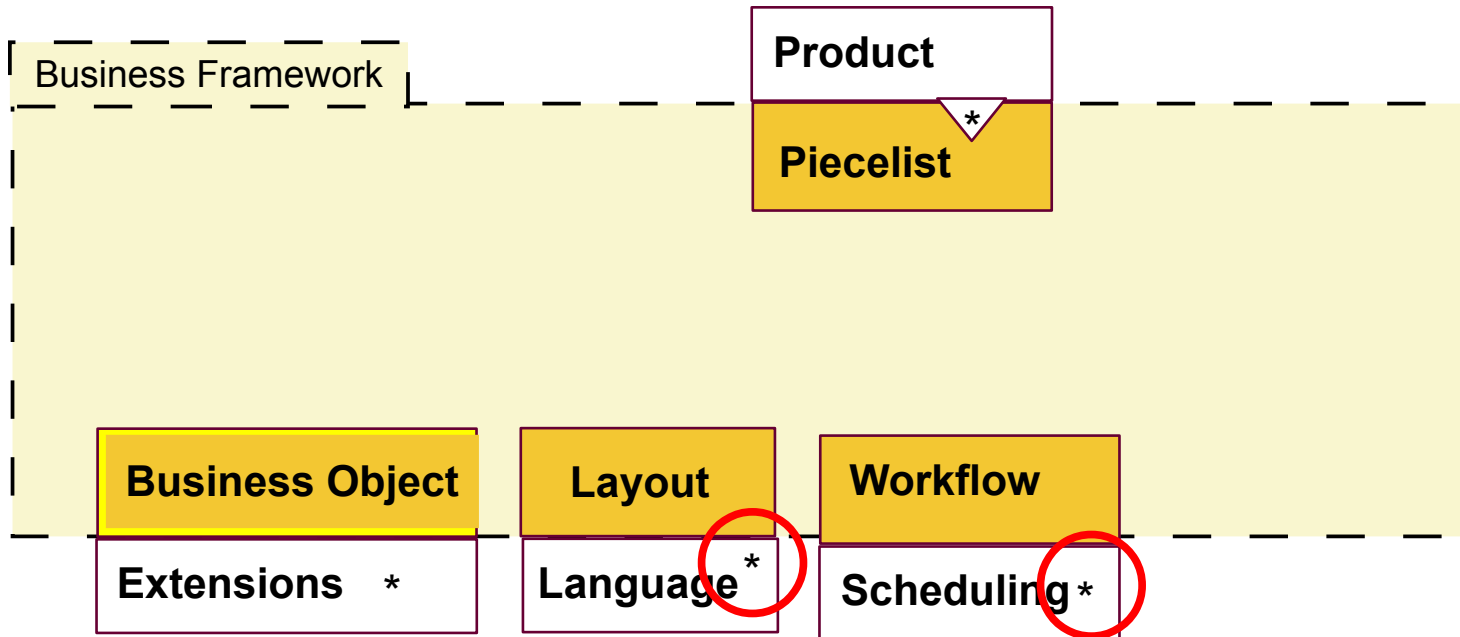
79



# A Business Framework with Several Languages Simultaneously

80

- ▶ Problem: business frameworks have an enormous number of framework hooks



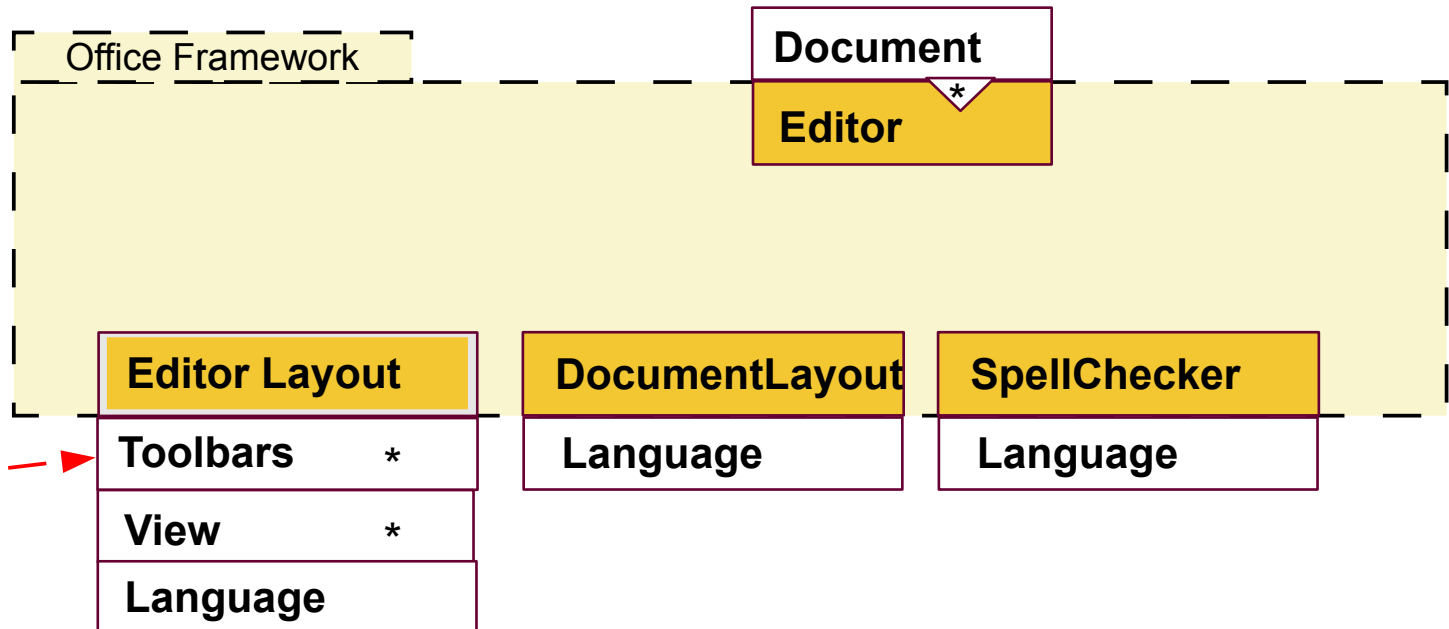


# OpenOffice

81

## ▸ Variabilities

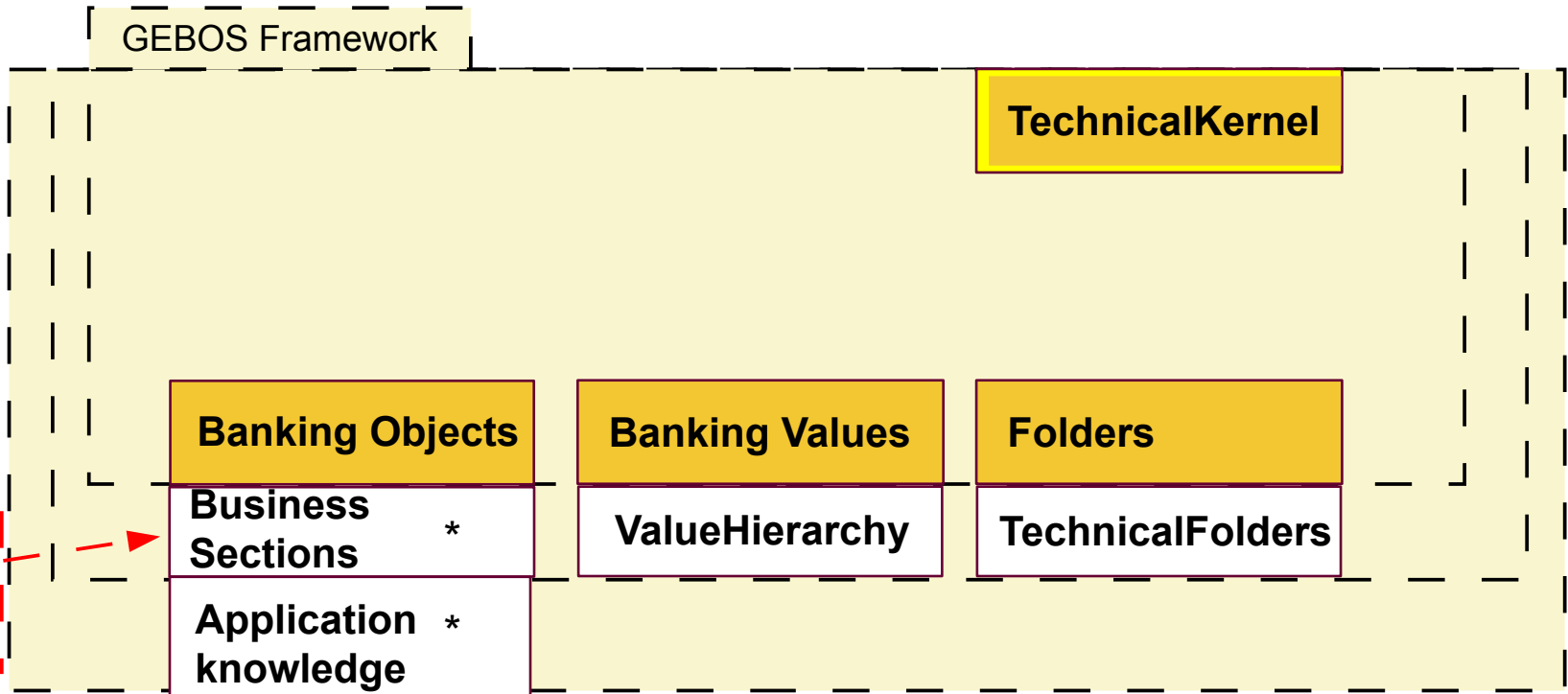
- Type of program (word, slides, drawings, calc, ...)
  - Structured documents (Composite pattern)
  - Embeddings of all document types into other document types possible
- Language
- GUI
  - Visible toolbar (visibility, position) of MainToolbar, FunctionBar, ObjectBar, ColorBar, OptionBar, PresentationBar, HyperlinkBar
  - Views, such as StandardView, OutlineView, HandoutView



# GEBOS Banking Layered Framework

82

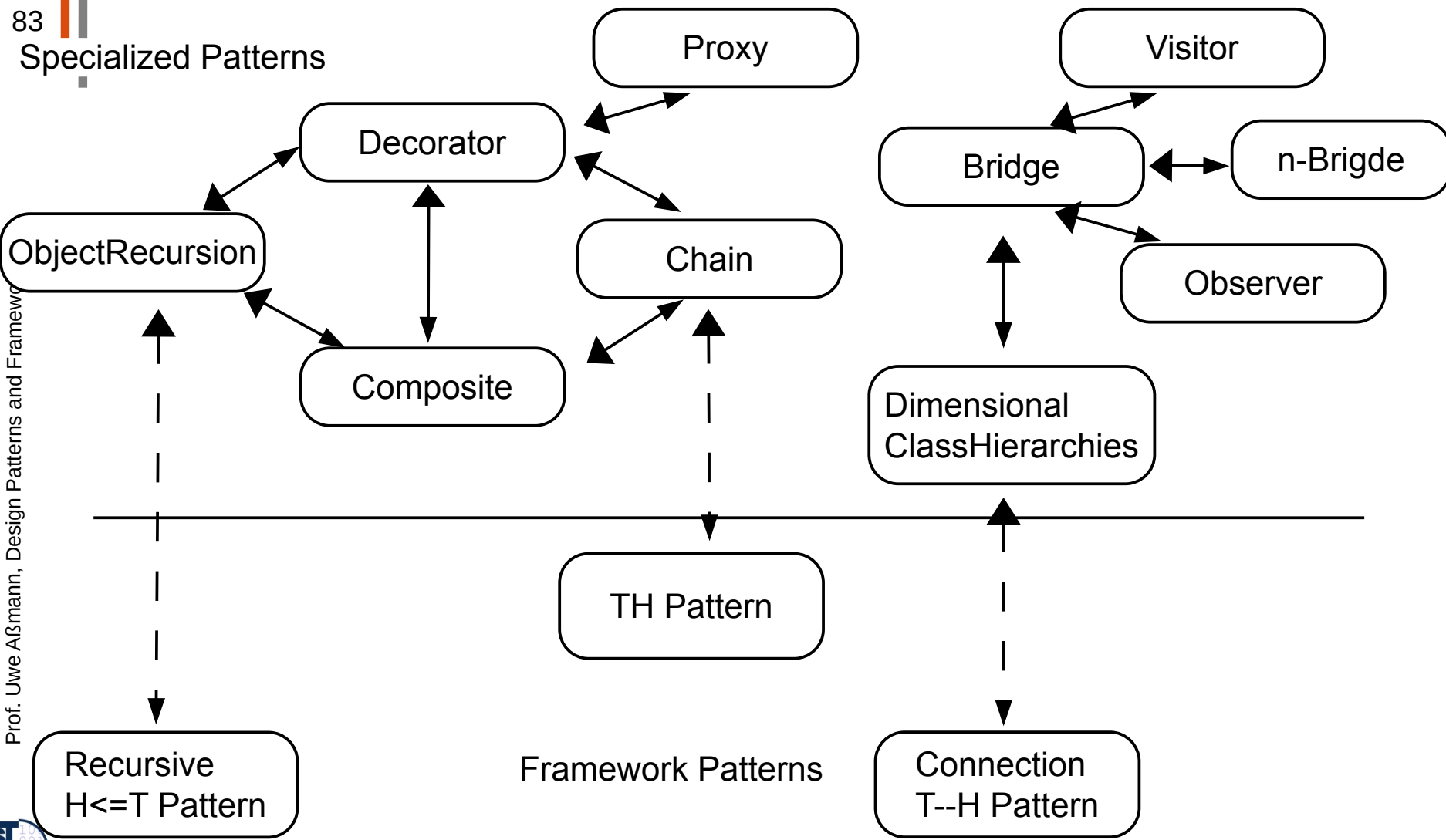
- ▶ If a template class of a framework hook has several hook classes (e.g., as an n-Bridge), then the Framework becomes layered



# Relations Extensibility Patterns

83

Specialized Patterns



# Summary

84

- ▶ When overlaid with a T--H metapattern, a design pattern becomes a *framework hook pattern* for the interface of a framework
- ▶ These are *mini-connectors* between a framework and its application classes
  - More flexible than just generic classes (generic frameworks) or delegation (blackbox) or inheritance (whitebox)
- ▶ The framework hook patterns determine very precisely how a framework is to be instantiated
- ▶ There are more kinds of dimensional frameworks
  - Dimensional T—H (n-Bridge LF),  $H \leq T$ , TH,  $T > H$  dimensional frameworks
- ▶ 1-T&H framework hook patterns can be used for variability of the framework
- ▶ n-T&H for extensibility.

# The End

