Construction of Flexible Software

UNIVERSITÄT SALZBURG

Contents

- Configuration parameters
- Concepts and contruction principles for flexible, object-oriented product families
- Design Patterns

UNIVERSITÄT SALZBURG

Configuration

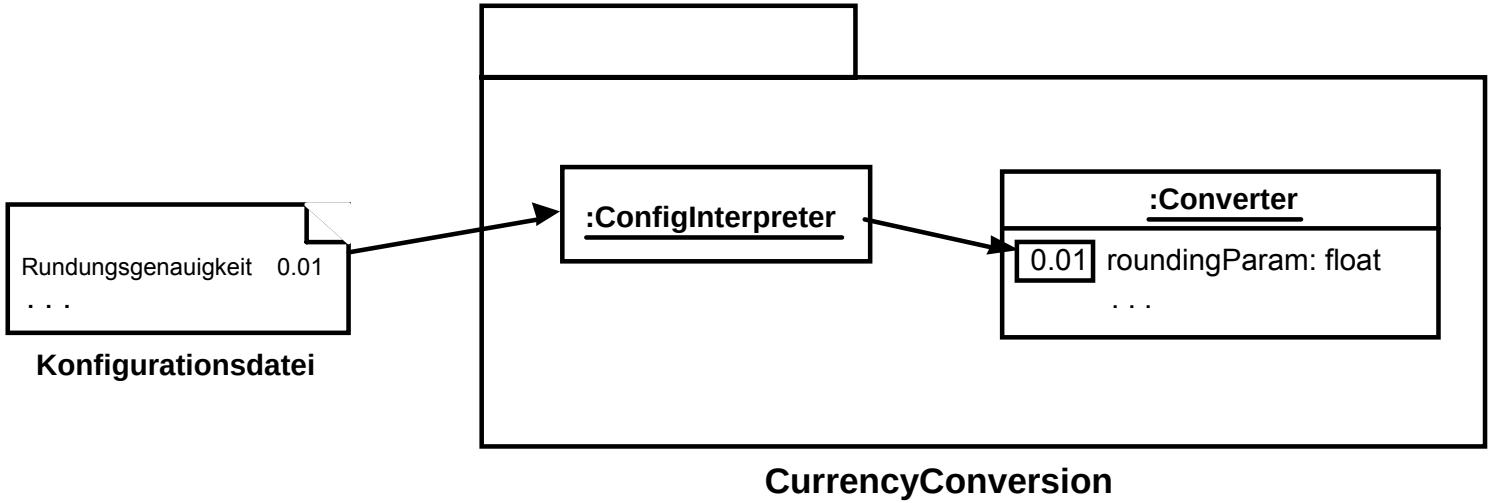
UNIVERSITÄT SALZBURG

Definition

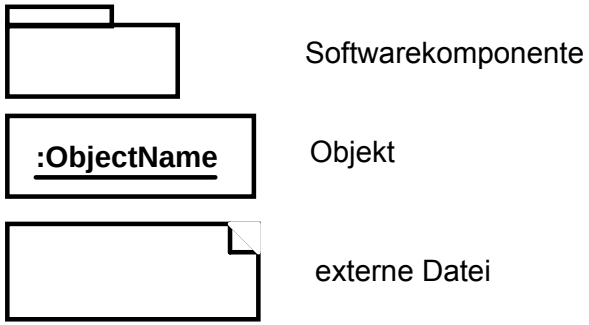
- Configuration parameters are placed in configuration files.
- Configuration parameters correspond to persistent, global (= static) variables.

UNIVERSITÄT SALZBURG

Example

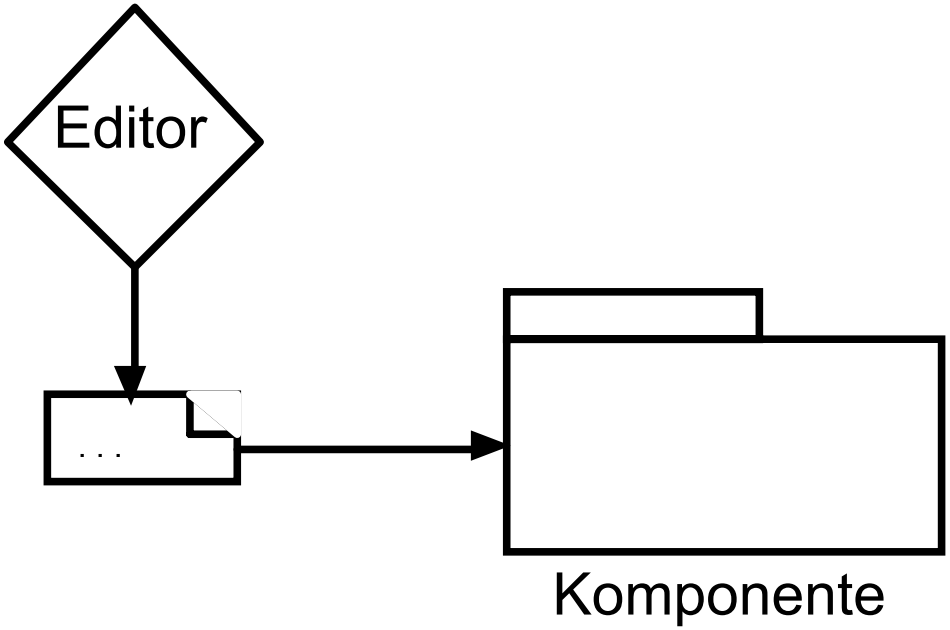


Legende:



UNIVERSITÄT SALZBURG

Generating the Configuration File



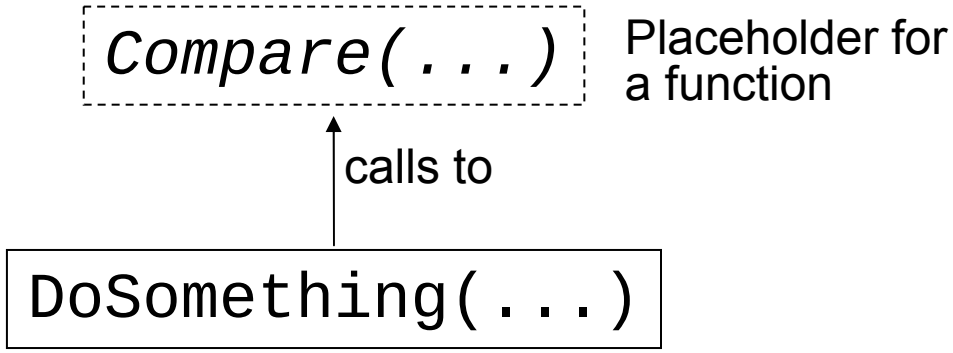
Example:

GUI Configuration file = Resource file

Visual, interactive construction with help from resource editors

UNIVERSITÄT SALZBURG

The Callback Style of Programming (I)



DoSomething calls a function which it has received as an argument. This shows the meaning of the callback style of programming:

One can conceptually distinguish whether a function or a procedure is **called directly (call)** or whether a function or a procedure passed as a parameter is **called indirectly (by means of callback).**

UNIVERSITÄT SALZBURG

The Callback Style of Programming(II)

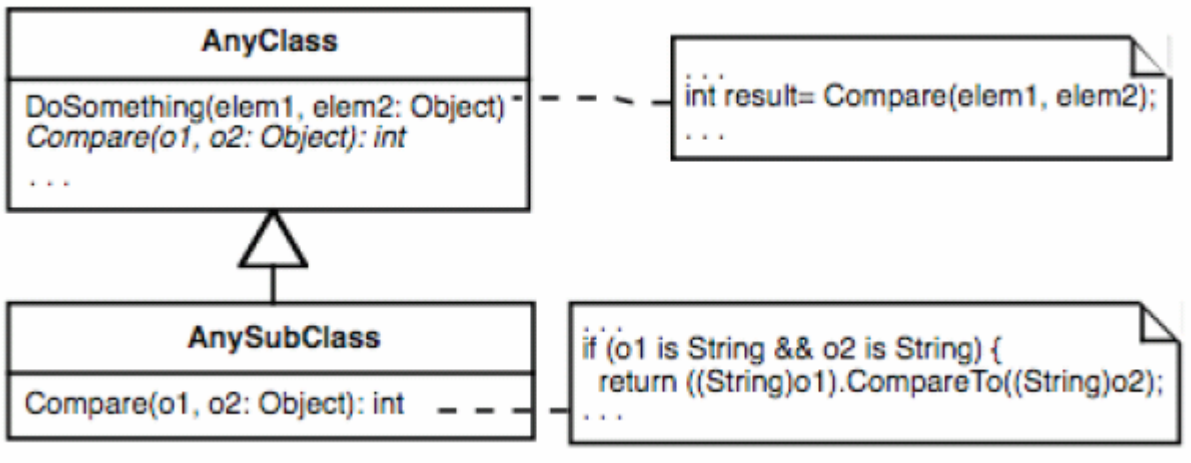
```
void DoSomething(int (*Compare)(void*, void*),
                                 void* elem1, void* elem2 )


int StringCompare(void* string1, void* string2) {
        return strcmp(    // C-Bibliotheksfunktion strcmp
                          (char*)string1,
                          (char*)string2
                    );
  }  // StringCompare

DoSomething(StringCompare, “first“, “second“);
```

UNIVERSITÄT SALZBURG

The Callback Style of Programming(III)



UNIVERSITÄT SALZBURG

Concepts and Construction Principles for Flexible Object-Oriented Product Families

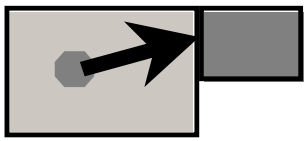
UNIVERSITÄT SALZBURG

Definition

Product Family: A piece of software from which different applications can be formed by the callback style of programming, i.e. its behavior is changeable and/or expandable.

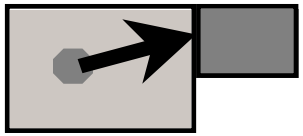
UNIVERSITÄT SALZBURG

Abstract Coupling



GPS-Komponente

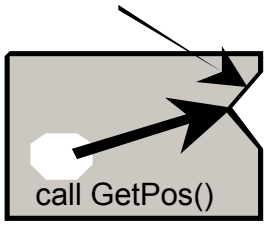
Navigationskomponente

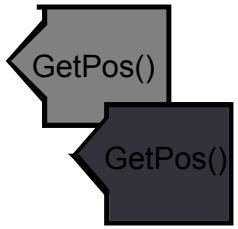


Galileo-Komponente

Navigationskomponente

„Stecker“ PosSystem





GPS- Komponente

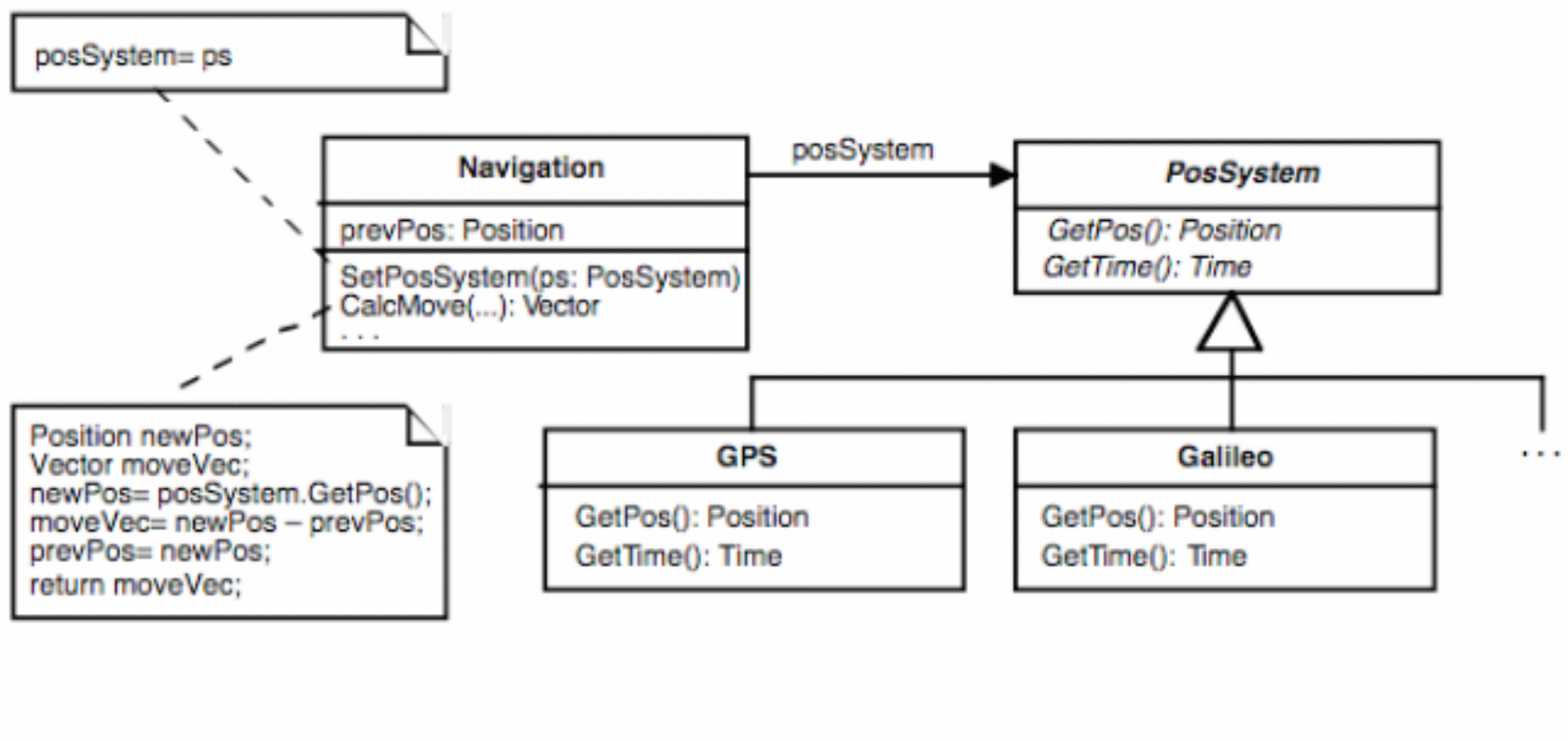
Galileo- Komponente

Navigationskomponente

UNIVERSITÄT SALZBURG

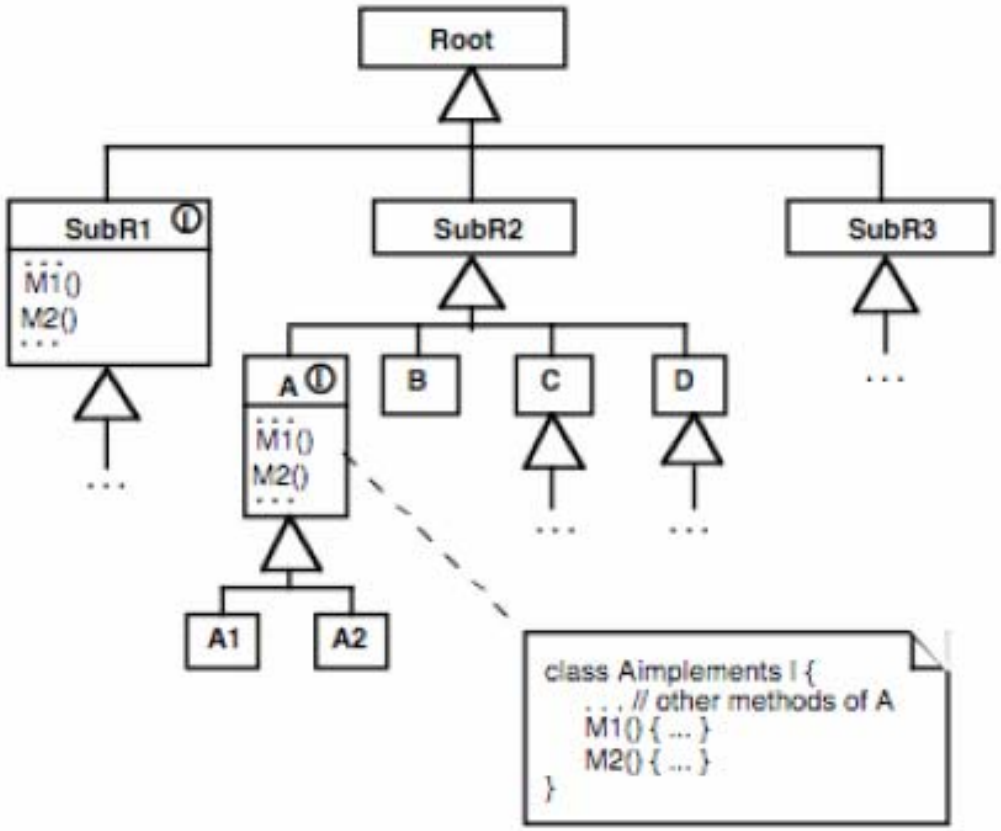
Abstract Coupling by Abstract Classes

Navigation system example:



UNIVERSITÄT SALZBURG

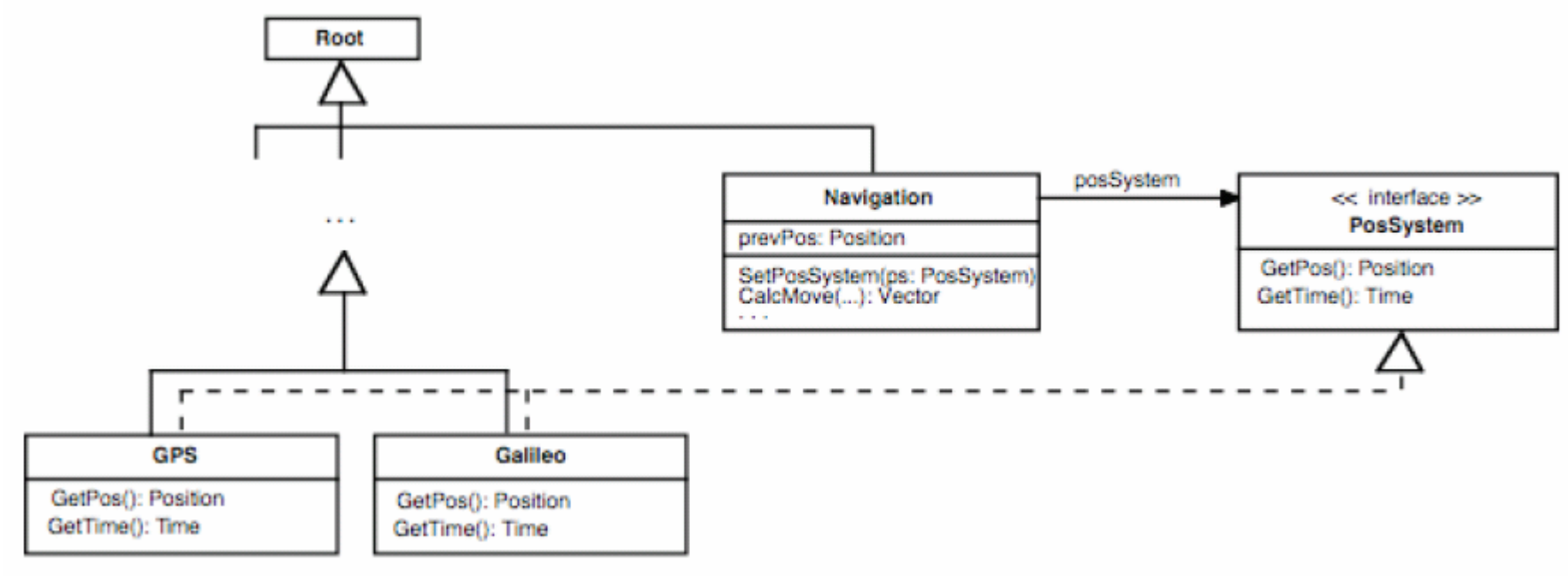
Alternative: Interfaces



UNIVERSITÄT SALZBURG

Abstract Coupling by Interfaces

Navigation system example:



Template and Hook Methods

UNIVERSITÄT SALZBURG

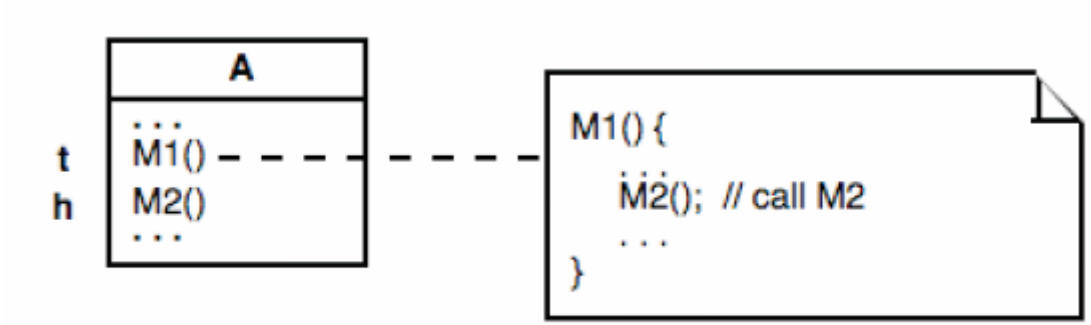
Definition

If a method is called in another method's implementation, then we call the **calling method the Template method** and the **called method the Hook method**.

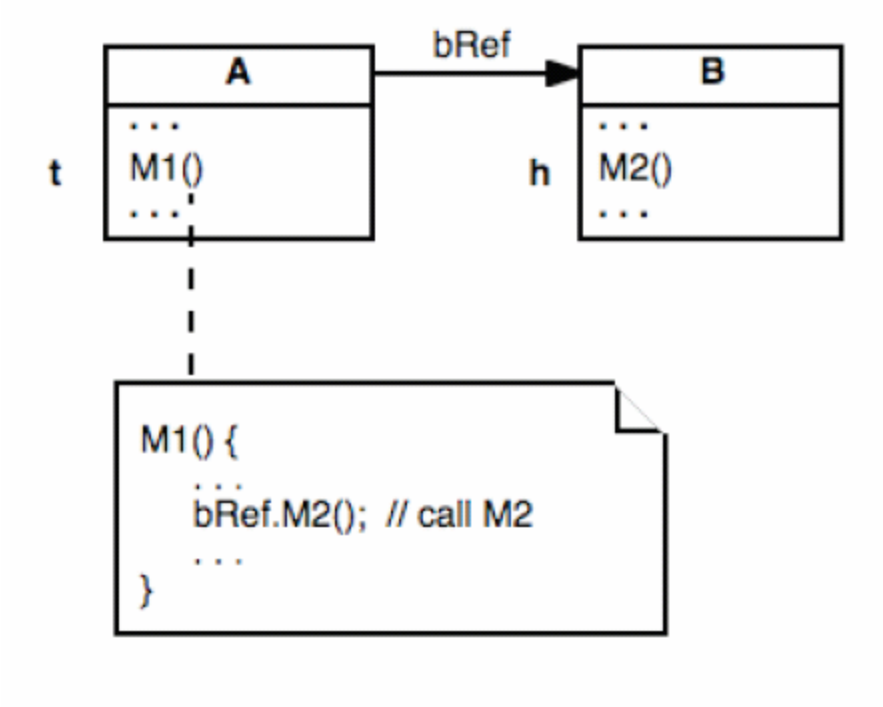
The template method addressed here has nothing to do with the C++ language construct `template`.

UNIVERSITÄT SALZBURG

Both Methods in the Same Class

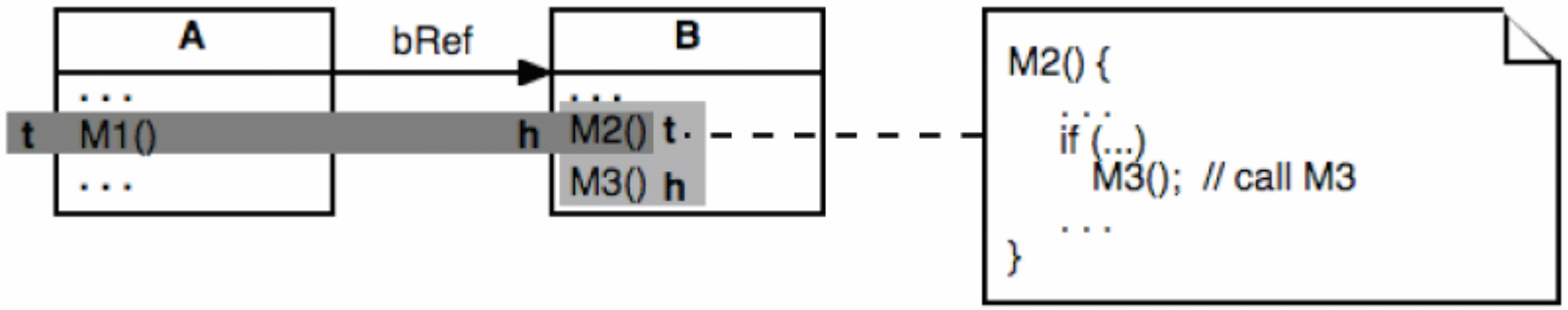


Template and Hook Methods in Different Classes



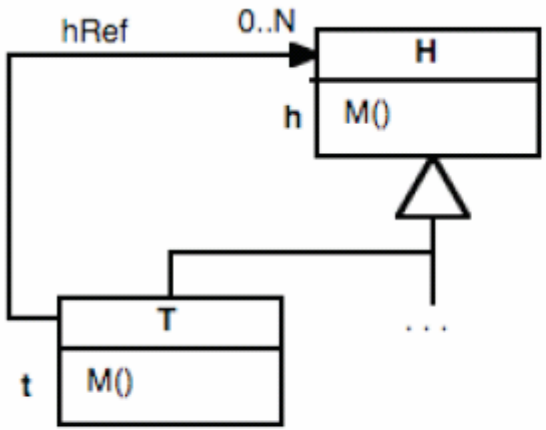
UNIVERSITÄT SALZBURG

The same method can be both Template and Hook depending on the context

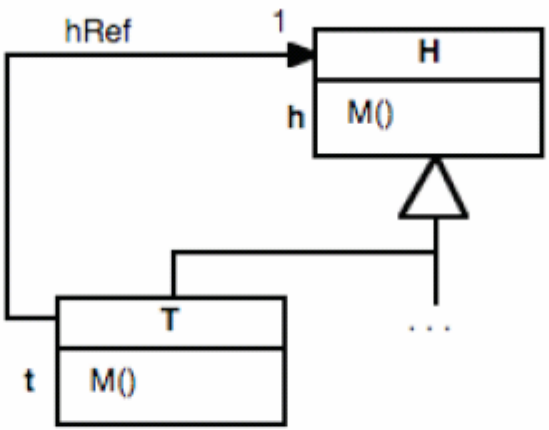


UNIVERSITÄT SALZBURG

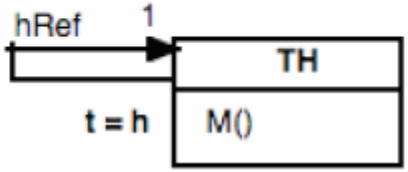
Combinations With Recursiveness



Composite



Decorator



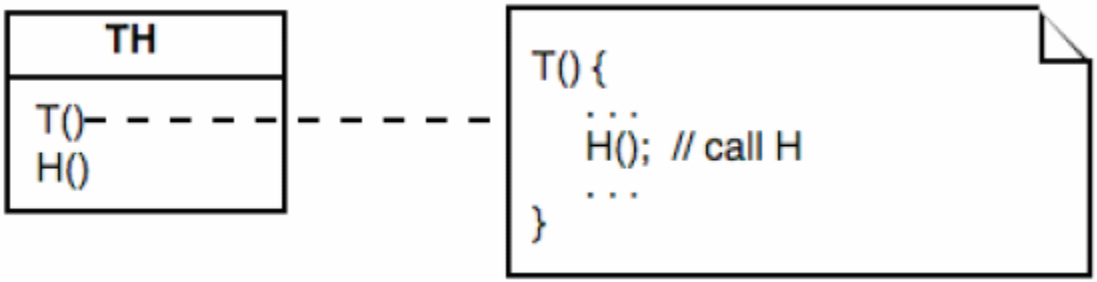
Chain-Of-Responsibility

UNIVERSITÄT SALZBURG

Hook Method Construction Principle

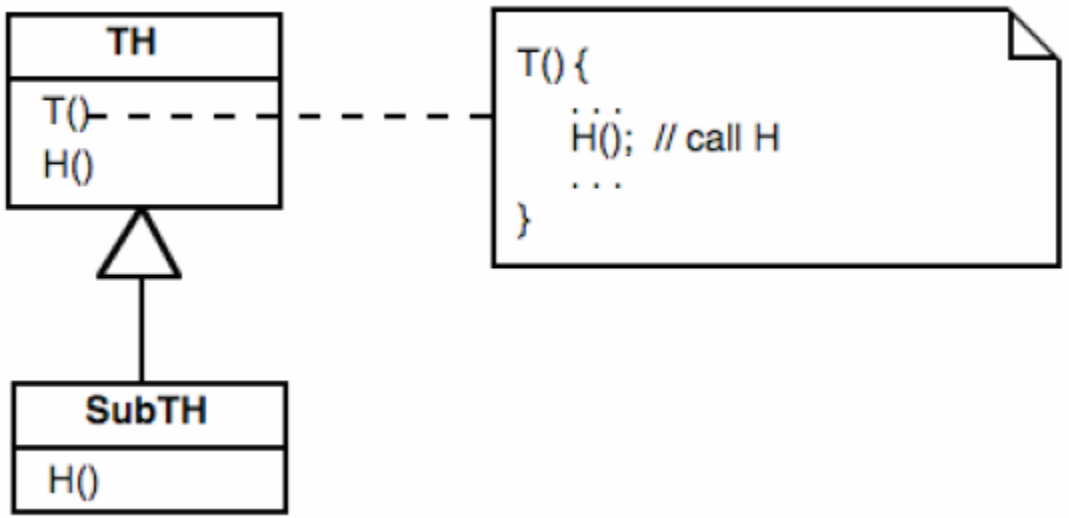
UNIVERSITÄT SALZBURG

Hook Method: Adaptation of T() by overwriting of H()



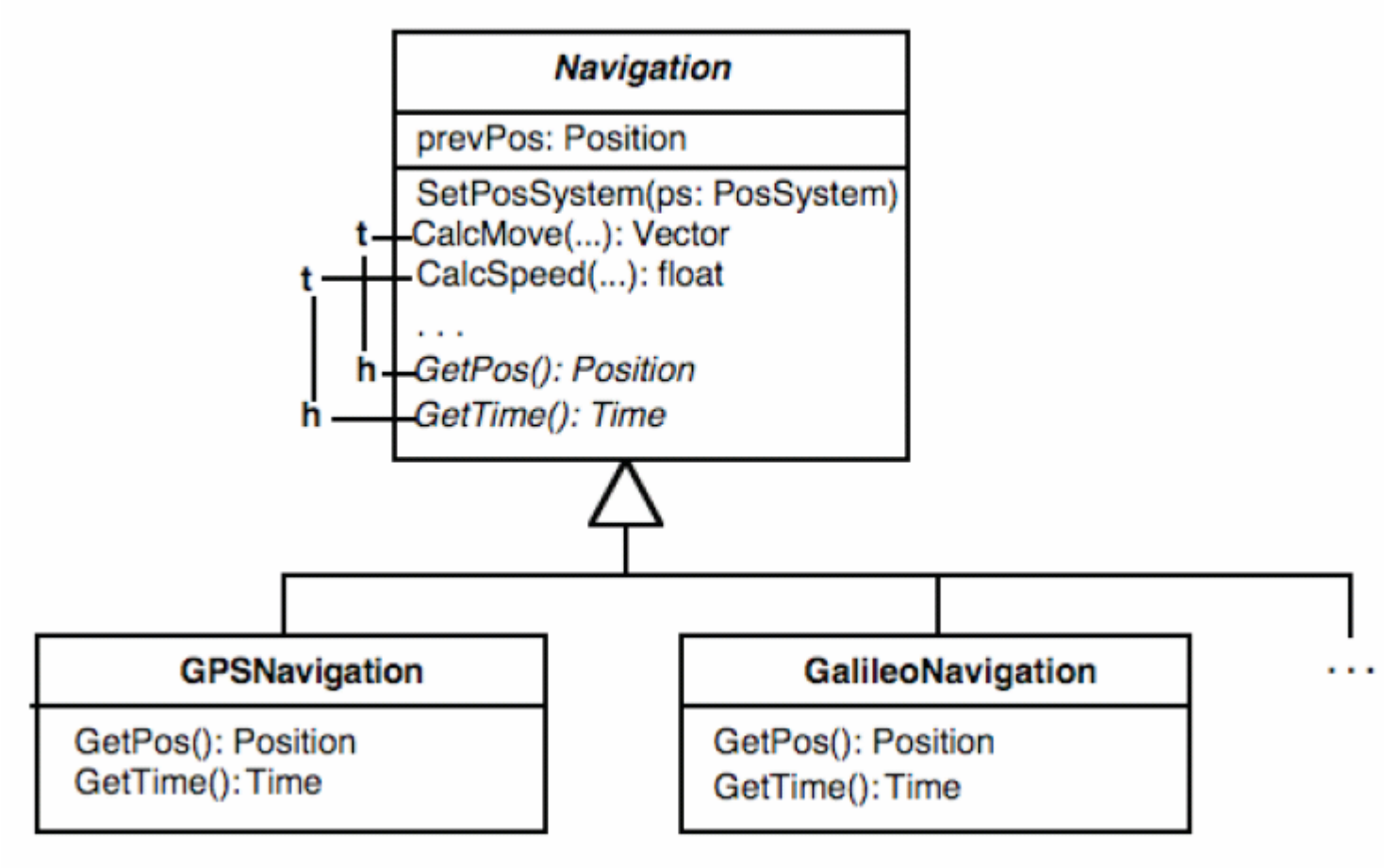
UNIVERSITÄT SALZBURG

Adaptation by Overwriting the Hook Method H()

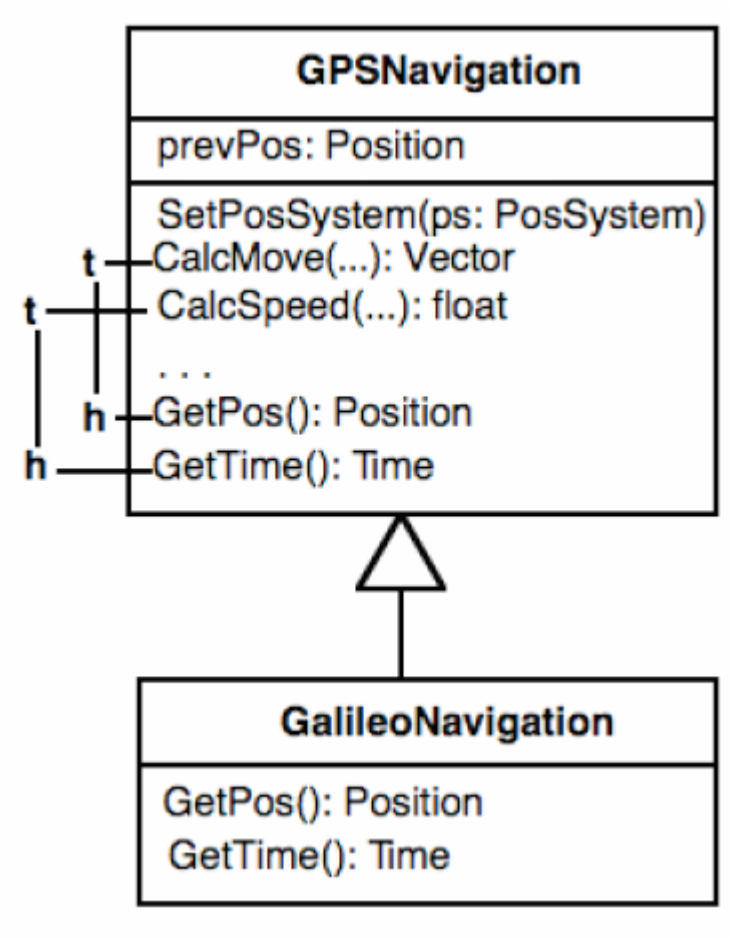


UNIVERSITÄT SALZBURG

Application Example: Navigation System(I)



Application Example: Navigation System(II)



Problem: Galileo is not a specialization of GPS!

UNIVERSITÄT SALZBURG

Summary *Hook Method*

+ Simplicity: For an adaptable behavior, one must plan only a hook method.

- Adaptability requires a subclass formation and the overwriting of the hook method.

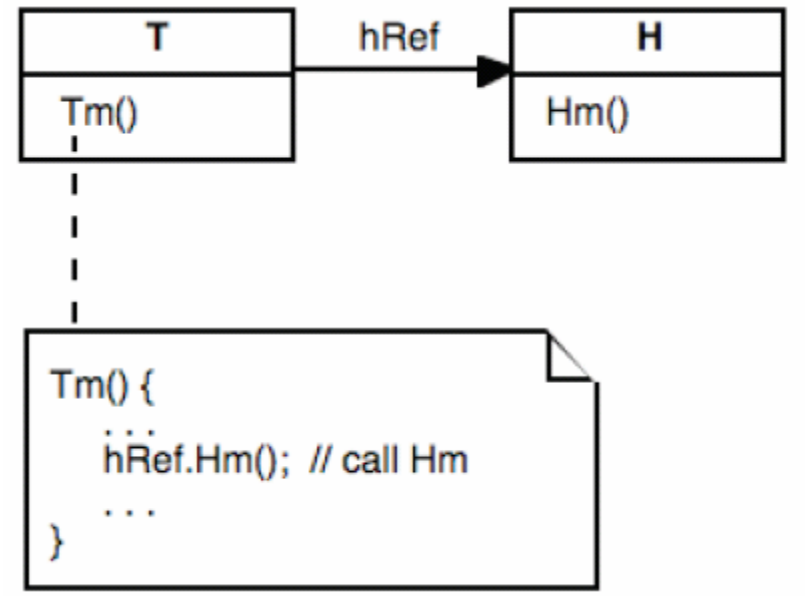
  In many cases, the hook method construction principle is sufficient to achieve the flexibility required for adaptation.

UNIVERSITÄT SALZBURG

The Hook Object Construction Principle

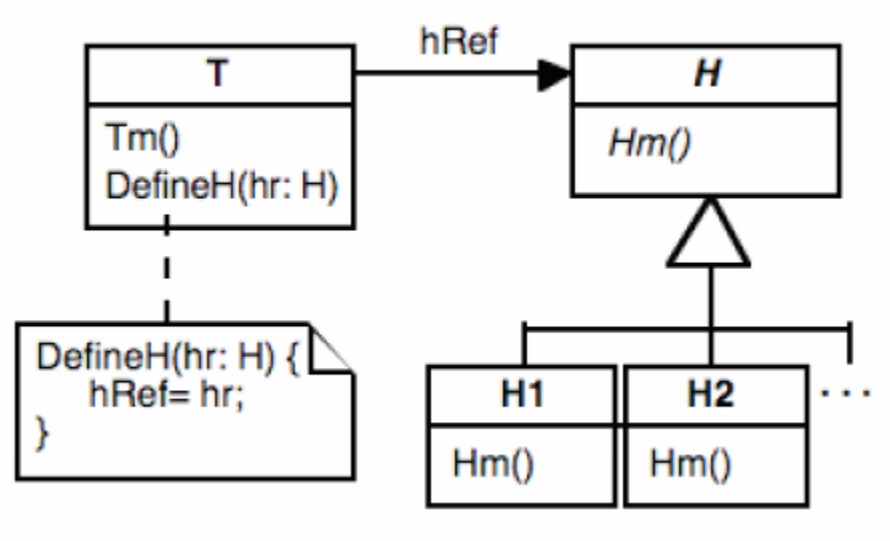
UNIVERSITÄT SALZBURG

Hook Object: Adaptation of T() by Plugging an H Object In



UNIVERSITÄT SALZBURG

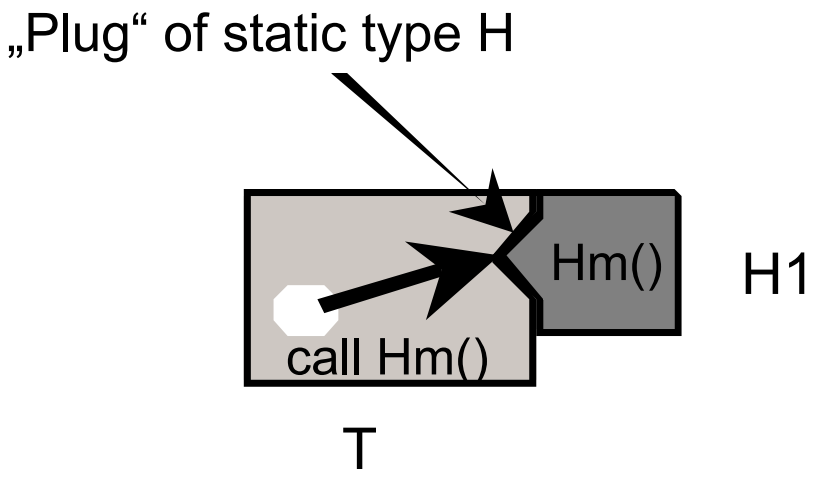
Adaptation by Composition (I)



⇒ Adaptability at runtime

UNIVERSITÄT SALZBURG

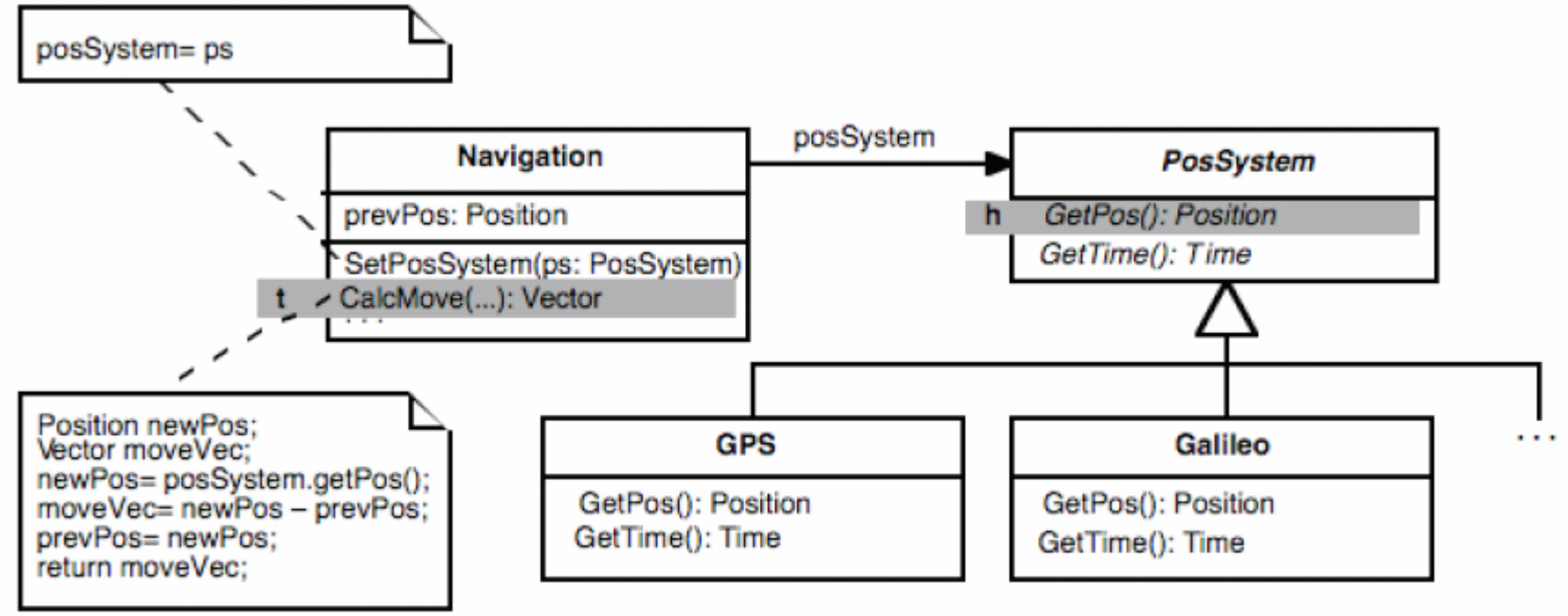
Adaptation by Composition (II)




```
T sampleT= new T();
sampleT.DefineH(new H1());
```

UNIVERSITÄT SALZBURG

Application Example: Navigation System(I)



Application Example: Navigation System(II)

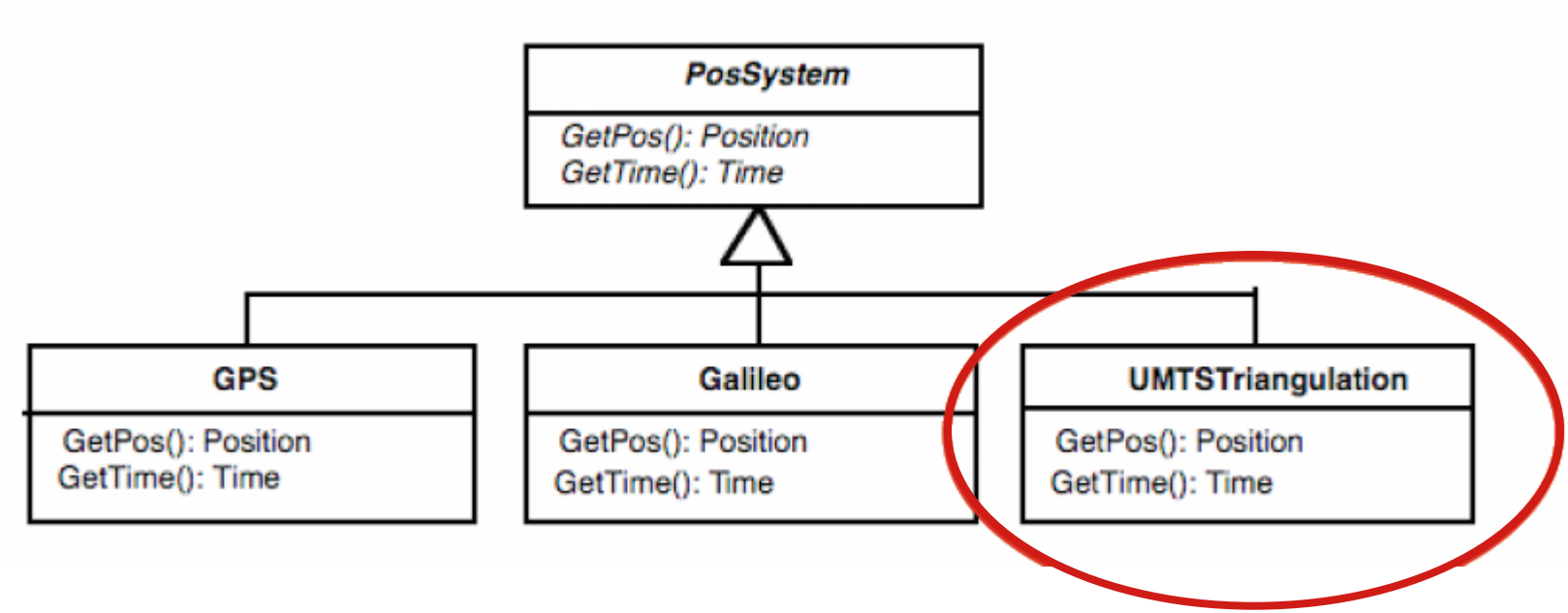
Composition for achieving a navigation system:

(a) GPS-based

(b) Galileo-based

UNIVERSITÄT SALZBURG

Extension of the Pluggable Components at Runtime?




```
Navigation navigation= new Navigation(...);
String nameOfAddtlClass= “UMTSTriangulation“;
Object anObj= new nameOfAddtlClass;  // not possible
                                     // correct solution follows
navigation.SetPosSystem((PosSystem)anObj);
```

UNIVERSITÄT SALZBURG

By Reflection in .NET/C#


```
Navigation navigation= new Navigation(...);
. . .
String nameOfAddtlClass= “UMTSTriangulation“;
Type typeOfAddtlClass= Type.GetType(nameOfAddtlClass);
Object anObj;
PosSystem posSys;

if (typeOfAddtlClass != null) {
   anObj= Activator.CreateInstance(typeOfAddtlClass);
    if (anObj != null && anObj is PosSystem)
        posSystem= (PosSystem) anObj;
    else  ...  // error handling
}
navigation.SetPosSystem(posSys);
```

Summary *Hook Object*

+ Simple configuration, also at runtime

- Higher complexity of design and implementation than in the hook method principle

UNIVERSITÄT SALZBURG