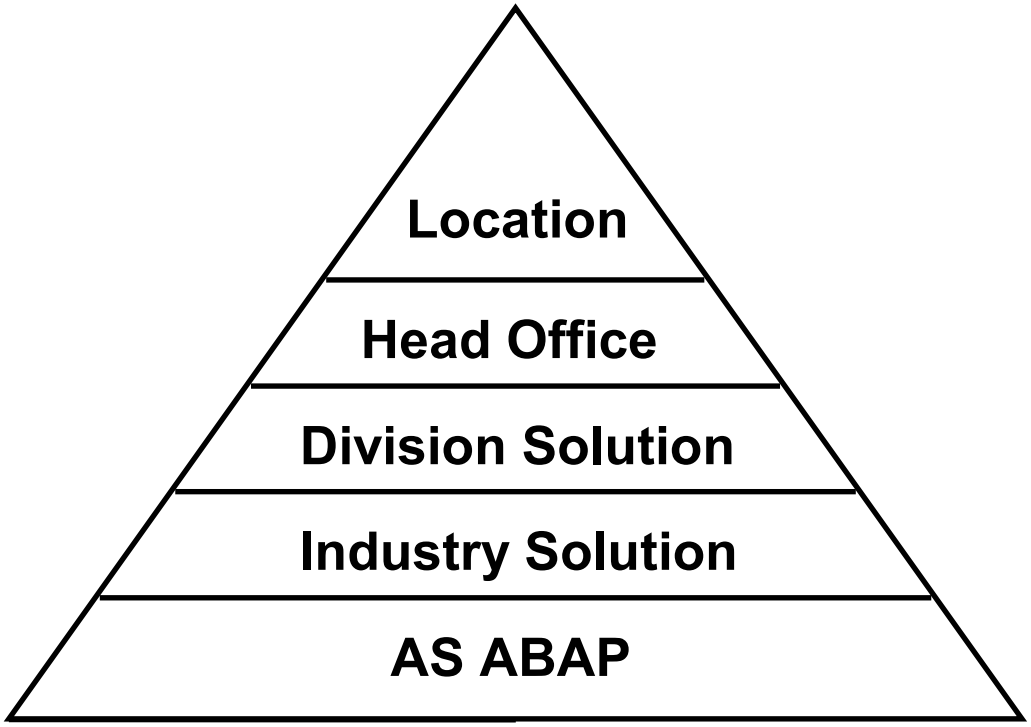
ABAP Software Architecture - Modularization and Composition of Huge Applications

Tobias Trapp

SDN Session on SAP Community Day, 2008

Software Pyramid

Software pyramid is a standard architectural pattern



Software Components

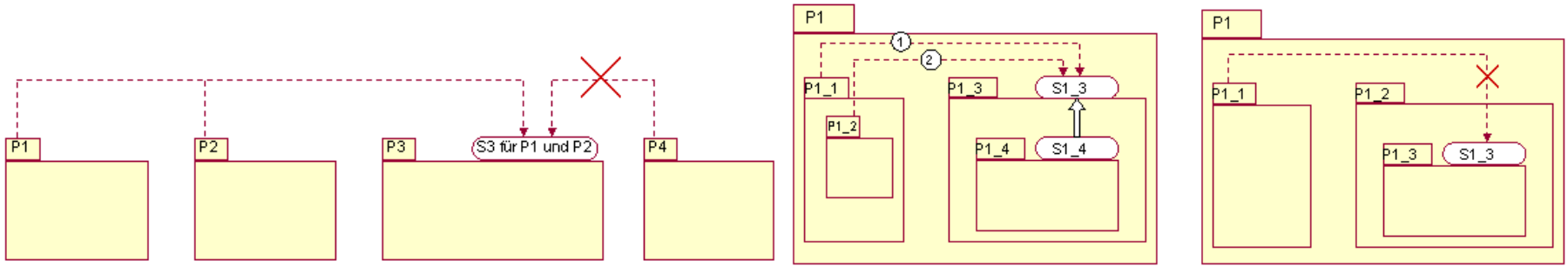
What exactly is it?

- ✓ Usually each layer of the software pyramid is located in a *software component*.
- ✓ The software component is a concept from software logistics. It contains a set of ABAP development elements (including packages) that belong together and can only be deployed usefully as a whole.
- ✓ In custom development projects at first you have to define the dependencies from the SAP standard software components.
- ✓ A software component may contain different applications.
- ✓ In custom development projects usually you have one software component per system type (ERP, CRM, BI…)

Software Components vs. Packages

What are the differences?

- ✓ The package is a concept from software architecture, which structures development elements (hierarchically if necessary) and assigns them to clearly defined package interfaces.
- ✓ In terms of software components you have a strict component hierarchy – the dependencies between packages can be more complex.



How to start?

You should answer the following questions:

- ✓ Which applications you want to develop?
- ✓ Are there any dependencies between them?
- ✓ What about documentation and support? How do your customers know which release notes of your products have to be applied?
- ✓ What are the dependencies from SAP standard products? What kind of impact will the evolution of the SAP standard have on your products?

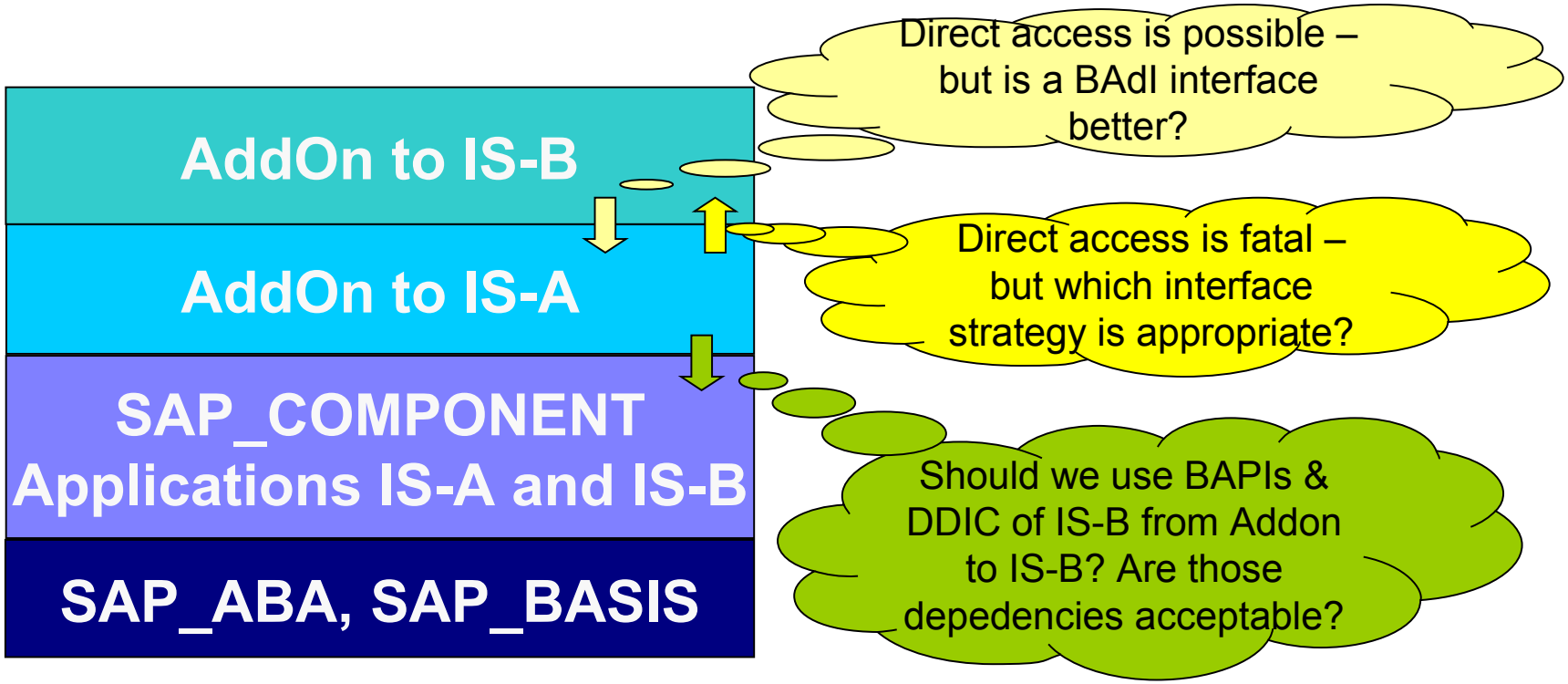
Modularization

What it is good for:

- ✓ In terms of applications: You create a product portfolio.
- ✓ In terms of software components: You get sets of applications that can be installed and used independently.
- ✓ You can achieve maintainable applications by using the ABAP package concept.

Hierarchy of Software Components

Even Simple Cases Require non Trivial Interface Strategies



Interfaces between Applications

This leads to the question how to link applications inside the component.

- ✓ Functional dependencies are more severe than technical dependencies (usage of DDIC).
- ✓ Besides explicit interfaces (package interfaces, BAPIs) there are a lot of interface strategies:
  1. Enhancements (new BAdI)
  2. Enhancements using a specific framework (BTE, BDT, Post Processing Framework…)
  3. Create your own framework for extensions
  4. Publish & Subscribe interfaces by using BOR events or its ABAP OO counterpart.

But which is appropriate in which situation?

Interfaces between Applications: Rules of thumb

oscare AOK systems

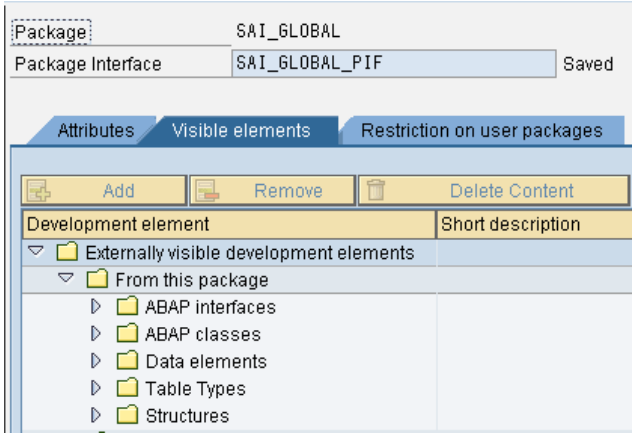
What is the right interface strategy:

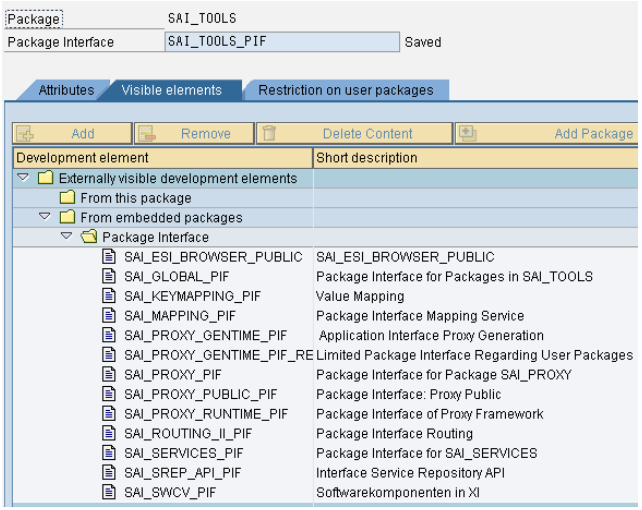
- ✓ If SAP Standard uses a special framework for extensions you should use this in your AddOns, too.
- ✓ BAdIs are the right choice when an interface has the character of an enhancement or if you want to have the chance to reimplement an interface.
- ✓ Publish & Subscribe interfaces are the right choice for loose coupling. This is the case if you want to decouple processes from post processes and it effects your choice of error-handling strategy
- ✓ Explicit interfaces by using package interfaces are always best.

The ABAP Package Concept: Benefits

Differences between packages and development classes:

- ✓ We can nest packages and get a hierarchical structure.
- ✓ At the top level of such an hierarchy resides a structure package that corresponds to an application or even a component.
- ✓ Packages have interfaces that contain ABAP dictionary elements, classes etc. and classify what is private and what is public.





Anti Patterns in ABAP Architecture

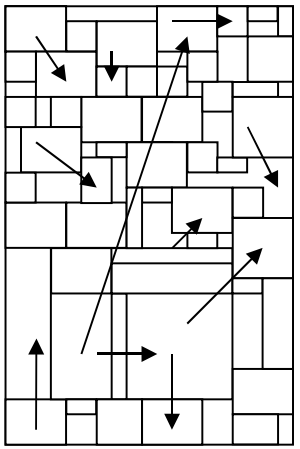
- ✓ Many developers don't understand the difference between packages and software components.
- ✓ The *reason* for this is that they think solely in terms of software logistics: transport requests and components.
- ✓ This results in design errors. I will introduce two Anti-Patterns in ABAP development.

What happens if you confuse the concepts of package and component?

Anti Patterns in ABAP Architecture

„Crawling Chaos“

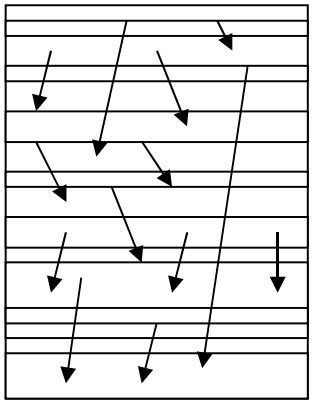
- ✓ A lot of developers don’t care about packages: “We deliver software as a whole – as one software component”.
- ✓ This was exactly the concept of a development class: Everybody uses everything.
- ✓ And this was the concept of R/3: It consisted of 800.000 development objects in 3000 development classes.
- ✓ This is hard to maintain.



Anti Patterns in ABAP Architecture

„Dung Pile“

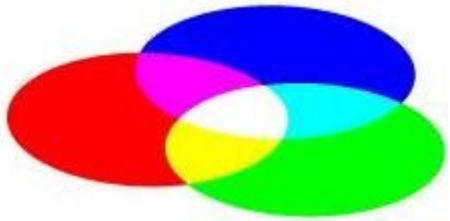
- ✓ This is inspired by transport hierarchy: Packages can be installed like components from below to above.
- ✓ There are only dependencies from higher packages downwards but not upwards.
- ✓ In huge applications strict hierarchies are hard to apply.



Best Practices

Strict layer models like „dung pile“ are history.

- ✓ Define main packages which group other packages. Main packages correspond to partial applications.
- ✓ Define packages for generic tools as the base for your application.
- ✓ Try to avoid cyclic dependencies.
- ✓ Don’t confuse shared with cross-functional aspects.



Cross sectional functions should reside on top of a package hierarchy.

Discussion

What are your experiences with modularization and the ABAP package concept?

- ✓ Who defines packages and interfaces in your organization?
- ✓ What are your experiences with the ABAP package concept? How did you introduce it in your development projects? How do you deal with legacy code that violates SAP standard interfaces?
- ✓ What kind of package checks do you use: R3ENTERPRISE, RESTRICTED or NONE?
- ✓ What are your experiences with the creation of structure packages and interfaces?
- ✓ What are your rules of thumb for modularization?