Components ?

Why do I have the feeling something is wrong ...

Petr Tůma
Distributed Systems Research Group
Faculty of Mathematics and Physics
Charles University, Czech Republic

http://nenya.ms.mff.cuni.cz

The Rose Garden

"Components have become an important part of software technologies."

Adamek, Plášil, 2003

"In the near future, the majority of software applications will be composed from reusable, potentially off-the-shelf software components."

Plášil, Višňovský, 2002

"Component platforms ... play a major role in the research, development and operation of current distributed industrial information systems."

Brada, 2002

"The trend to construct software systems as a collection of cooperating reusable components ... has become widely accepted."

Bálek, Plášil, 2001

The Rose Garden

“Software developers welcome the emergence of a robust marketplace of software components.”

Seacord, Hissam, Wallnau (SEI CMU), 1998

“Component-based software technologies have been increasingly considered as necessary for creating, testing, and maintaining the ... software of the future.”

Orso, Harrold, Rosenblum (ISR UCI), 2000

“There exists a clear trend for business, industry and society to place increasing dependence on Information Systems, consisting of the integration of numerous, disparate and autonomous component systems.”

Zarras, Kloukinas, Issarny (INRIA Rocquencourt), 2002

“Component-based software engineering has existed in one form or another for a number of years.”

Parrish, Dixon, Cordes (DCS Alabama), 2001

Look Around !

If components are so widely used ...

- **How many component applications did you write ?**
- **How many component applications did you use ?**
- **How many component applications did you see ?**

Look Around !

If components are so widely used ...

- How many component applications did you write ?
- How many component applications did you use ?
- How many component applications did you see ?

My take on this ...

- **Components are present but not ubiquitous.**
- **Most components only fit into predefined frameworks.**

Look Around !

If components are so beneficial ...

- What do components bring that libraries do not ?
- What do components bring that modules do not ?
- What do components bring that frameworks do not ?

Look Around !

If components are so beneficial ...

- What do components bring that libraries do not ?
- What do components bring that modules do not ?
- What do components bring that frameworks do not ?

My take on this ...

- Components do not supersede these concepts.
- The sole benefit of components is runtime integration.

Look Around !

If component architectures are so helpful ...

- How many development tools do you know ?
- How many design methodologies do you know ?
- How many times did you use component architectures ?

Look Around !

If component architectures are so helpful ...

- How many development tools do you know ?
- How many design methodologies do you know ?
- How many times did you use component architectures ?

My take on this ...

- Components are too precise for early design stages.
- Runtime benefits do not translate to design benefits.

To Quickly Summarize ...

“The proof of the puding is in the eating !”

Components bring little benefit during design ...

- The concept alone is nothing new.
- The design needs to be vague.

Components bring tangible benefit during runtime ...

- The runtime integration is useful.

… And How Does This Relate To Us ?

Design problems in SOFA ...

- **It seems too difficult to write a complete architecture description in the early design stages.**

  Probably comes from trying to apply the concept of "runtime components" to "architecture components".

Coding problems in SOFA ...

- **It is extra work having to use the language mapping.**

  Probably comes from seeing that everyone has mappings.

Runtime problems in SOFA ...

- **It is difficult to integrate components with other code.**

  Probably comes from trying to apply the concept of a component as an all purpose building block.

- **Almost no reflection is provided at the architecture level.**

  Because we are worried about changing the architecture.

What I Would Do ...

Changes to the architecture description language ...

- Separate details not related to architecture.
- Enable multiple views of one design.

Changes to the repositories ...

- Strive for independence from the structure of data.

Changes to the runtime ...

- Reflect a flexible architecture model.

Architecture Description Language

We use it for ...

- **Specifying how to combine components.**
- **Specifying the interfaces of components.**
- **Maybe something else ?**

How about we separate these functions ...

- **Architecture description refers to interfaces by names.**
- **Interfaces are described separately for coding purposes.**

ADL: Example Interfaces

```
module SOFA {
  module demos {
    module cplayer {

      interface ControlInterface {
        void play();
        void stop();
      };

      interface DisplayInterface  {
        void updateStatus(in string status);
      };
```

... this would go into separate interface definition.

ADL: Example Architecture

```
system architecture CUNI ::SOFA::demos::cplayer::Main
implements ::SOFA::libs::Application {

  inst ::SOFA::demos::cplayer::Display      display;
  inst ::SOFA::demos::cplayer::Controller   controller;
  inst ::SOFA::demos::cplayer::Speaker     speaker;
  inst ::SOFA::demos::cplayer::ReadingUnit readingUnit;

  bind controller:control to readingUnit:control using CSProcCall;
  bind readingUnit:display to display:display using EventPassing;
  bind readingUnit:audio to speaker:audio using DataStream;
};
```

... this would remain in the architecture definition.

ADL: What will this change give us ?

The ability to use more flexible interface definitions ...

- **The interface definitions need not use our syntax.**

  We definitely need to step outside CORBA IDL.

- **The interface definitions can use multiple languages.**

  An application can use Java for most of its interfaces and CORBA IDL for those interfaces that need to be distributed.

The ability to describe architectures that use other components than just SOFA components.

- **We will be able to use some of our tools elsewhere.**

  We might generate connectors for Fractal with practically no change to our tools, or deploy Fractal components as easily.

ADL: Can we go further ?

I believe we should stabilize our architecture description language as a tool for static configuration description, which has its uses.

Besides that, here are some thoughts ...

- **Interfaces are just groups of methods. Roles are just groups of interfaces. Frames are just groups of interfaces or roles.**

  How about we devise a more general way of hierarchically grouping methods or interfaces ?

- **By definition, any design is vague in details.**

  We should maybe think why the vagueness of UML does not hinder people from using it while the preciseness of SOFA ADL does.

- **Architecture has many aspects. Behavior. Dynamism. Etc.**

  It would be nice if we could express those aspects in some extensible manner rather than piling up more and more deployment descriptors to augment SOFA ADL.

Repositories

Our current repository structure follows SOFA ADL ...

- Most changes in SOFA ADL require repository change.
- We have nowhere to store additional information.

Can we think of a more flexible structure ?

- Some sort of ability to store entities and relationships.
- Maybe a system of plugins to help specific tools use specific information in a more handy way ?
- And if we have these plugins, we can even think about accessing other repositories than just SOFA.

Runtime

Our runtime was not designed to be open ...

- **A person working on a specific task should be able to extend SOFA without having to interfere with just about everything and everyone.**

Again, can we devise a more open design ?

- **Julia controllers seem to make sense from this perspective.**

I Know, Everybody Is A Critic ...

... but I still believe these proposals are feasible.

- **Separation of interfaces should not require much work.**
- **Flexible repository is probably a tough nut design wise.**
- **Open runtime is something we are approaching anyway.**

The End

100%