Component Software – selection of chapters 1-4

Prof. Dr. Wolfgang Pree

Department of Computer Science
University of Salzburg
cs.uni-salzburg.at

UNIVERSITÄT SALZBURG

Contents

- Motivation
- Market vs Technology
- Standards
- Foundations

UNIVERSITÄT SALZBURG

Motivation

going beyond OO

- "*Object orientation has failed but component software is succeeding*" Udell, 1994
- All other engineering disciplines introduced components as they became mature and they still use components.
- Solve current software crisis via an analogy to "Software Integrated Circuits"
- Components are independent units of deployment, allowing independent development

What are Software Components?

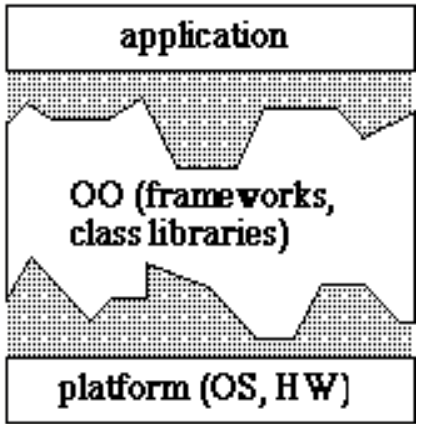
- Binary units of independent production / deployment
- Procedures, classes, modules and even applications in Executable form that remain composable (e.g., libraries)
- Components are for composition
- Composition enables existing components to be reused by rearranging them into even new components

Why Software Components?

- OO technology
  - mostly used for monolithic applications
  - ignores market aspects – only a small amount of class libraries and/or frameworks exists in the public/commercial domain
  - Definition of objects is purely technical – does not include notions of independence or late composition

- Component = unit of deployment
  - Framework, module or set of classes compiled and linked in a package

What is missing in OO?



visual/interactive configuration

interoperability

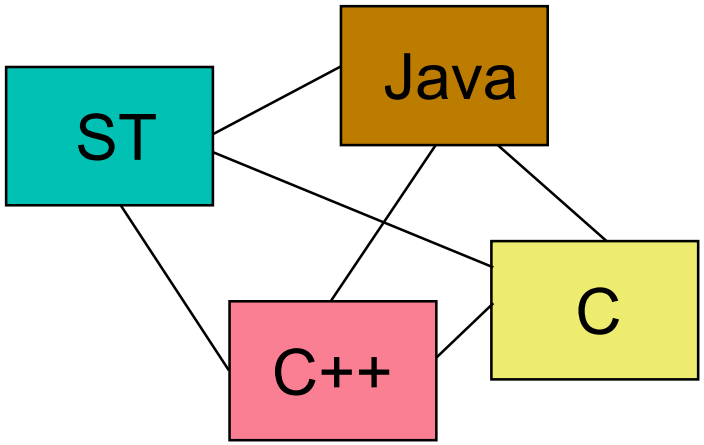
UNIVERSITÄT SALZBURG

Our definition of the term (software) component

A piece of software with a programming interface

Wiring standards

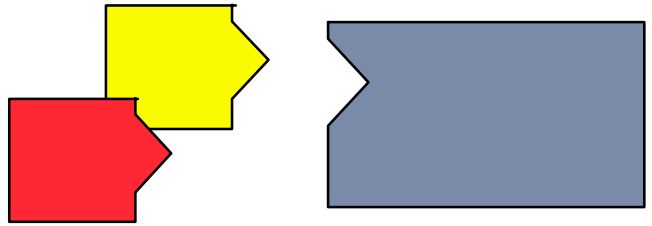
Interoperability problem:



=> wiring standards

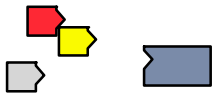
Filling the gap

Mega components (SAP, DB systems, operating systems)



only a few medium-sized components exist so far

very small components (GUI components, etc.)



Mechanistic view

Currently software components assembly requires exact matching of interfaces:

Adaptive architectures

Alternative: components configure themselves automatically through testing & fitting.

?

Sources of inspiration:

- Sun's Jini, Microsoft's .NET
- agent technology
- ontologies

Software development

- Custom made
  - Flexible and tailored to customer needs
  - Expensive, often too late to be productive
  - Requires major updates periodically – software revolution
- Parameterized standard software
  - Major trend toward “outsourcing”
  - Cheaper and up to date (vendor has the burden of maintenance)
  - No competitive advantage
  - Not always possible to adapt to all needs – business has to be adapted
- Component software provides a middle path – software evolution

Software Components

- Requirements - Sufficient variety and Quality
- First component vendors shape the market (e.g. Windows CE)
- Software => blueprints for products
- Software IC failed to capture the most distinctive aspect of software as a metaproduct.

Benefits from using components

- Process of component assembly allows significant customization of standard components.
- Individual components can be custom-made for specific requirements
- No more massive upgrade cycles (*evolution replaces revolution).*
- Modeling advantages of OO technology are of value when constructing a component.

- Software based on components benefits from combined productivity and innovation of all component vendors

Caveats

- General confusion between abstractions and instances
- Distinction between class and object is frequently omitted
- Mathematical modeling fails to capture the engineering and market aspects of component technology.
- Software technology is an engineering discipline
- Formally everything can be done without components
- Reuse, time to market, quality and viability concepts are greatly diminished.

Market versus Technology

Market vs Technology

- Market is a vital issue for component design in the way to success. Technology is just a basic support for the market. But without a market, even the best technology cannot survive. (C. Szyperski)

- Imperfect technology in a working market is sustainable; perfect technology without any market will vanish.

  (C. Szyperski)

Market vs Technology

- Market
  - Provide enough supply for estimated demand and create extra demand to maintain a good market.
  - Component warehouses
- Technology
  - Not adequately developed, especially regarding to the third-party component integration.
  - Modular checking, components should interoperate correctly and safely
  - Performance issues

Standards

Standards

- Component standards - essential for the development and success of component markets
- No component could address all needs in all environments
- Specify the required “interfacing” between a set of components as needed by clients and vendors.
- Standards
  - First introduced are vendor specific
  - Neutral organization - no proprietary solution

Wiring standards for components

- Components need to be interconnected to be useful
- A common standard is desirable, but not essential
  - as long as the market is large enough
  - adapters can be made to solve “wiring problems”
- Standards can coexist and also compete but many cannot survive in the long run.
- Today most standardization efforts are either at the “wiring” level or at the intra-component level.

Foundations

Definition of a component

- A component is a
  - **binary unit** of
  - independent
    production, acquisition and deployment
  - that interacts to form a functional system

    (C. Szyperski)

Component characteristics

- Units of independent deployment
  - Separated from the environment and from other components
  - Never deployed partially
  - Hidden construction details
- Units of composition
  - encapsulate implementation
  - interaction through well-defined interfaces
- No persistent state, attributes not influencing functionality

Object characteristics

- unit of instantiation
- may have stae and this can be externally observable
- encapsulates its state and behavior