

Software components

Content

- Introduction
- Attributes of software components
- White-box vs. Black-box models
- Costs
- Services as Components
- Examples
- Conclusions

Introduction

Historical meaning of components:

- Interchangeable parts in industry chains
 - Small components could be fabricated by third-party companies, then assembled altogether.

Introduction

What is a component in software?

“A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties”, definition by *Szyperski*

Introduction

What is a component in software?

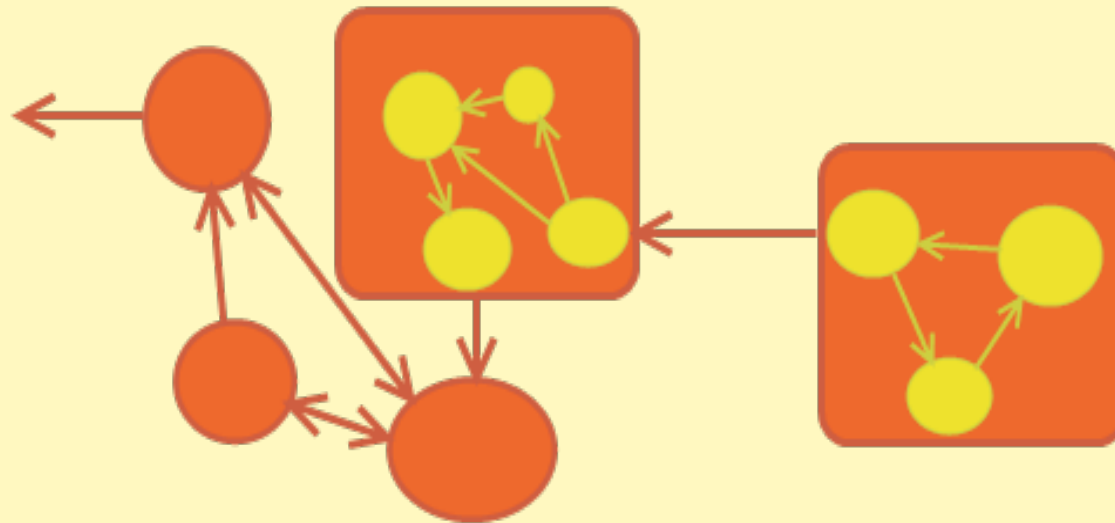
“A component is a software element (modular unit) satisfying the following conditions:

1. It can be used by other software elements, its ‘clients’.
2. It possesses an official usage description, which is sufficient for a client author to use it.
3. It is not tied to any fixed set of clients.”, definition by Meyer

Introduction

What is a component in software?

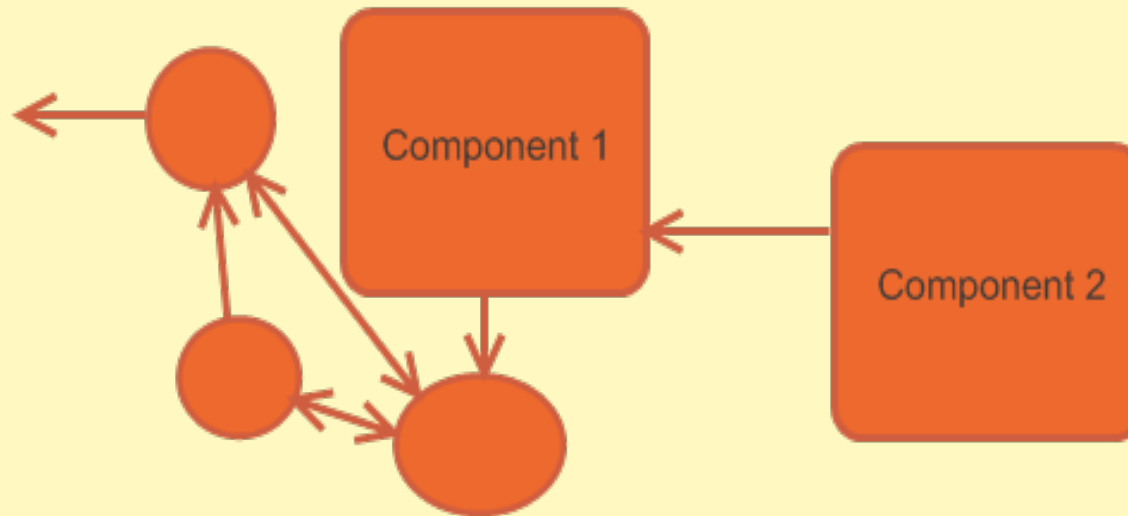
- An element of software



Introduction

What is a component in software?

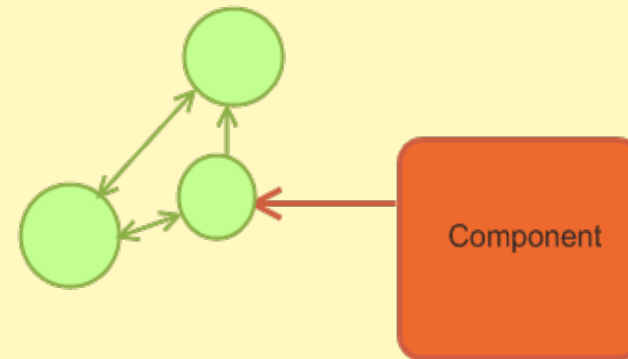
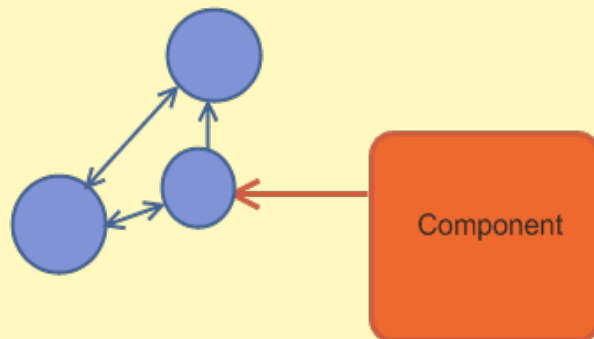
- An element of software
- A unit of composition



Introduction

What is a component in software?

- An element of software
- A unit of composition
- Can be composed in other systems



Introduction

Component-based software engineering

- Aims for building software systems by gluing components altogether.
- Doesn't want to represent real world (like OO does).

Introduction

Component-based software engineering

- Reduction of development time
- Aims for reusability
- Might reduce our costs
- Might reduce the mainainance needed

Introduction

Approaches to component-based software engineering:

- JavaBeans
- CORBA
- SOFA
- Koala

Attributes of software components

Coupling

- Dependencies of the component with other components/libraries
- Always look for low coupling → Avoid unnecessary dependencies

Attributes of software components

Granularity

- Indicates the amount of functionality in one component.
- Can be classified as fine, coarse or something in between.

Attributes of software components

Infrastructure

- Restrictions about what does the component need to run.
- e.g. A Java component needs a Java Virtual machine to run.

Attributes of software components

Licenses

- Restrictions about how to use/modify the components
- (c), cc, GLP, LGPL, BSD, ...

White-box vs. Black-box

Black-box Model

- Components that can't be modified
- Source code is not accessible
- Its behaviour can only be modified through the input parameters

White-box vs. Black-box

White-box Model

- Components that can be modified
- Source code is accessible
- Usually from the same organization or under open licenses.

White-box vs. Black-box

Black box → No customization, easier to update or modify

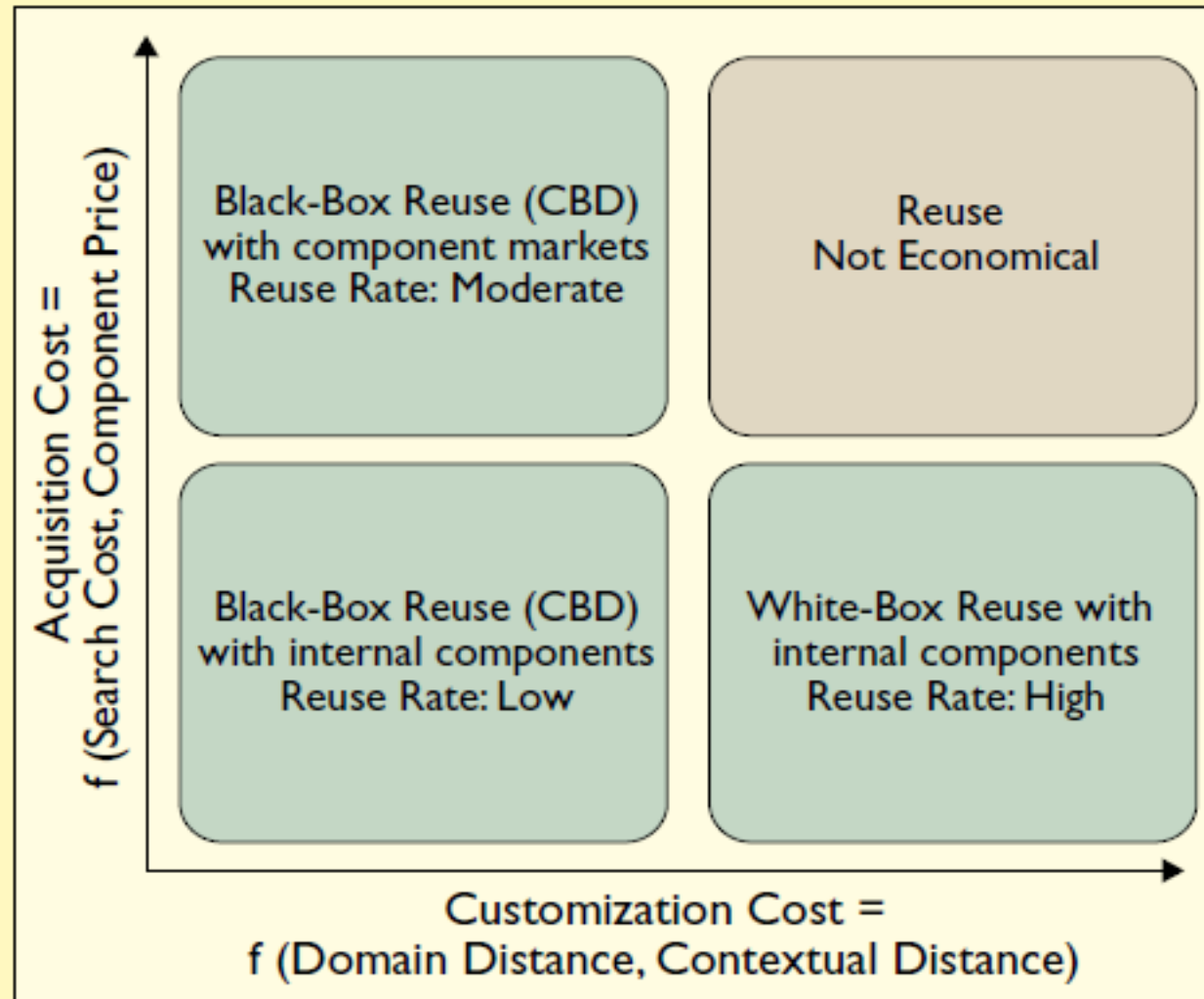
White box → Customization allows to fit it in our solution, but makes it difficult to maintain

Costs

We divide this into two kind of costs:

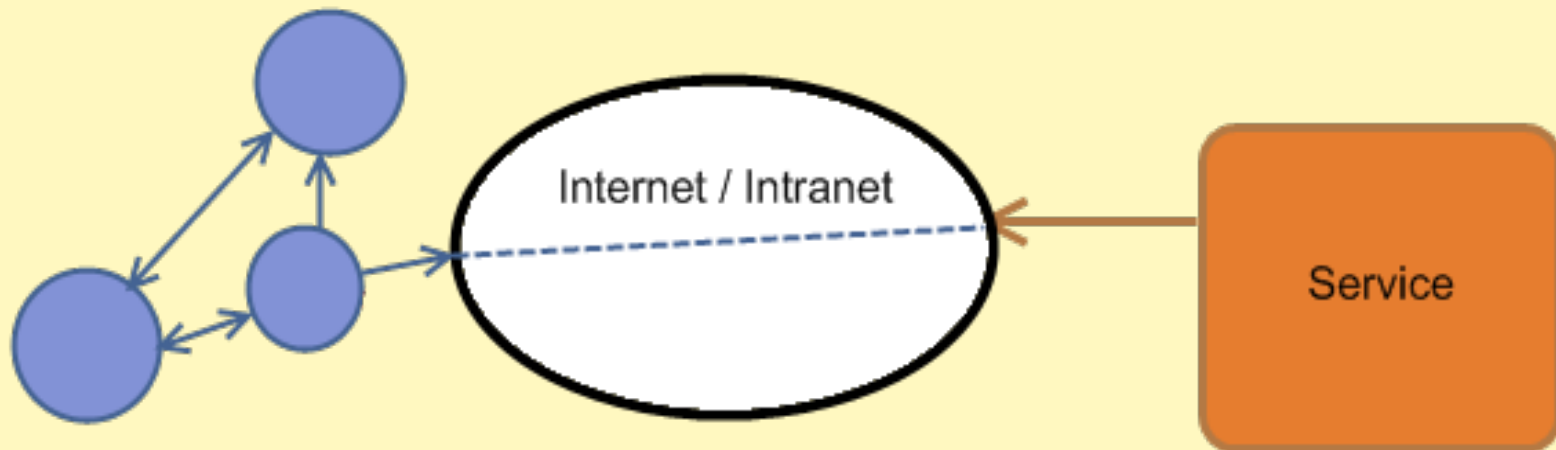
- Acquisition costs → Cost of searching + Cost of purchasing
- Customization costs → Cost of adapting our solution to the component + Cost of modifying the component

Costs



Services as components

- Services might act as remote components
- Nowadays network makes it easier



Services as components

Advantages:

- No need to include the whole component into our solution
- Back-end not exposed
- Service can be updated

Services as components

Standard protocols for using services:

- SOAP
- REST
- Java RMI
- .Net Remoting

Examples

WebKit

- Component for HTML and CSS displaying
- Open Source project
- (<http://webkit.org/>)



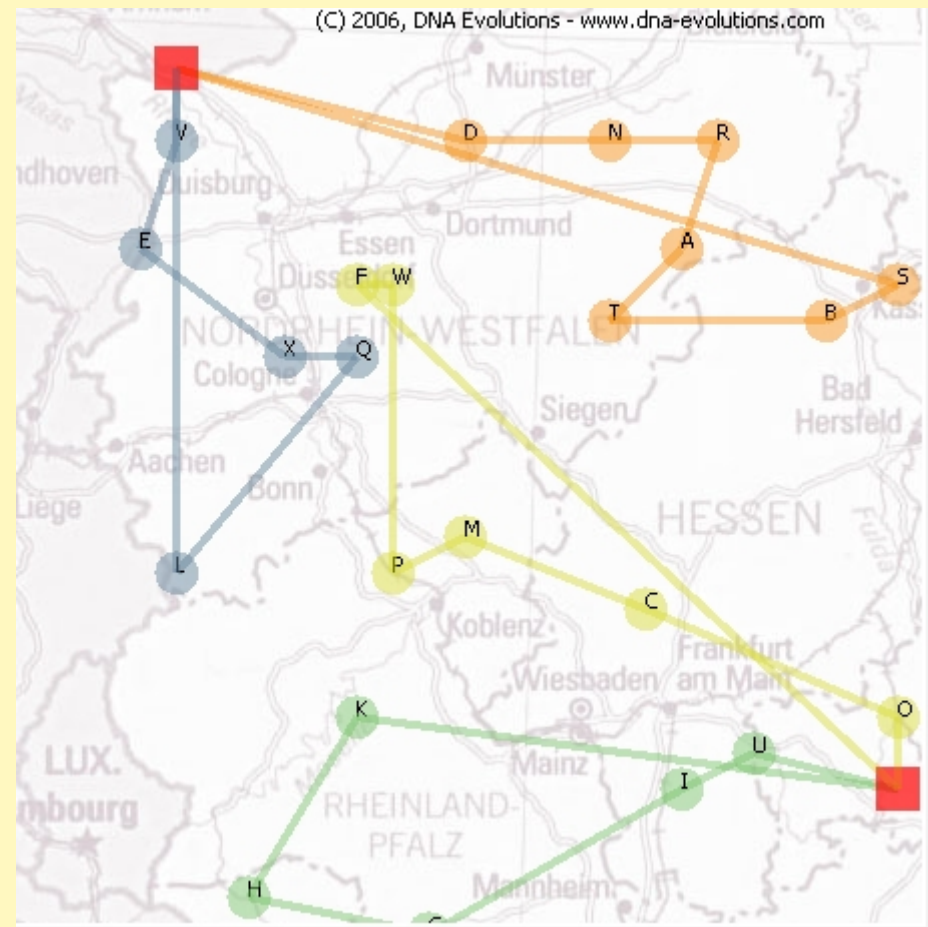
Examples

	Webkit
License	LGPL/BSD
Granularity	Medium
Extensible	Yes
Specificity	Low
Infrastructure	C++
White-box?	Yes
Costs	Free

Examples

JOpt

- Component for vehicle routing optimization
- Use of very specific techniques for optimization
- (www.dna-evolutions.com)



Examples

	Webkit	JOpt
License	LGPL/BSD	©
Granularity	Medium	Medium
Extensible	Yes	No
Specificity	Low	High
Infrastructure	C++	J2EE / .NET
White-box?	Yes	No
Costs	Free	500€ - 10000€

Examples

*Musicbrainz Mp3
information retrieval*

- *Service for retrieving
Mp3 information from
a database*
- ([http://musicbrainz.org/
doc/Web_Service](http://musicbrainz.org/doc/Web_Service))



Examples

	Webkit	JOpt	MusicBrainz WebService
License	LGPL/BSD	©	©
Granularity	Medium	Medium	Fine
Extensible	Yes	No	No
Specificity	Low	High	Low
Infrastructure	C++	J2EE / .NET	REST
White-box?	Yes	No	No
Costs	Free	500€ - 10000€	Free

Conclusions

- Componentization can make our development easier, cheaper and faster
- Choose wisely between white-box and black-box models
- Look for customization/aquisition costs trade-off
- Take into account the use of external services

References

- [1] Messerschmitt, David G.(2007). Rethinking components: From hardware and software to systems. Multi-Campus: Retrieved from: <http://escholarship.org/uc/item/65t2v4cr>
- [2] T. Ravichandran and Marcus A. Rothenberger (2003), "*Software reuse strategies and component markets*", Communications of the ACM
- [3] Clemens Szyperski and David G. Messerschmitt, "The Flexible Factory", Software Development, december 2003
- [4] *Michael Stal*, "Web Services: Beyond Component-Based Computing", Communications of the ACM October 2002/Vol. 45, No. 10
- [5] Kung-Kiu Lau and Zheng Wang, "Software Component Models", October 2007, IEEE Transactions on software engineering, VOL. 33, NO. 10
- [6] Kung-Kiu Lau and Zheng Wang, "A Taxonomy of Software Component Models", School of Computer Science, the University of Manchester
- [7] Parastoo Mohagheghi & Reidar Conradi, "Quality, productivity and economic benefits of software reuse: a review of industrial studies", Springer Science + Business Media, LLC 2007
- [8] "An Overview of the Koala Component Model", Distributed Systems Research Group (), Charles University Prague, Faculty of Mathematics and Physics

End