Component-Based Software Engineering	**Exercise Sheet No. 5**
Dipl.-Inf. Florian Heidenreich Office Hours: Wednesdays, 1430–1530 hours INF 2080 `http://st.inf.tu-dresden.de/teaching/cbse`	Software Technology Group Institute for Software and Multimedia Technology Department of Computer Science Technische Universität Dresden 01062 Dresden

Software Architecture

Task 1: Keyword in Context

In 1972, David Parnas proposed the following problem [1]:

> The KWIC (Key Word In Context) index system accepts an ordered set of lines; each line is an ordered set of words, and each word is an ordered set of characters. Any line may be "circularly shifted" by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.

In this exercise, we are going to discuss some ways of realising such a system.

1a)

Develop a rough system architecture for this problem, where a central master control component invokes components for the individual processing steps. How can you share data between the processing steps?

Use boxes and lines to present your architectural design. Provide a legend of what your boxes and lines stand for.

1b)

Develop a rough system architecture for the same problem, but this time use a pipes-and-filters approach, where the individual processing steps are performed by filter components connected by appropriate pipes.

Use boxes and lines to present your architectural design. Provide a legend of what your boxes and lines stand for.

1c)

Compare your solutions from above. In particular, discuss properties such as performance, evolvability/flexibility, robustness, reusability of individual components, etc. of the resulting application.

Bibliography

[1] David L. Parnas: On the criteria to be used in decomposing systems into modules. *Communications of the ACM,* 15(12):1053–1058, December 1972.
`http://doi.acm.org/10.1145/361598.361623`

Task 2: KWIC in Java

ArchJava (www.archjava.org) is a pre-processor for java that allows to encode architectural information in Java programs and restricts communication between components to the channels defined by explicit connections between components. Download and install ArchJava.

2a)

How are components, ports, and connections represented in ArchJava?

2b)

Implement the pipe-and-filter solution for the KWIC example from the last task using ArchJava.

Task 3: Starting to Fly with Wright

Download and read [1]. Do not necessarily read the complete paper, just as much as you need to complete the tasks below.

Use WRIGHT to model a simple application consisting of a sensor component, which periodically sends out sensor values, and an alarm component which reads data and sends out an alarm signal, if the data is higher than a certain value. Connect the two components using a pipe connector. The following figure gives a graphical overview of the application structure.

3a)

Write a WRIGHT specification of the components, connector, and the complete system, but leave out the formal specification of ports, roles, components, and glue. Instead, use comments explaining informally what happens in each element.

3b) *

Enhance your specification from the previous subtask by providing CSP-specifications of the ports, roles, components and glue.

Bibliography

[1] Robert Allen and David Garlan. *The Wright Architectural Specification Language*, Technical Report CMUCS-96-TB, School of Computer Science, Carnegie Mellon University, Pittsburgh, Sept. 1996. http://citeseer.ist.psu.edu/allen96wright.html