

33. Composition of Stream-Based Tools

1

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und

Multimediatechnik

<http://st.inf.tu-dresden.de>

Version 12-0-9, 07.12.12

- 1) Extension of Stream-Based Tools
- 2) and XML-Mashups
- 3) Aspect-Oriented Extension
- 4) EAI-Decomposition of Tools
- 5) EAI-Based Composition of Tools



Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Aßmann

Literatur

2

- ▶ Informatik Forum <http://www.infforum.de/>
- ▶ Structured Analysis Wiki <http://yourdon.com/strucanalyzer/wiki/index.php?title=Introduction>
- ▶ De Marco, T.: Structured Analysis and System Specification; Yourdon Inc. 1978/1979. Siehe auch Vorlesung ST-2
- ▶ McMenamin, S., Palmer, J.: Strukturierte Systemanalyse; Hanser Verlag 1988
- ▶ Raasch, J.: Systementwicklung mit Strukturierten Methoden; Hanser Verlag (3.Aufl.) München 1993
- ▶ [Altinel07] Mehmet Altinel, Paul Brown, Susan Cline, Rajesh Kartha, Eric Louie, Volker Markl, Louis Mau, Yip-Hing Ng, David E. Simmen, and Ashutosh Singh. DAMIA - A data mashup fabric for intranet applications. In C. Koch, et.al., editors, VLDB, pages 1370-1373. ACM, 2007.



33.1 Extension of Stream-Based Tools by DFD

3

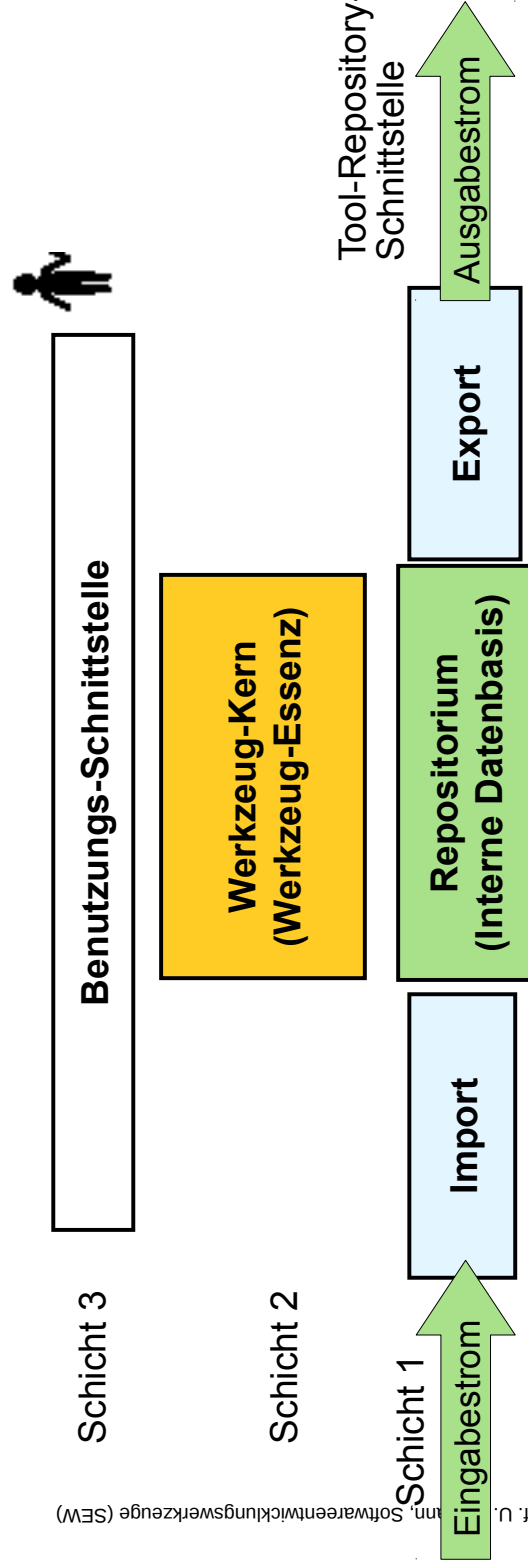
And composition of stream-based tools



Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Aßmann

Rpt. Architektur eines datenflussgesteuerten, strom-basierten Werkzeugs

- ▶ Arbeit wird stückweise erledigt; meist pro gelesenem Datenpaket.
- ▶ Eine DFD- oder Workflow- Sprache verknüpft (komponiert) die Werkzeuge durch ein DFD oder Workflow (Mashup) zu komplexeren Werkzeugen



Prof. U. Aßmann, Softwareentwicklungswerkzeuge (SEW)

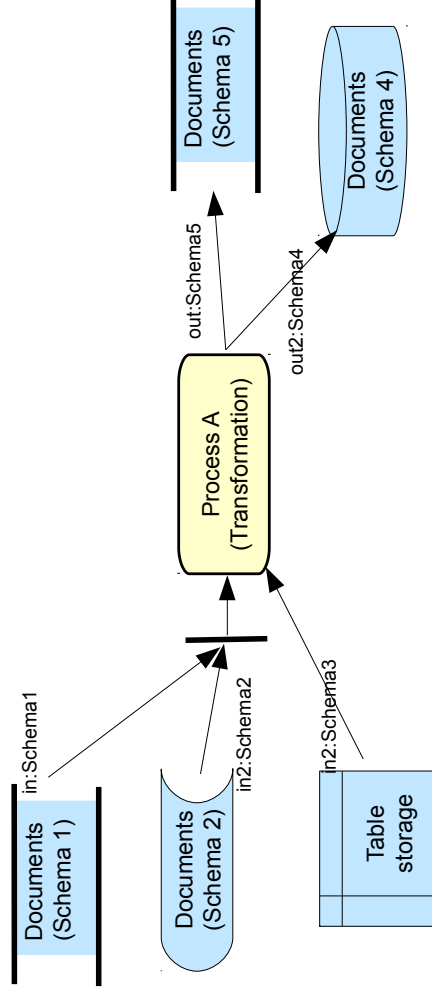


Stream Merging

5

- ▶ The architecture of stream-based tools can be described by DFD or (Web-)Mashups
- ▶ Three operations are important:
 - **Input stream synchronization:** does a process read from input channels synchronously or alternatingly?
 - **Input stream merge:** how does a process merge two input channels?
 - **Output stream replication:** does a process replicate output data in different streams or produce different output formats?

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)

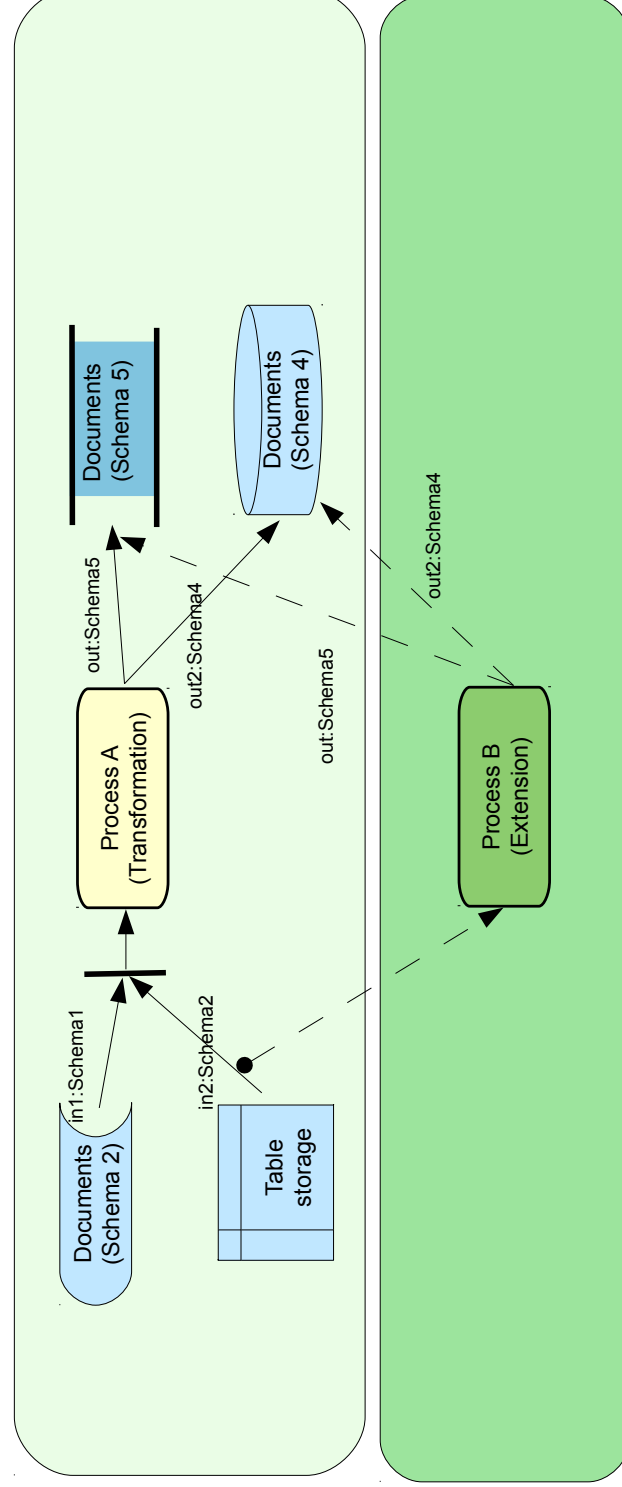


Tool Extension by Stream Duplication and Asynchronous Merge

6

- ▶ DFD are easily extensible, because input streams can be replicated to deliver their content into the processes of the extension (extension listening on stream of core)
- ▶ Output streams of extensions can write asynchronously into output storages

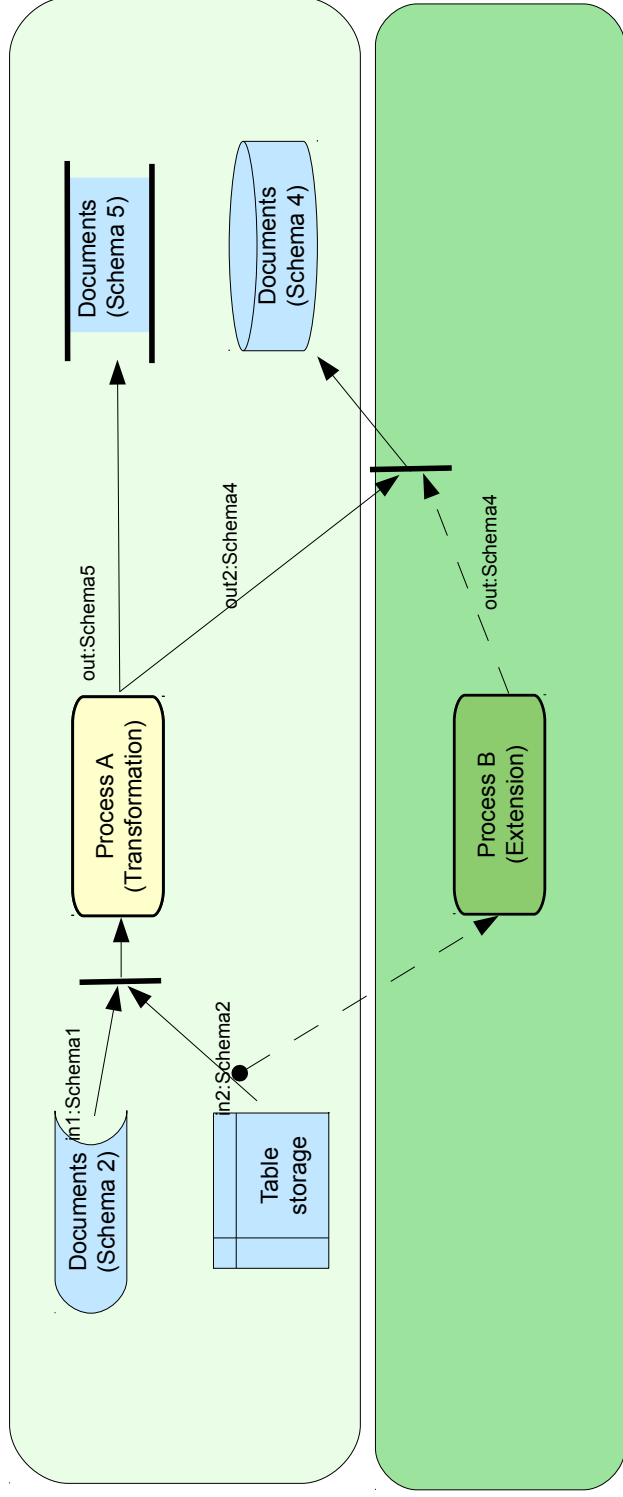
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Synchronizing Extension of Core Tool

7

- ▶ Output streams of extensions can write synchronously into output storages by adding new synchronizing activities guarding output storages



Prof. U. Alsmann, Softwareentwicklungswerkzeuge (SEW)



33.2. Extensible Stream-Based Tools: DQL und DTL in DFD-Mashups

8

Ex.: Technical Space Treeware-XML

XML Mashups are special DFD

The example can be transferred to Graphware or
Grammarware using other DQL and DTL

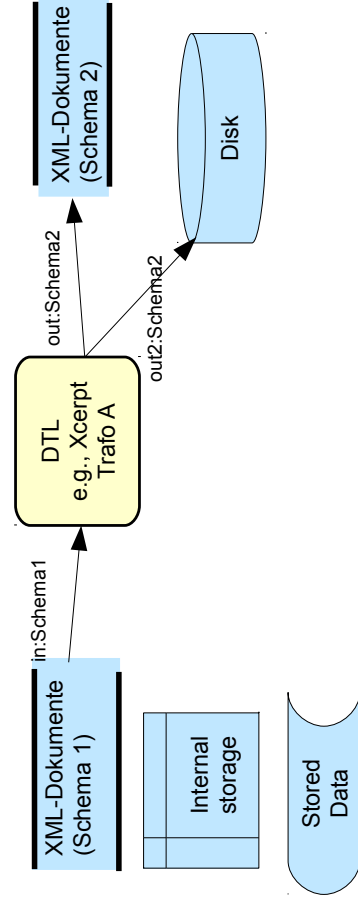


Use of DQL and DTL in DFD (e.g., Mashups)

9

- ▶ DTL and DQL (Xquery, Xcerpt and others) can be employed as generators and transformers in DFD
 - A DDL describes the types of data on the streams (types, schemata)
 - String rewrite systems can be used to specify processes if streams transport texts
 - Term rewrite systems can be used to specify processes if streams transport trees
 - XML rewrite systems: With XML and XSD, Xcerpt can be used
 - Graph rewrite systems can be used if streams transport graphs
- ▶ Mashups are easily extensible, because channels can be replicated and extended
- ▶ Mashups are extremely important for extensible tools

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)

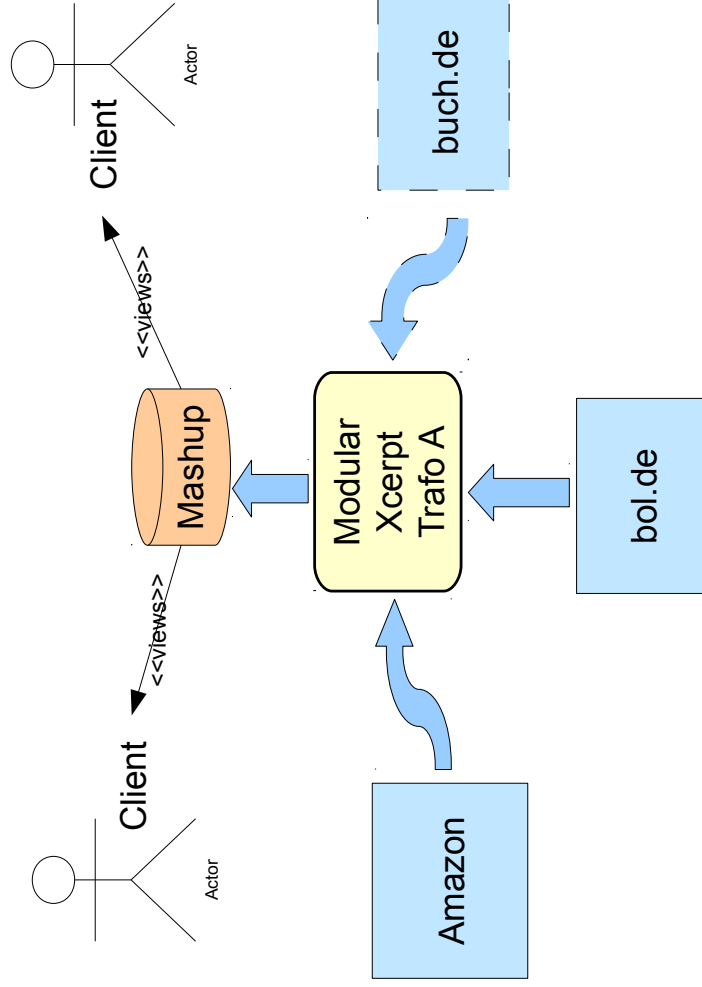


XML-Mashups with Modular Xcerpt

10

Use Modular Xcerpt for creating a CD mashup of our favourite music LPs

- "mashing-up" freely available data from online stores
- easily extensible with new sources or processing steps



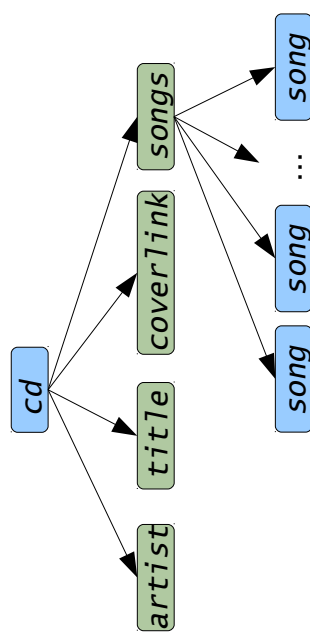
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Mashups with Modular Xcerpt

- ▶ First we need a data structure for CDs, so that we can use it for our virtual store of aggregated data
- ▶ Model with Xcerpt data terms (XML trees)

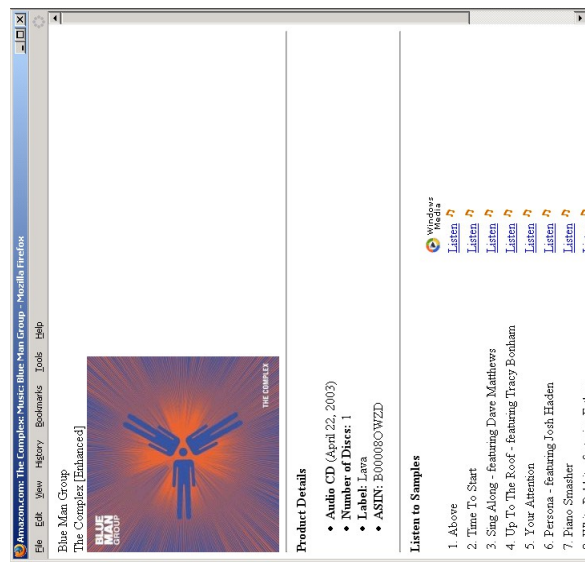
```
cd [
  ▶ artist,
  ▶ title,
  ▶ coverlink,
  ▶ songs [
    ▶ song, song ... song
  ]
]
```



Mashups with Modular Xcerpt

- ▶ Next step: creating import modules to aggregate data from our sources

```
MODULE AmazonQuery
CONSTRUCT
public cd [
  artist [ var ARTIST ],
  title [ var TITLE ],
  coverlink [ var COVERLINK ],
  songs [
    all song [ var SONGTITLE ]
  ]
]
FROM
public html [ head [ [ ] ],
              body [ [
                var ARTIST, br,
                var TITLE, br,
                img {
                  attributes {src { var COVERLINK }}
                },
                table [ [
                  tr [
                    th [ [ ] ]
                  ],
                  tr [
                    td [ var SONGTITLE ],
                    td [ [ ] ]
                  ]
                ]
              ]
            ]
          ]
        ]
      ]
    ]
  ]
END
```



(Example HTML Source)

Mashups with Modular Xcerpt

13

- ▶ Import modules are independent from a concrete source
 - pass the resource locations to the modules
 - collect all data from modules by introducing a virtualroot node (dummy)

```
MODULE MainProgram
IMPORT /import/AmazonQuery.mxcerpt AS Amazon
IMPORT /import/BuchdeQuery.mxcerpt AS BuchDE
CONSTRUCT to Amazon (
  var DATA
)
FROM in {
  resource { "file:data/amazon-blue_man_group-
the_complex.html", "xml" },
  var DATA
}
END
CONSTRUCT to BuchDE
...
END
```

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)

```
// Filling variable CDINFO with
// dummy virtual root node
CONSTRUCT
  virtualroot [ all var CDINFO ]
FROM in Amazon (
  var CDINFO -> cd [ [ ] ]
)
END
CONSTRUCT
  virtualroot [ all var CDINFO ]
FROM in BuchDE (
  var CDINFO -> cd [ [ ] ]
)
END
```

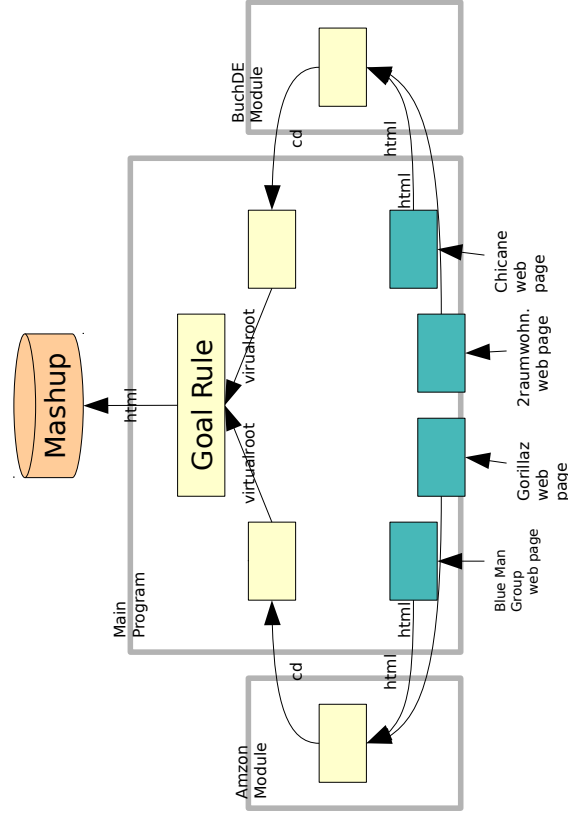
Mashups with Modular Xcerpt

14

- ▶ Construct rules “mash up” the data – create a new webpage
 - in Xcerpt a goal rule must be specified (program entry point)

```
GOAL
out {
  resource { "file:mashup.html", "xml" },
  htm [
    head [
      title [ "Mashup" ]
    ],
    body [
      table [
        all tr [
          td [ var ARTIST ],
          td [ var TITLE ]
        ]
      ]
    ]
  ]
}
FROM virtualroot [
  cd [
    artist [ var ARTIST ],
    title [ var TITLE ]
  ]
]
END
```

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



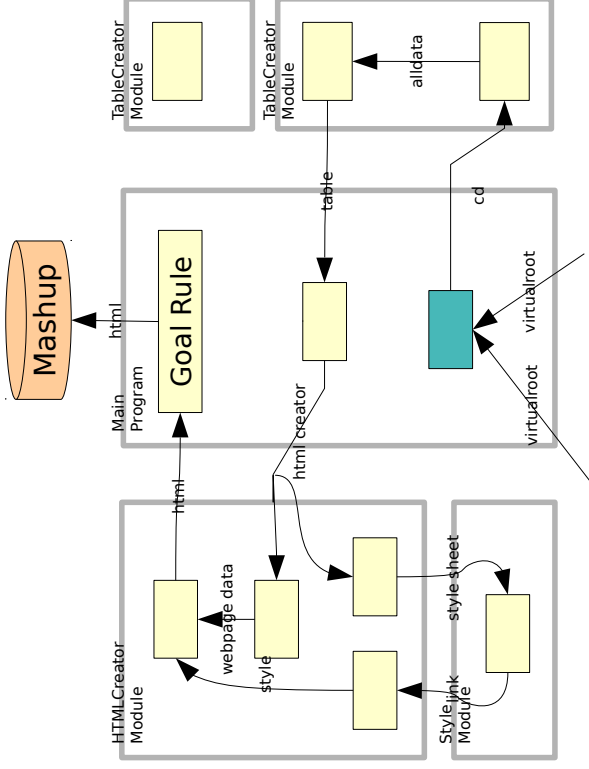
(Structure of the Modular Xcerpt program)



Mashups with Modular Xcerpt

15

- ▶ Further decomposition of program possible
 - HTML creator can be an extra module
 - Table layout and style sheet linking can be made configurable



(advanced Modular Xcerpt program)

33.3. Aspect-Oriented XML-Weaving with XML Transformations

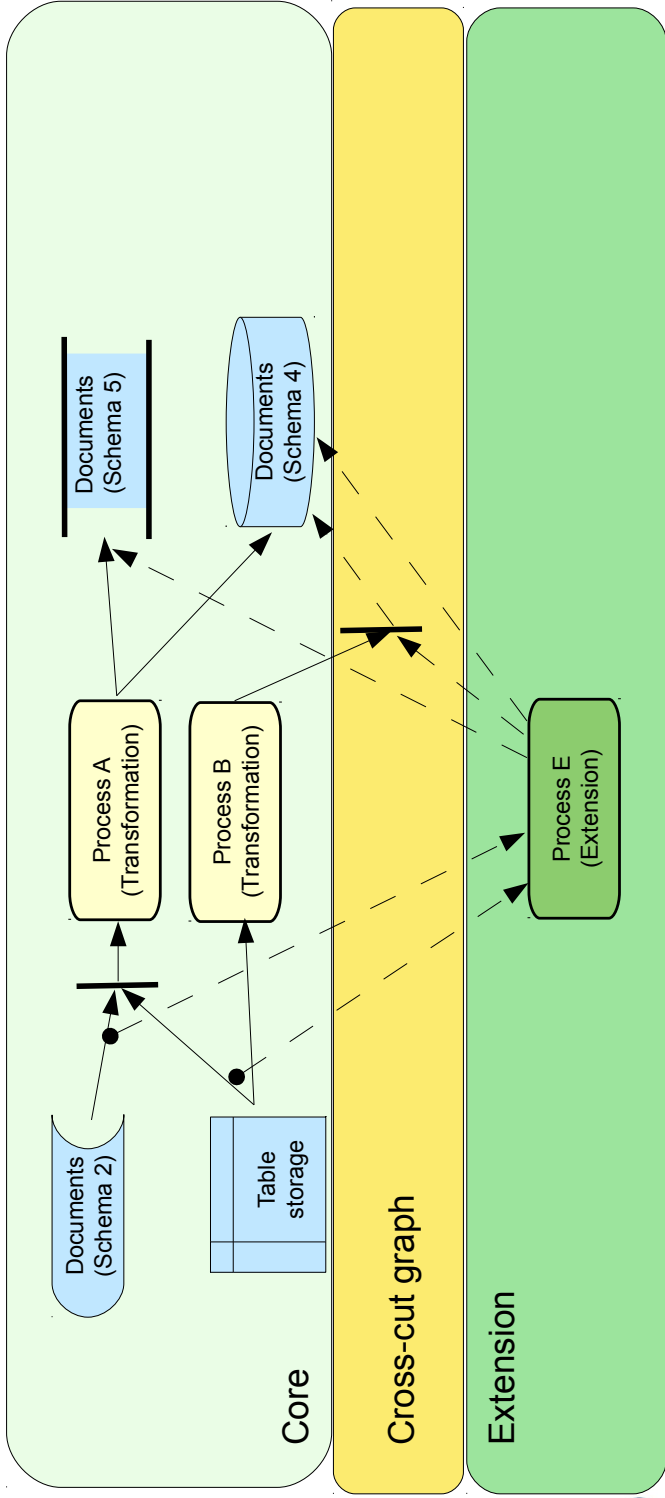
16

- For aspect-oriented extensions of DFD und Mashups

Aspect-Oriented Tool Extension by Crosscut-Graph between Core and Extension

17

- ▶ If an extension extends many places in a core (scattering), a *crosscut-graph* describes the



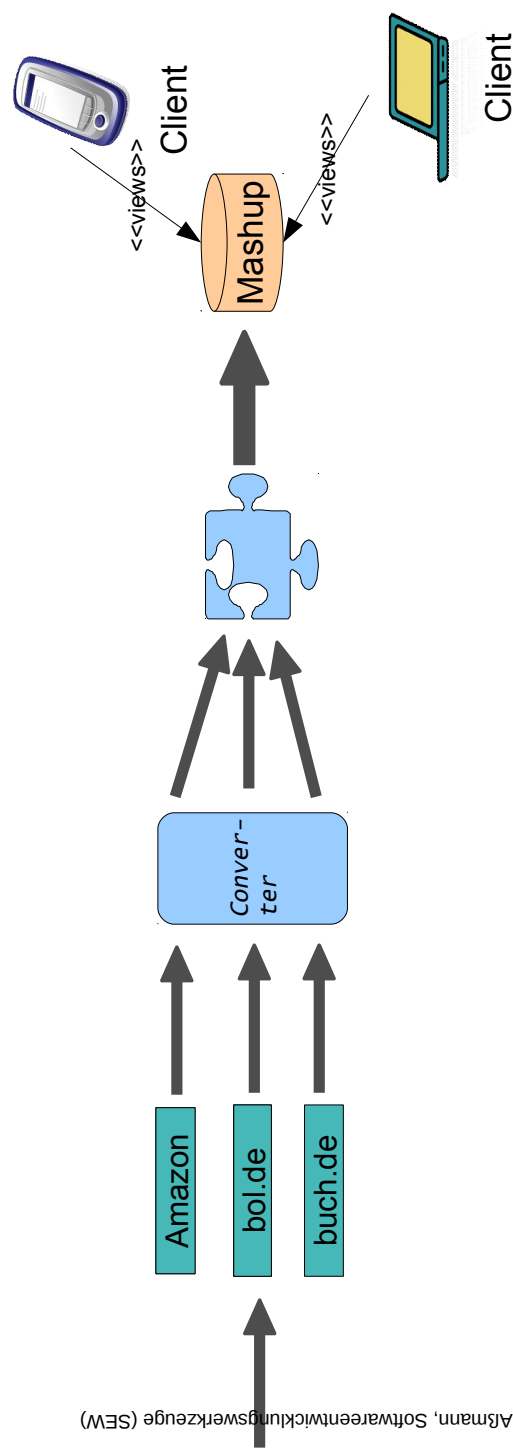
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



XML Adaptation Aspects (HyperAdapt Weaver)

18

- ▶ Xcerpt mashups induce data-flow architecture
- ▶ Mashups should be rendered for different target devices, e.g., mobiles, tablets → *Adaptation Aspects*

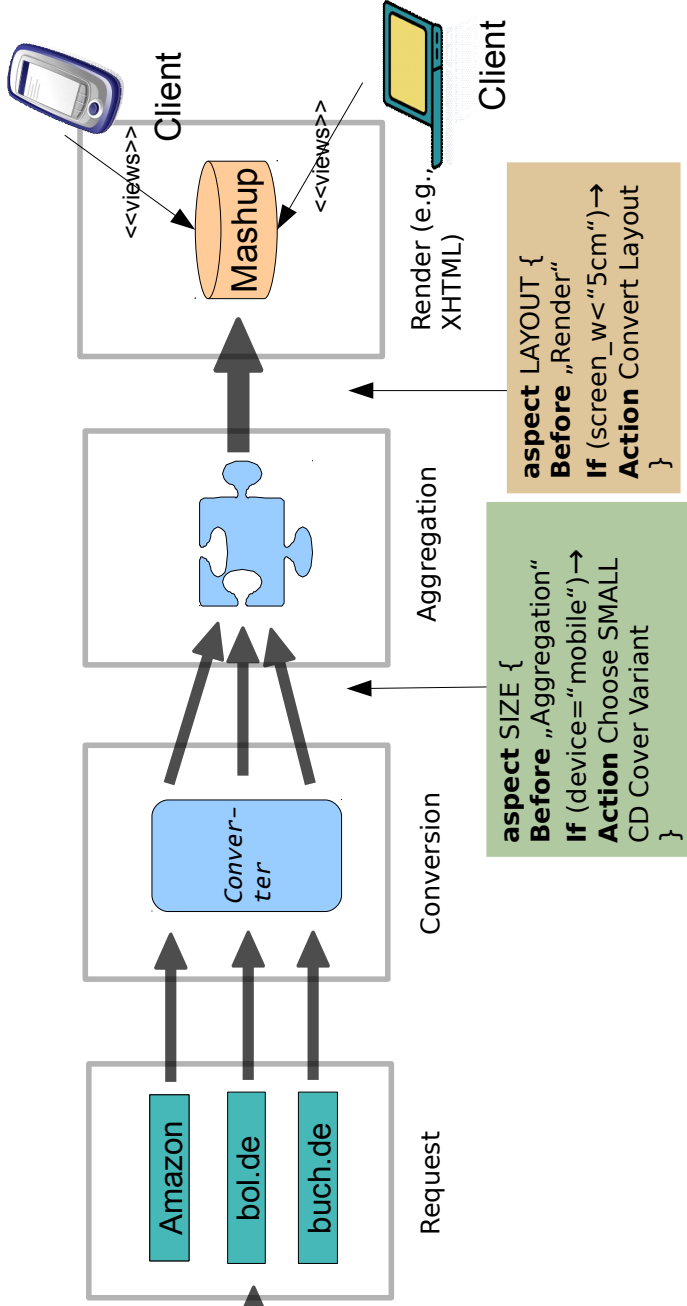


Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



XML Adaptation Aspects (HyperAdapt Weaver)

- ▶ The tool “HyperAdapt Weaver” modifies the streams by transformation: “aspect actions” are “woven” into the stream



XML Adaptation Aspects (HyperAdapt Weaver)

- ▶ Example: Virtual Storage Music Database before aggregation phase as plain XML

```

<music-database xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://music.music.xsd" xmlns="http://music">
  <album inStock="yes">
    <title>How to Be a Megastar-Live!</title>
    <artist>
      <pseudonym>Blue Man Group</pseudonym>
      <artist>
        <id>B00166GIVO</id>
        <edition>First</edition>
        <publisher>Rhino (Warner)</publisher>
        <image size="SMALL" url="...">
          <image size="MEDIUM" url="...SS500.jpg"/>
          <image size="TINY" url="...SS500_tiny.jpg"/>
        </image>
        <medium kind="CD">
          <tracks>
            <song name="Above" length="3.30" />
            <song name="Drumbone" length="3.25" />
            <song name="Time To Start" length="4.22" />
            <song name="Up To The Roof" length="4.16" />
            <song name="Altering Appearances" length="2.23" />
            <song name="Persona" length="4.12" />
            <song name="Your Attention" length="4.04" />
            <song name="Piano Smasher" length="6.01" />
            <song name="Shirts And Hats" length="4.40" />
            <song name="Sing Along" length="3.10" />
          </tracks>
        </medium>
      </album>
    </music-database>
  
```

Callout for the `image size="SMALL" url="..."` line:

```

aspect SIZE {
Before „Aggregation“
If (device=“mobile“) →
Action Choose SMALL
      CD Cover Variant
    }
  
```



XML Adaptation Aspects (HyperAdapt Weaver)

21

▶ Example: Document adaptation specified as HyperAdapt Adaptation Aspect, written in the XML-based HyperAdapt Aspect Language

- Interpreting these aspects, the weaver weaves aspect slice into streams

```
<?xml version="1.0" encoding="UTF-8" ?>
<aspect name="choose-image">
  <interface>
    <core id="core" type="http://music" />
  </interface>
  <adviceGroup>
    <scope>
      <xpath>/music:music-database</xpath>
      <before>Aggregation</before>
    </scope>
    <advicees>
      <chooseVariant>
        <pointcut>/music:album/music:image[1]</pointcut>
      </chooseVariant>
    </advicees>
  </adviceGroup>
</aspect>
```

Prof. U. Alsmann, Softwareentwicklungswerkzeuge (SEW)



SMALL



LARGE



TINY

(Pictures from amazon.de)

33.4 Essential Decomposition of Tools

22

Development with DFD

23

- ▶ **Prozess-oriented Refinement/Decomposition** refines processes/activities step by step into smaller processes (divide-and-conquer)
 - One dimension of decomposition
- ▶ **Essential Decomposition** uses aspect-oriented decomposition and distinguishes three aspects: [McMenamen/Palmer]
 - Essence (E): essential processes, activities, storage. Functionality that cannot be stripped
 - Administration (A): administrative activities (for consistency checking of data in internal storages; for contract checking of processes on input and output streams)
 - Infrastructure (I): activities for communication and adaptation to platform (platform-specific details)

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)

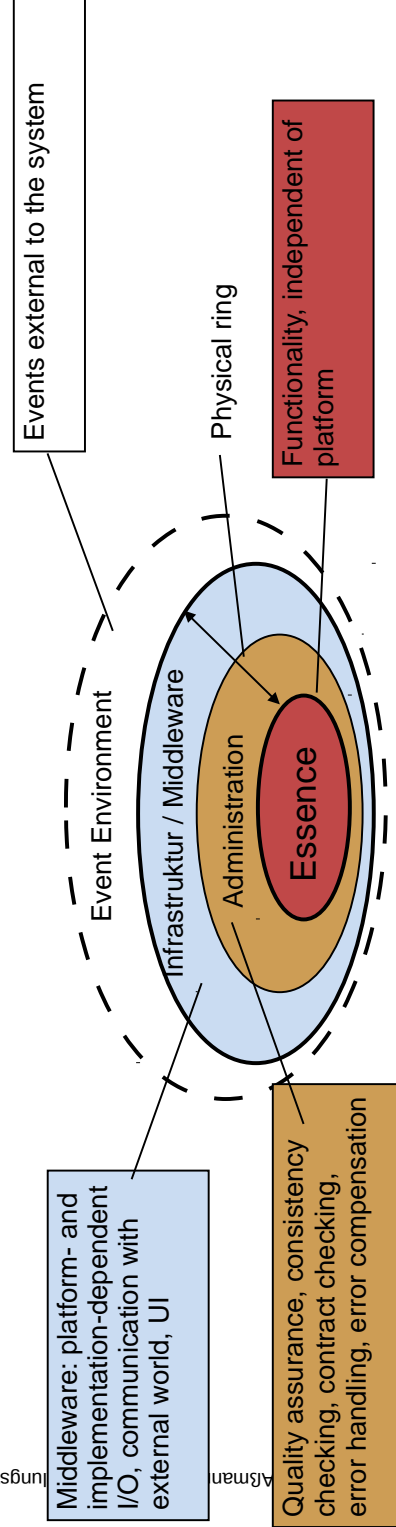


EAI-Decomposition

24

- ▶ **Essential decomposition (EAI decomposition)** separates the **essence** of a system from implementation-specific parts (**infrastructure**) and quality assurance (**administration**).
 - Essence assumes perfect technology [McMenamen/Palmer]
 - Processes do not need time, storage with unlimited capacity

Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



EAI-Decomposition of DFD-Based Tools

25

- ▶ With DFD, the decomposition into EAI-aspects (Essence, Administration, Infrastructure) is simple: by graph slicing
- ▶ EAI-aspects of a tool:
- ▶ Essence of a tool:
 - Functionality assuming perfect technology
- ▶ Administration of a tool:
 - Constraint checker, wellformedness checker on internal repository, contract checkers on streams
- ▶ Infrastructure of a tool:
 - Parser, tree constructor (import)
 - Pretty printer, code generator (export)

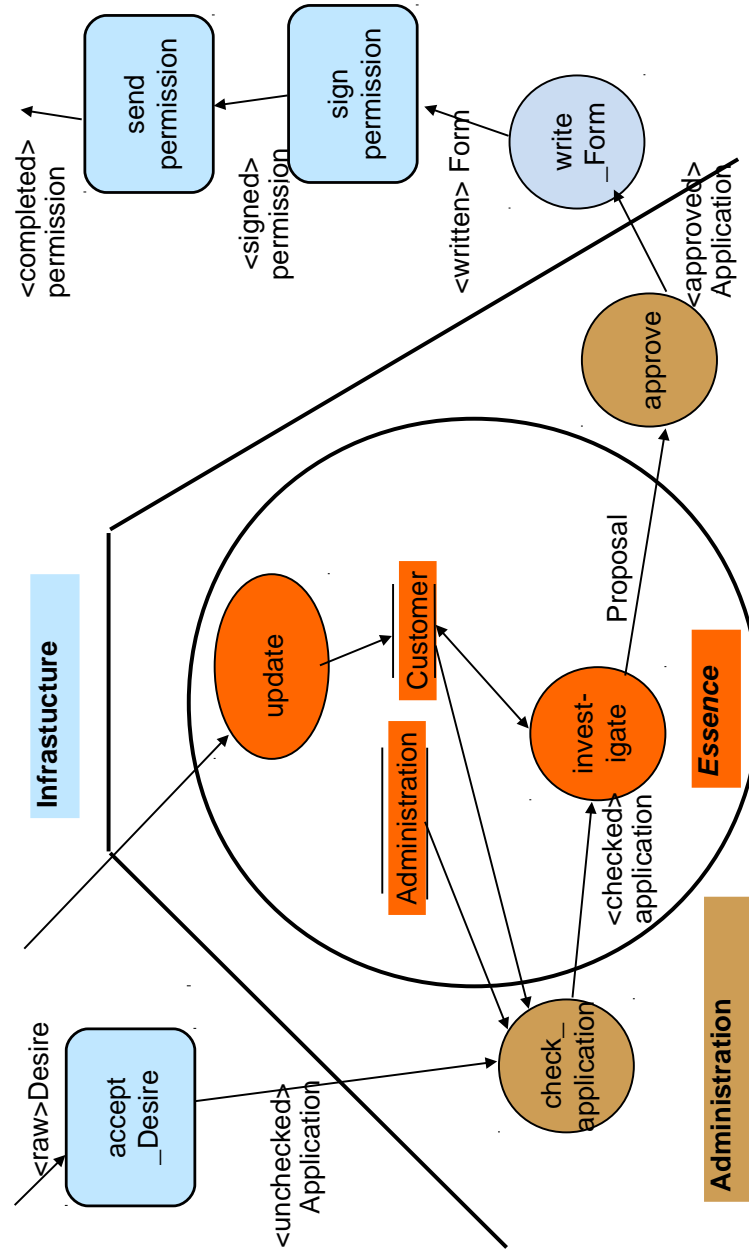
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Ex. EAI-Decomposition of a Process of a Tool “Task Management System”

26

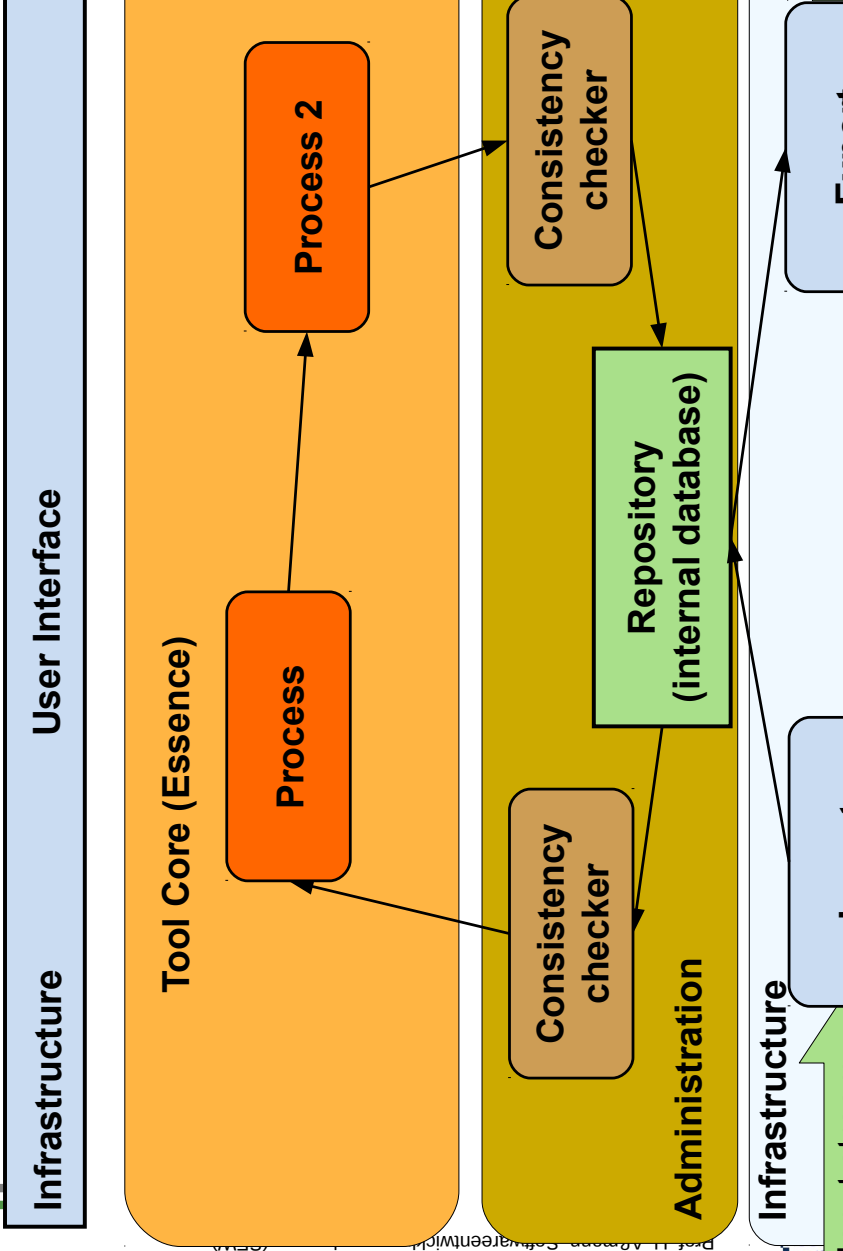
- ▶ EAI was invented for the Structured Analysis of applications, but can be used for tools



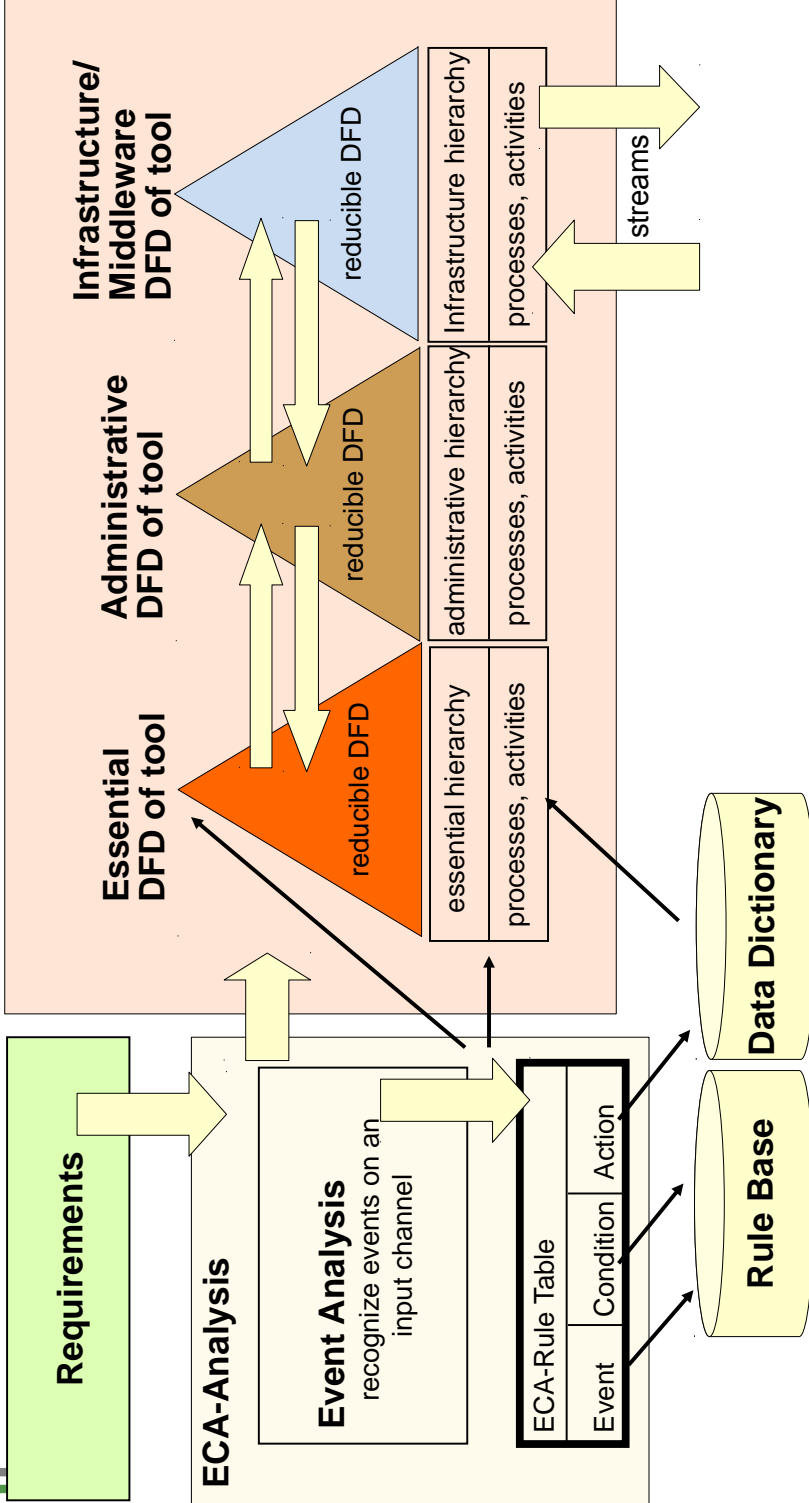
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



EAI-Decomposition of a Stream-Based Tool



Essential Structured Analysis for Tools



33.5 Composition of Stream-Based Tools

29



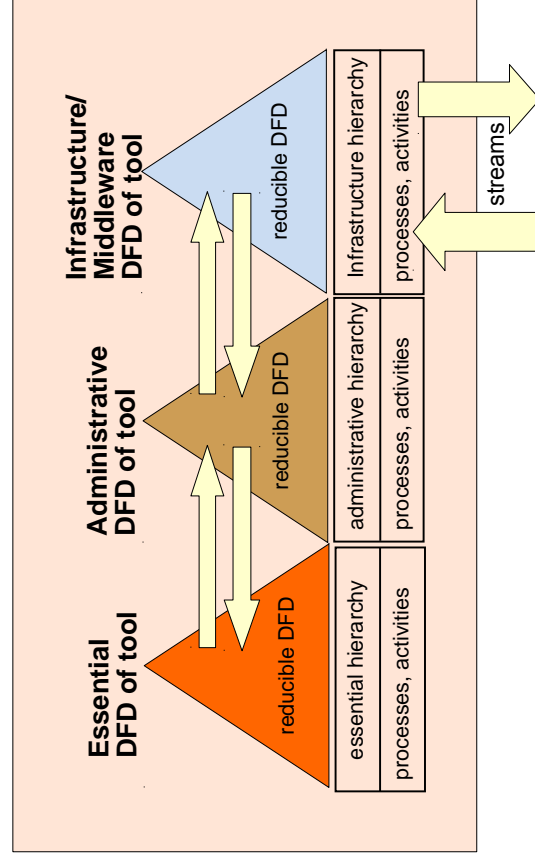
Softwareentwicklungswerkzeuge (SEW) © Prof. Uwe Aßmann

Process for Composition of Stream-Based Tools

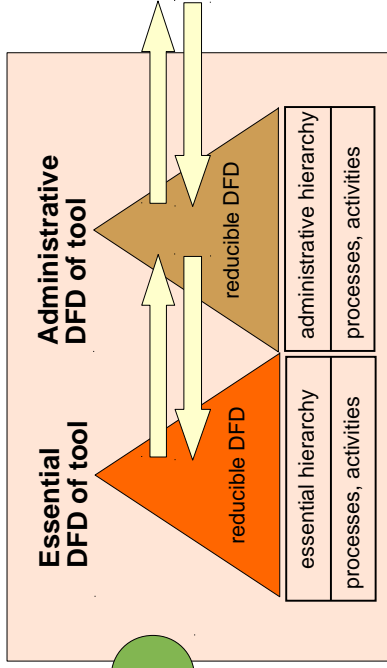
30

- 1) Strip the DFD Strip Essence of Administration and Infrastructure:
 - 1) remove parser, printer, GUI, etc.
- 2) Compose the essential DFD of the tools
 - Extend and merge streams with the same schema (respect typing)
 - Extend core tools by asynchronous merge of output streams
 - Extend core tools by synchronous merge of output streams
 - Use aspect-oriented extension with cross-cut-graphs
- 3) Add Administration
- 4) Add Infrastructure to the composed DFD

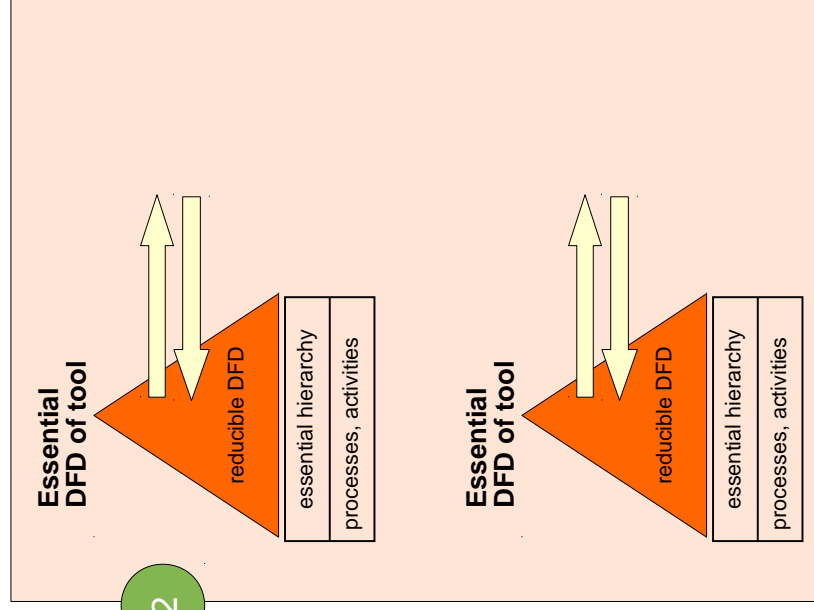
Prof. U. Aßmann, Softwareentwicklungswerkzeuge (SEW)



1) Strip Infrastructure 2) Strip Administration



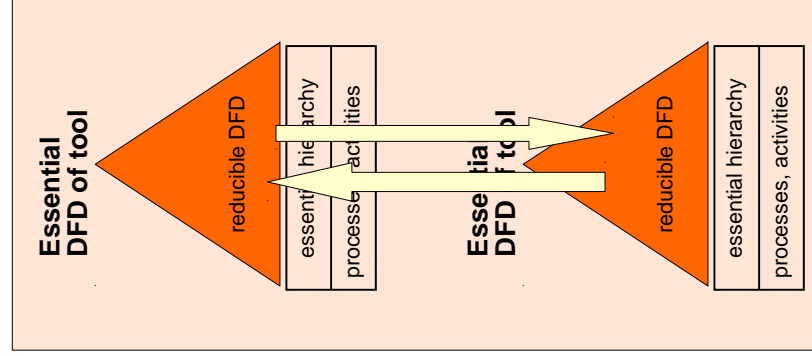
1



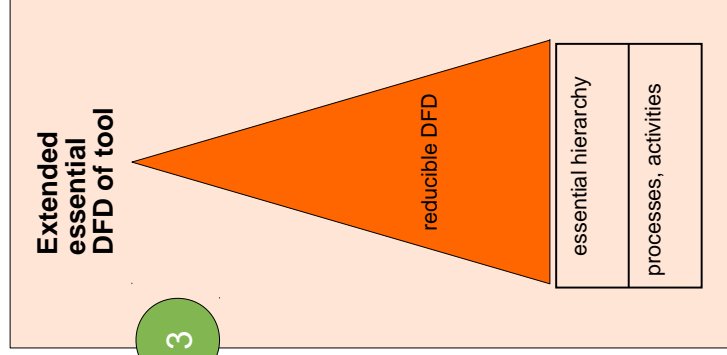
2



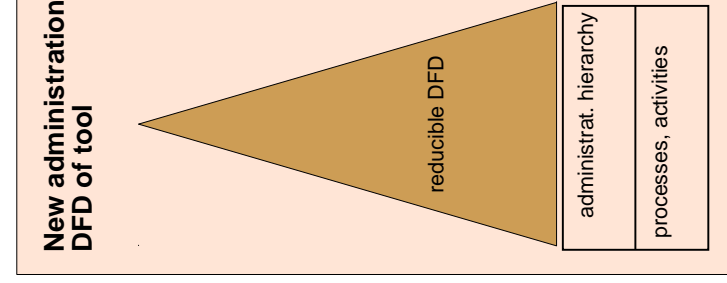
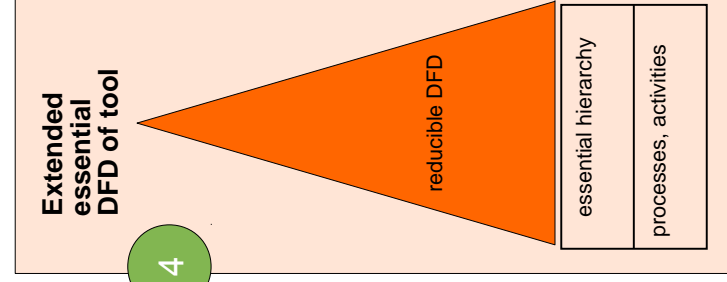
3) Extend Essence 4) Add Administration



3

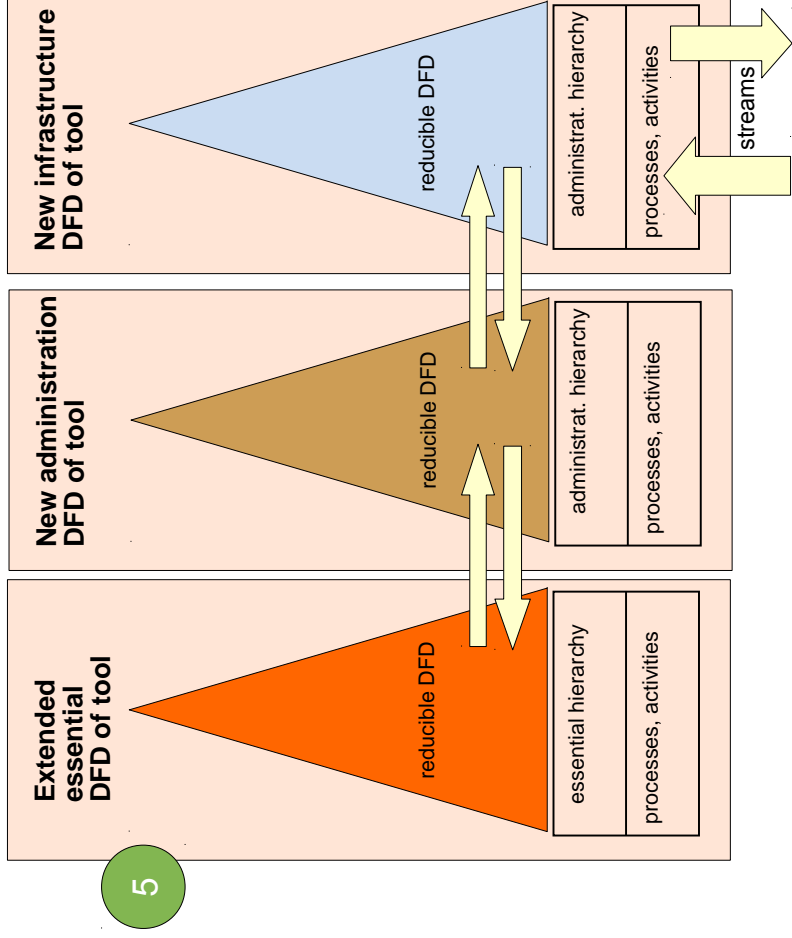


4



5) Add New Infrastructure

33



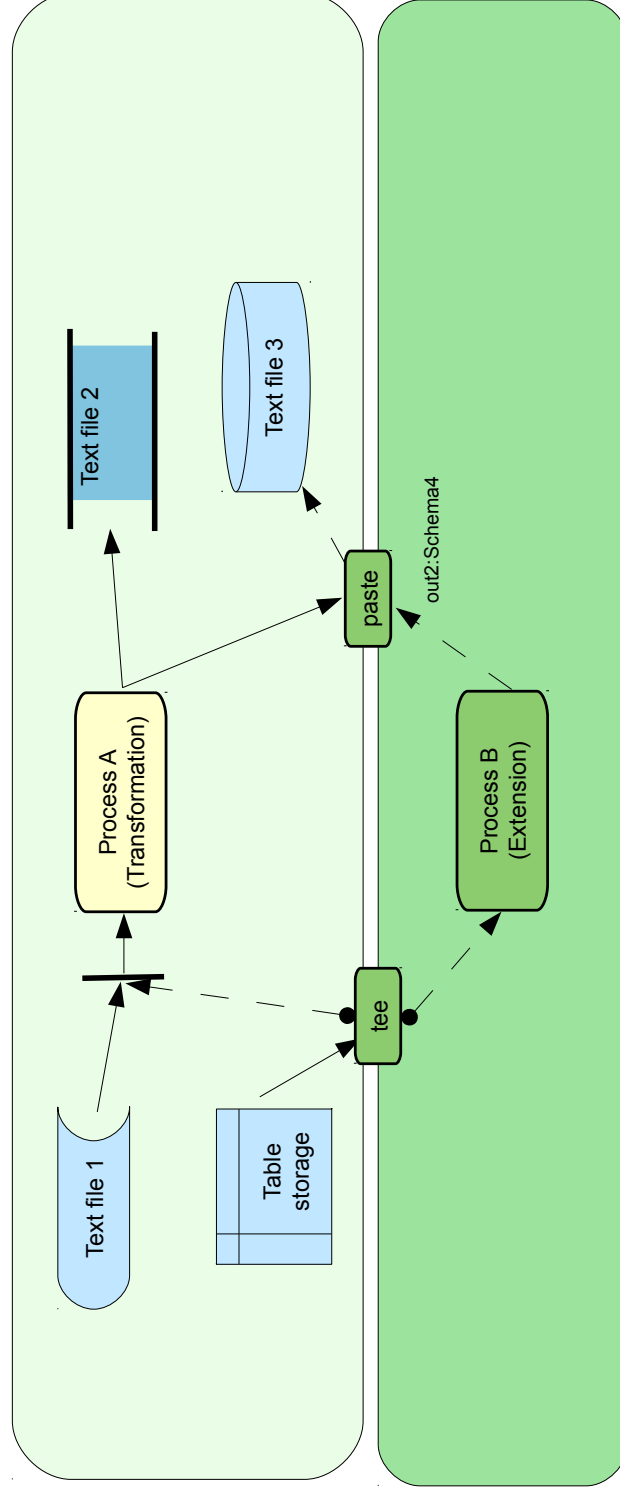
Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



Example: Shell Script Extension in Linux

34

- Streams are text streams (untyped)
- tee is a little filter replicating a text stream
- paste or `Iam` are little filters merging two streams



Prof. U. Asmann, Softwareentwicklungswerkzeuge (SEW)



The End – What did we learn?

35

- ▶ Stream-based tools can easily be extended and composed
 - with input stream replication
 - with asynchronous or synchronous output stream merge
 - with aspect-oriented extension
- ▶ Tools should be composed only with regard to their Essence, disregarding Administration and Infrastructure aspects