1 LM

1.1 Basic concepts

`Normal` and **`bold Type-Writer`** and *`italic`* and ***`bold italic`***

1.2 Some text

This chapter introduces the notion of component and the various associated concepts. This presentation is the result of the analysis of numerous existing component model. All these component models are archjava are more concerned by the task of programming components.

The insterested reader may consult the following recent state of the art or the following books Szyperski98, oussalah05 or icar. **Software components** were proposed as a solution to really increase the productivity by promoting *reuse to a large* scale. To make it possible to reuse components, they enforce a very low coupling with their environment by using architecture as the key artifact of the development life cycle. Components foster better reuse than objects by replacing inheritance and reference by composition and ports.

Component paradigm is built upon two main concepts:

- *architectural elements* that encapsulate treatments. They are sometimes called building blocks (BB). The functionalities realised by these treatments are called *services*.
- *architectures* that (statically) describe the set of architectural elements and their relations (sometimes called binding). We call *composition* the action of putting in relation architectural elements.

No precise definition of an architectural element is universally accepted[1] is very often cited both by its supporters and detractors. but there is a consensus in defining it as a *piece of software* that have the following properties: (1) it is reusable, (2) it may be subject to composition and (3) it may be configured. The exact consequences of these properties are discussed below and detailed in the following sections describing in more details the notion of architectural element.

```
let refine ae =
  ae.avatar <-
    (match ae.avatar with
      | Abstract -> Specification (specify ())
```

[1] C. Szyperski in Szyperski98

```
    | Specification aes -> Implementation (implem aes)
    | Implementation aei -> Package (pack aei)
    | Package aep -> Instance (inst aep)
    | Instance _ as aea -> print_endline "Nothing to do"; aea
  )
```

2 Baskerville, Skia and Courier

2.1 Basic concepts

`Normal and` **`bold Type-Writer`** `and` *`italic`* `and` ***`bold italic`***

2.2 Some text

This chapter introduces the notion of component and the various associated concepts. This presentation is the result of the analysis of numerous existing component model. All these component models are archjava are more concerned by the task of programming components.

The insterested reader may consult the following recent state of the art or the following books Szyperski98, oussalah05 or icar. Software components were proposed as a solution to really increase the productivity by promoting reuse to a large scale. To make it possible to reuse components, they enforce a very low coupling with their environment by using architecture as the key artifact of the development life cycle. Components foster better reuse than objects by replacing inheritance and reference by composition and ports.

Component paradigm is built upon two main concepts:

- *architectural elements* that encapsulate treatments. They are sometimes called building blocks (BB). The functionalities realised by these treatments are called *services*.
- *architectures* that (statically) describe the set of architectural elements and their relations (sometimes called binding). We call *composition* the action of putting in relation architectural elements.

No precise definition of an architectural element is universally accepted[2] is very often cited both by its supporters and detractors. but there is a consensus in defining it as a *piece of software* that have the following properties: (1) it is reusable, (2) it may be subject to composition and (3) it may be configured. The exact consequences of these properties are discussed below and detailed

[2] C. Szyperski in Szyperski98

in the following sections describing in more details the notion of architectural element.

```
let refine ae =
  ae.avatar <-
    (match ae.avatar with
      | Abstract -> Specification (specify ())
      | Specification aes -> Implementation (implem aes)
      | Implementation aei -> Package (pack aei)
      | Package aep -> Instance (inst aep)
    | Instance _ as aea -> print_endline "Nothing to do"; aea
    )
```

3 Didot, Lucida Grande and Monaco

3.1 Basic concepts

`Normal and bold Type-Writer and italic and bold italic`

3.2 Some text

This chapter introduces the notion of component and the various associated concepts. This presentation is the result of the analysis of numerous existing component model. All these component models are archjava are more concerned by the task of programming components.

The insterested reader may consult the following recent state of the art or the following books Szyperski98, oussalah05 or icar. **Software components** were proposed as a solution to really increase the productivity by promoting reuse to a large scale. To make it possible to reuse components, they enforce a very low coupling with their environment by using architecture as the key artifact of the development life cycle. Components foster better reuse than objects by replacing inheritance and reference by composition and ports.

Component paradigm is built upon two main concepts:

- *architectural elements* that encapsulate treatments. They are sometimes called building blocks (BB). The functionalities realised by these treatments are called *services*.
- *architectures* that (statically) describe the set of architectural elements and their relations (sometimes called binding). We call *composition* the action of putting in relation architectural elements.

No precise definition of an architectural element is universally accepted[3] is very often cited both by its supporters and detractors. but there is a consensus in defining it as a *piece of software* that have the following properties: (1) it is reusable, (2) it may be subject to composition and (3) it may be configured. The exact consequences of these properties are discussed below and detailed in the following sections describing in more details the notion of architectural element.

```
let refine ae =
  ae.avatar <-
    (match ae.avatar with
      | Abstract -> Specification (specify ())
      | Specification aes -> Implementation (implem aes)
      | Implementation aei -> Package (pack aei)
      | Package aep -> Instance (inst aep)
      | Instance _ as aea -> print_endline "Nothing to do"; aea
    )
```

4 Hoefler Text, Gill Sans and Andale Mono

4.1 Basic concepts

```
Normal and bold Type-Writer and italic and bold italic
```

4.2 Some text

This chapter introduces the notion of component and the various associated concepts. This presentation is the result of the analysis of numerous existing component model. All these component models are archjava are more concerned by the task of programming components.

The insterested reader may consult the following recent state of the art or the following books Szyperski98, oussalah05 or icar. **Software components** were proposed as a solution to really increase the productivity by promoting *reuse to a large* scale. To make it possible to reuse components, they enforce a very low coupling with their environment by using architecture as the key artifact of the development life cycle. Components foster better reuse than objects by replacing inheritance and reference by composition and ports.

Component paradigm is built upon two main concepts:

[3] C. Szyperski in Szyperski98

- *architectural elements* that encapsulate treatments. They are sometimes called building blocks (BB). The functionalities realised by these treatments are called *services*.

- *architectures* that (statically) describe the set of architectural elements and their relations (sometimes called binding). We call *composition* the action of putting in relation architectural elements.

No precise definition of an architectural element is universally accepted[4] is very often cited both by its supporters and detractors. but there is a consensus in defining it as a *piece of software* that have the following properties: (1) it is reusable, (2) it may be subject to composition and (3) it may be configured. The exact consequences of these properties are discussed below and detailed in the following sections describing in more details the notion of architectural element.

```
let refine ae =
  ae.avatar <-
    (match ae.avatar with
      | Abstract -> Specification (specify ())
      | Specification aes -> Implementation (implem aes)
      | Implementation aei -> Package (pack aei)
      | Package aep -> Instance (inst aep)
      | Instance _ as aea -> print_endline "Nothing to do"; aea
    )
```

5 Optima, Chalkboard and American Typewriter

5.1 Basic concepts

Normal and **bold Type-Writer** and italic and ***bold italic***

5.2 Some text

This chapter introduces the notion of component and the various associated concepts. This presentation is the result of the analysis of numerous existing component model. All these component models are archjava are more concerned by the task of programming components.

The insterested reader may consult the following recent state of the art or the following books Szyperski98, oussalah05 or icar. **Software components** were

[4] C. Szyperski in Szyperski98

proposed as a solution to really increase the productivity by promoting reuse to a large scale. To make it possible to reuse components, they enforce a very low coupling with their environment by using architecture as the key artifact of the development life cycle. Components foster better reuse than objects by replacing inheritance and reference by composition and ports.

Component paradigm is built upon two main concepts:

- *architectural elements* that encapsulate treatments. They are sometimes called building blocks (BB). The functionalities realised by these treatments are called *services*.

- *architectures* that (statically) describe the set of architectural elements and their relations (sometimes called binding). We call *composition* the action of putting in relation architectural elements.

No precise definition of an architectural element is universally accepted[5] is very often cited both by its supporters and detractors. but there is a consensus in defining it as a *piece of software* that have the following properties: (1) it is reusable, (2) it may be subject to composition and (3) it may be configured. The exact consequences of these properties are discussed below and detailed in the following sections describing in more details the notion of architectural element.

```
let refine ae =
 ae.avatar <-
  (match ae.avatar with
   | Abstract -> Specification (specify ())
   | Specification aes -> Implementation (implem aes)
   | Implementation aei -> Package (pack aei)
   | Package aep -> Instance (inst aep)
   | Instance _ as aea -> print_endline "Nothing to do"; aea
  )
```

[5] C. Szyperski in Szyperski98