CMDI 1.2
Component Lifecycle Management
Workgroup Report

Oddrun Ohren, Axel Herold, Twan Goosen

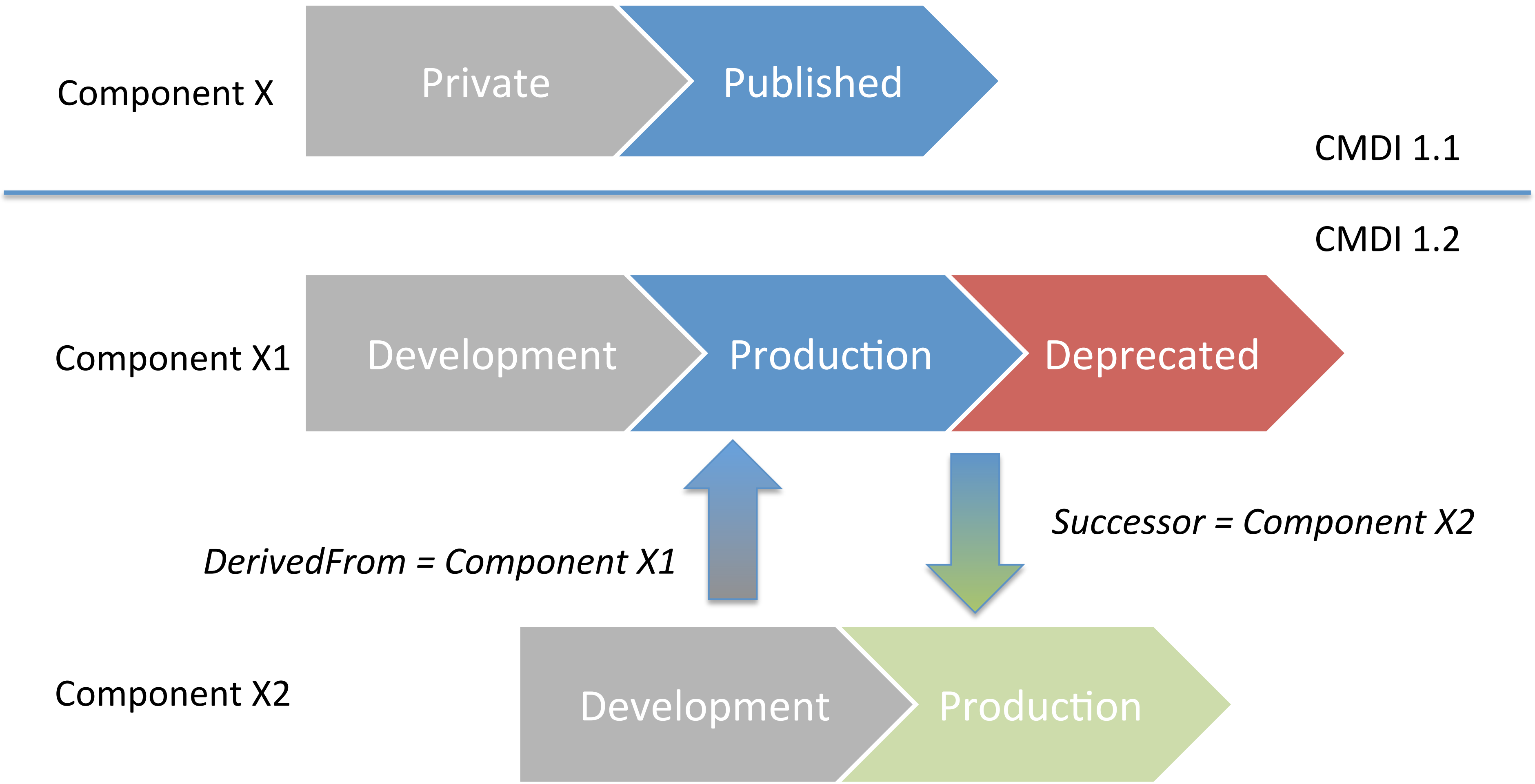
CMDI Taskforce Meeting
Utrecht, 21 February 2014

Topics

- **Versioning** of components and profiles
- **Deprecation** of components and profiles
- **Successors** and **derivatives** of components and profiles

- **Model** and **workflow** aspects of the above

Versioning and deprecation



Component X

Private

Published

CMDI 1.1

CMDI 1.2

Component X1

Development

Production

Deprecated

DerivedFrom = Component X1

Successor = Component X2

Component X2

Development

Production

Lifecycle status

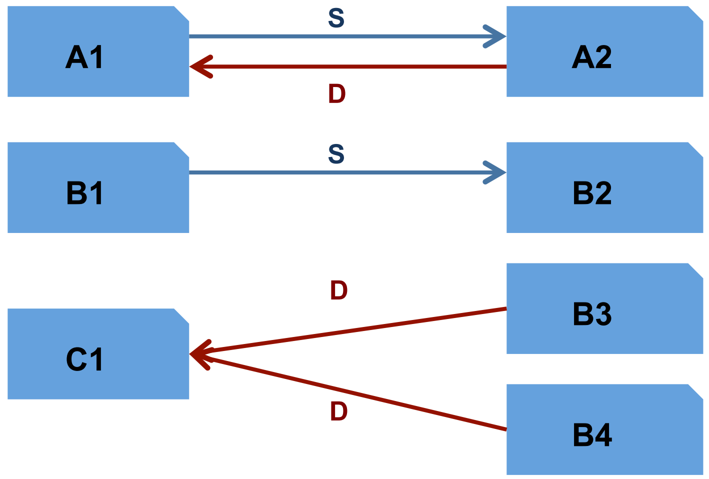
- **Development**
  - Unstable (subject to change)
  - Usage for production discouraged
- **Production**
  - Stable
  - Usage encouraged
- **Deprecated**
  - Stable
  - Usage discouraged
  - Successor may exist

Succession

- Successor relation is defined by the predecessor
  - Only one successor per deprecated component/profile
- Only deprecated components/profiles can have a successor
- Final successor should have production status
  - (Succession is a transitive relation)
  - Infrastructure should enforce this
- Successor of a component can be any other component
  - Successor relation does not imply compatibility

Derivation

- Purpose: keep a record of 'genetically' related components
- Derivation relation is <u>defined by the derivative</u>



- A derivative need not be a successor (rather a fork)
- A successor need not be a derivative!
- A component can have any number of derivatives

Component specification

- Lifecycle properties go into the header of component spec

Status: [development production deprecated]

Comment (optional)

Successor

DerivedFrom

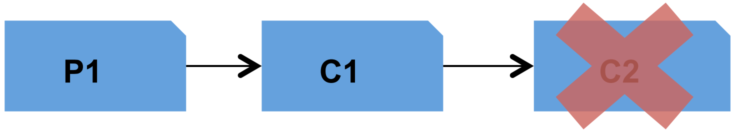
```
<ComponentSpec isProfile="true">
  <Header>
    <ID>clarin.eu:cr1:p_1289827960126</ID>
    <Name>LrtInventoryResource</Name>
    <Description>Resources as stored before in the CLARIN LRT
inventory</Description>
    <Status>deprecated</Status>
    <StatusComment>The following fields were missing: actor age,
content
language
</StatusComment>
    <Successor>http://catalog.clarin.eu/ds/ComponentRegistry/rest/
registry/profiles/clarin.eu:cr1:p_3989827960127</Successor>
    <DerivedFrom>http://catalog.clarin.eu/ds/ComponentRegistry/
rest/registry/profiles/clarin.eu:cr1:p_1227960126456</DerivedFrom>
  </Header>
```

Infrastructure: Lifecycle management

- Component Registry should enforce constraints on lifecycle status change:
  - Only the owner (and administrators of the Component Registry) can change the lifecycle status of a component
  - Only the following status transitions should be possible:
    - development ➔ production
    - production ➔ deprecated

Infrastructure: Lifecycle management

- Status is local, does not propagate



- ‘Edit as new’ fills in *DerivedFrom* by default
  - There should be an option for manual specification

Infrastructure: guidance

- Filter publicly visible components/profiles in view:
  - Development profiles ON/OFF
  - Deprecated profiles ON/OFF

- Warn users
  - By visually highlighting deprecated components in lists and referencing components
  - By asking for confirmation in case of reuse

- Notify users by e-mail or RSS when a component…
  - …referenced by their own components changes status
  - …they are manually subscribed to changes status
  - …is created that is derived from one of their components

Impact on other tools

- Tools do not *need* to be aware of lifecycle status
  - Status does not affect specification itself
  - A successor is just another component with a unique ID

- Tools that should support component lifecycle
  - Editors (e.g. Arbil)
    - Hide deprecated profiles
    - Warn users of deprecated profiles
  - SMC browser
    - Index lifecycle status for curation purposes
    - Visualise related profiles on basis of *derivedFrom* relations

Discussion

- Centre impact not considered yet?
- Other tools to consider?
- Other points?