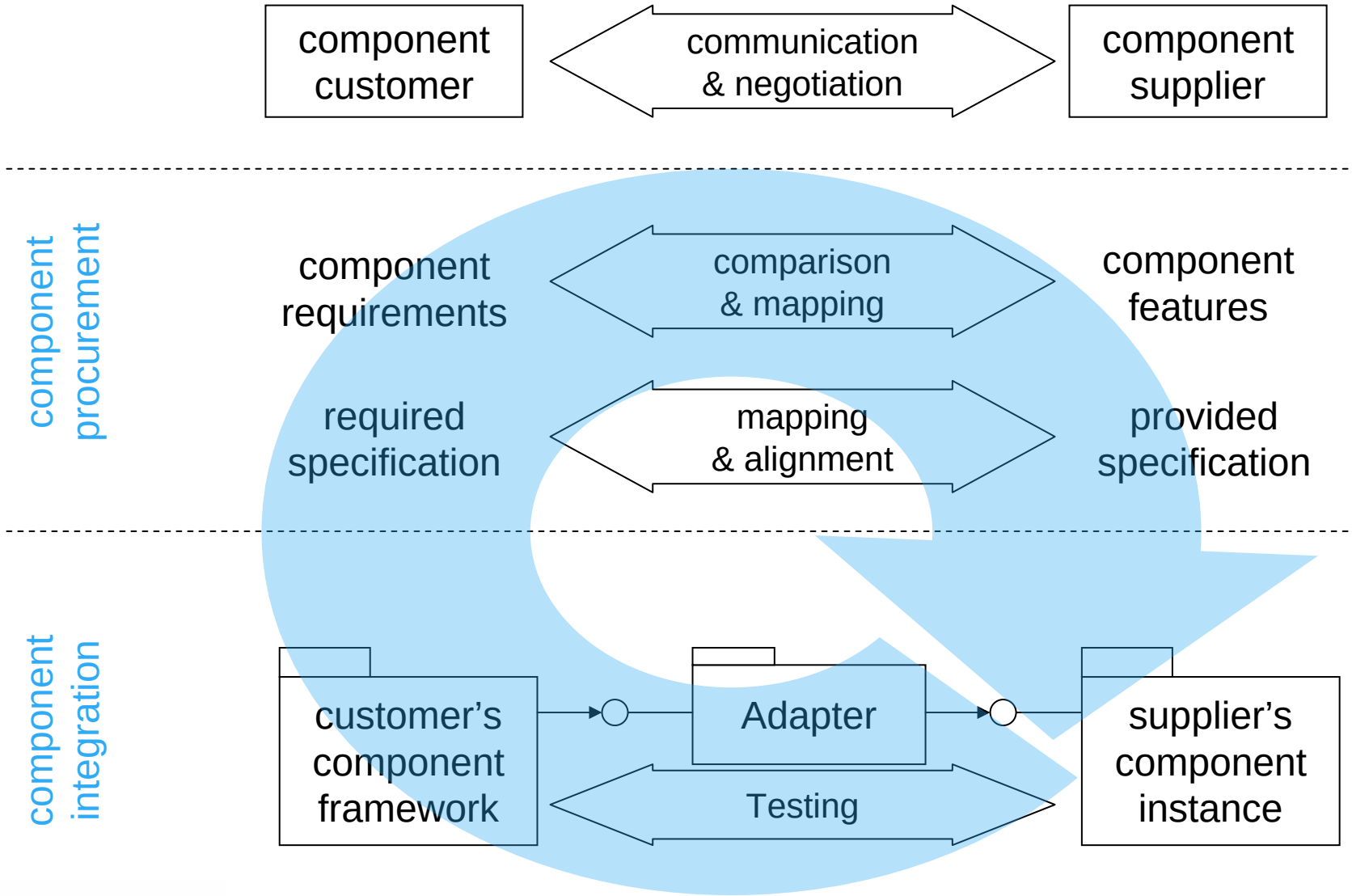
Bridging the Gap between Non-formal and Formal Software Component Requirements Specifications for ES Engineering

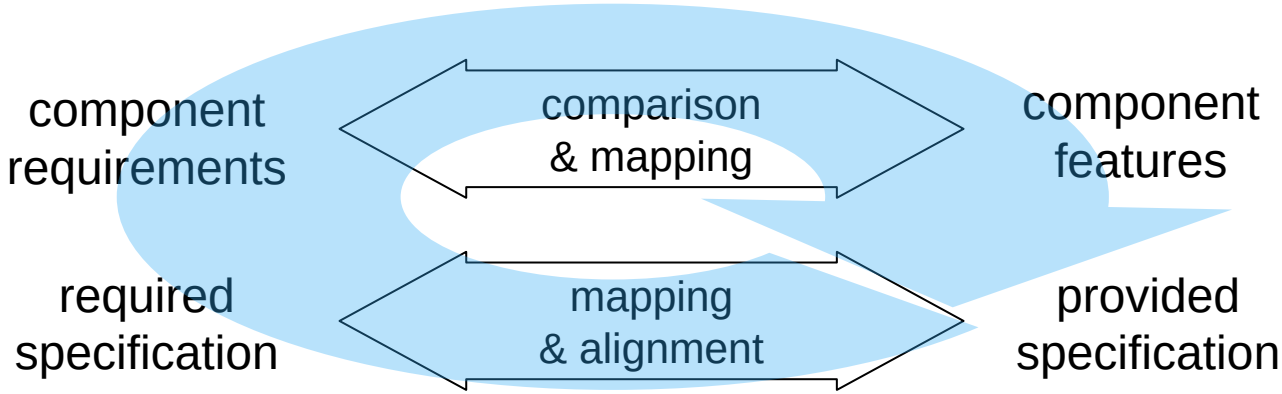
Hans-Gerhard Gross and Arjan van Gemund
Embedded Software Lab
Delft University of Technology
Mekelweg 4
2628 CD Delft

TU Delft



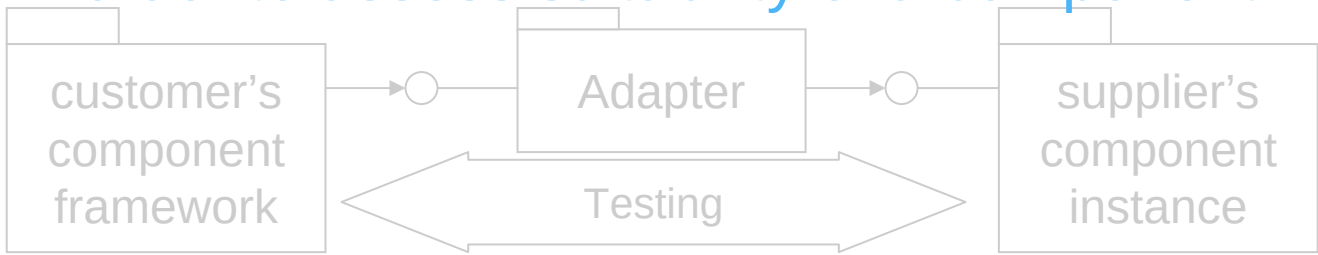
Complete development cycle necessary
In order to assess suitability of a component

component procurement



component integration

Ideally: only modeling cycle necessary
In order to assess suitability of a component



component customer

communication & negotiation

component supplier

component procurement

component **requirements**

comparison & mapping

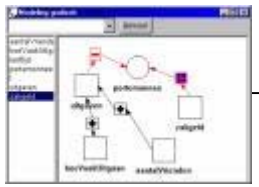
component features

formalization of requirements

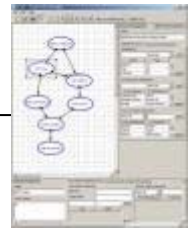
required **specification**

simulation & testing

provided specification



adapt



customer's component framework

Code Generation

Adapter

Code Generation

customer's component framework

Formalized Requirements

- Application of use case templates for early requirements mappings
  - Readily used in industry
  - Structured
  - Can be processed automatically (XML format)
    - Required is functionality x,y,z
    - Perform a search on the component repository: which one provides x,y,z?

Example: Formalization of Requirements Current Situation

If audio comes in from the tuner, it is routed as SIF signal to the analog audio decoder.

Audio signal path

(AnaAdec). After that it is baseband audio. It can be presented on all audio outputs analog or digital.

In case the audio is from HDMI it can either be the SPDIF content belonging to the HDMI signal, or it can be an analog fallback in case the HDMI signal is video only (DVI). The analog fallback can come from an audio-only AV-cluster. In the latter case the client needs observers to see that there is no SPDIF signal or that the signal can not be decoded. These observers are available from SpdifIn and DigAdec components.

Audio back-end

The audio back-end is similar for all use cases. The routing and selection of audio signal is specified by the parameters. For the speakers output there is a range of audio features possible, like sound styles (e.g. hall, movie) and improvements (e.g. dynamic base-boost, loudness, virtualizing etc.). For all other audio outputs there is a much more basic set of featuring, specified by the logical component instances in Section 4.46.

Video signal path

Video can come from the tuner input, in which the signal will go as an I/F signal to the analog video decoder. The signal can also come from an AvIn cluster and go as CVBS, Yc, YPbPr to the analog video decoder. After decoding, the video signal is a YUV signal, it will go to the VBI extractor and video back-end. Hdmi signals go directly to the VBI extractor and video back-end.

Example: Formalization of Requirements

Final desired form

Name	Switch_OPERATIONAL
Goal in Context	Use case for switching the TV on from standby
Actors	User
Trigger	Actor presses STANDBY or a DIGIT or UP or DOWN or a SOURCE on the remote control
Preconditions	TV is in standby
Post conditions on success	TV becomes operational and shows the selected program or source
Post conditions on failure	TV remains in standby TV does not show selected program or source
Description of basic course	Actor presses a key on the remote control
Description alternative courses	<undefined>
Exceptions	Program or source not available
NF-Requirements	<tbd>
Concurrent uses	<none>
Refines	<none>
Is refined by	<none>
Revisions	Draft

Intermediate form

Name	Switch_OPERATIONAL
Goal in Context	Use case for switching the TV 810 on from standby
Actors	User
Trigger	RemoteControl.UP( ) <xor> RemoteControl.DOWN( ) <xor> RemoteControl.DIGIT ( ) <xor> RemoteControl.PREVIOUS ( ) <xor> RemoteControl.SOURCE(source)
Preconditions	TVstate.STANDBY == true
Post conditions on success	TVstate.OPERATIONAL == true TVsubstate.SHOW (Program | Source) == true
Post conditions on failure	TVstate.STANDBY == true TVsubstate.SHOW (Program | Source) == false
Description of basic course	RemoteControl.KeyPressed (UP|DOWN|DIGIT|PREVIOUS)
Description alternative courses	<undefined>
Exceptions	TVInput (Program) == false TVInput (Source) == false
NF-Requirements	<tbd>
Concurrent uses	<none>
Refines	<none>
Is refined by	<none>
Revisions	Draft

TU Delft

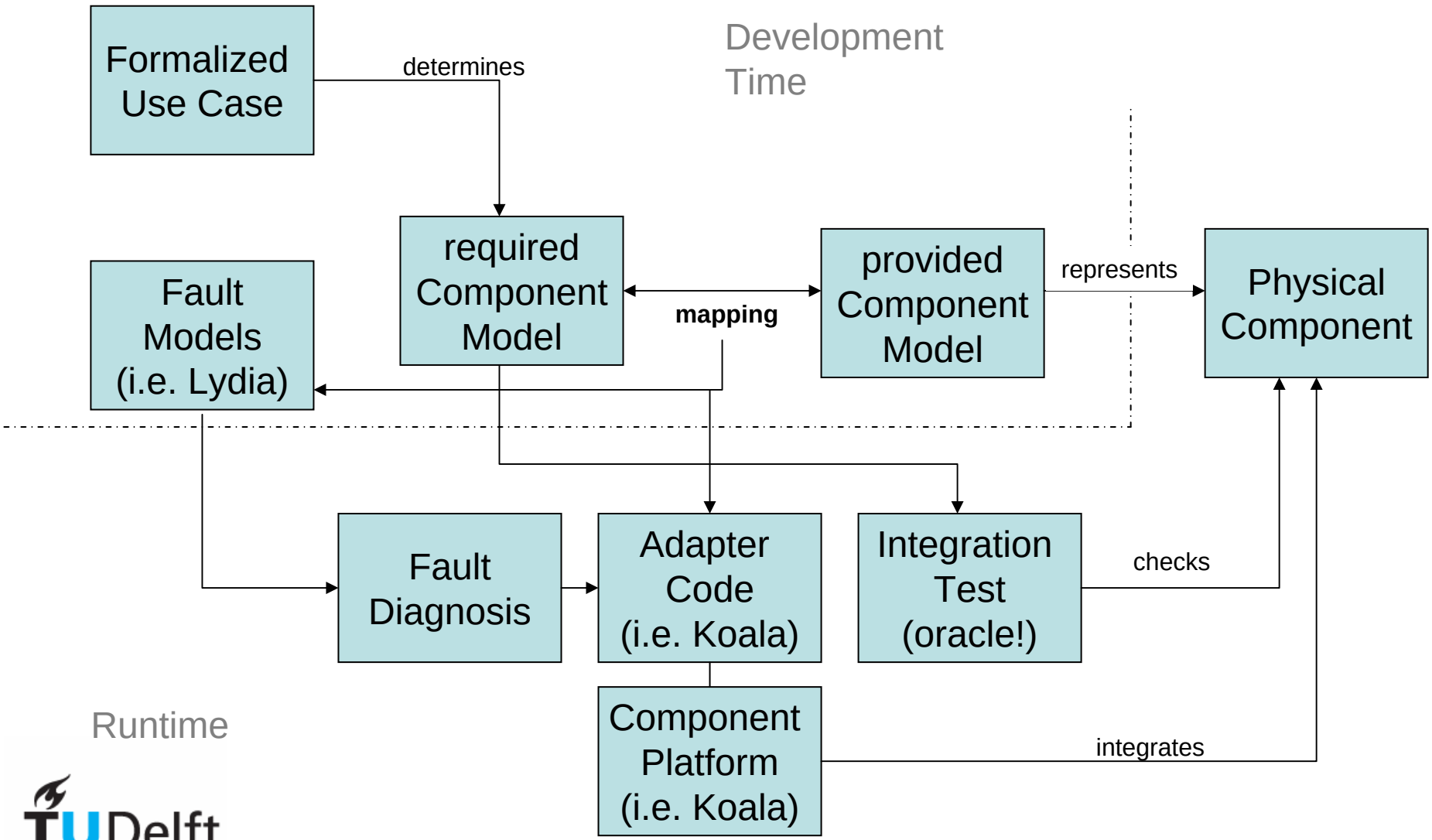
More formalized system design for component feature mapping

- Notation: UML → SDL
  - Industry is moving towards UML
  - UML has no formal semantics
  - SDL is fully formal and comes with powerful tools
    - Limited concepts for timing specifications
  - SDL/UML profile provides conceptual mapping
  - Transform from UML to SDL

Advantage of formalization: Code Generation

- Adapter Code
  - E.g. explicit specification of reentrant component operations
  - Uncover reentrancy mismatches
  - Generate serialization code in the adapter to resolve mismatches
- Integration test cases
  - Out of the formalized specifications
  - Partial generation out of UML
  - Full generation with SDL (oracle)
- Fault-diagnosis (run-time)
  - Trader project: injection of fault analysis mechanisms

Overview Artifacts



Summary & Evaluation

- Formalized component requirements
  - Eventually specification of full behavior and NF requirements (response time)
  - Start with very simple but frequently occurring pattern (e.g. reentrancy pattern)
  - We do not expect to close the semantic gap between requirements and component specs completely
- Introduction of a new modeling notation
  - Requires strong motivation
- Both will only be accepted if we can demonstrate considerable savings in effort → tools are essential