automotive HMI

Gefördert durch:

Bundesministerium für Wirtschaft und Technologie

aufgrund eines Beschlusses des Deutschen Bundestages

Model-Based Testing (WP7)

Dr. Christoph Steglich, Daimler AG

status meeting 13.12.2012

automotive HMI

Model-Based Testing
Motivation

Status quo
- Increasing complexity („More and more features & services“)
- Rising variability („More and more specific adaptions“)
- Rapid development-cycles („More and more consumer products“)

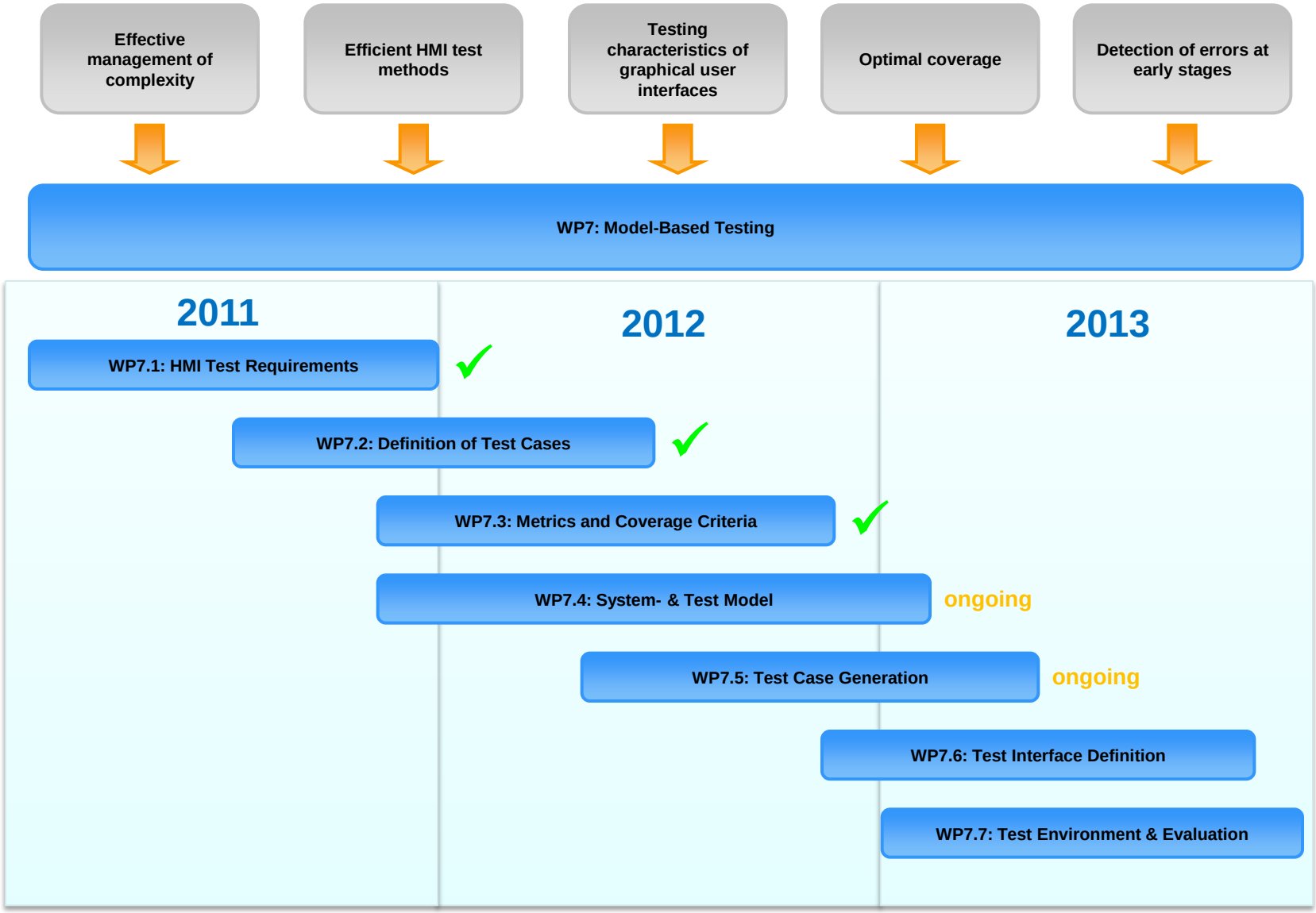
Objective
- Ensure high product quality across all possible variations

Challenges
- Quick and precise determination what consequences changes have
- Systematic detection of mutual dependencies
- Understandable and clear presentation

automotive HMI

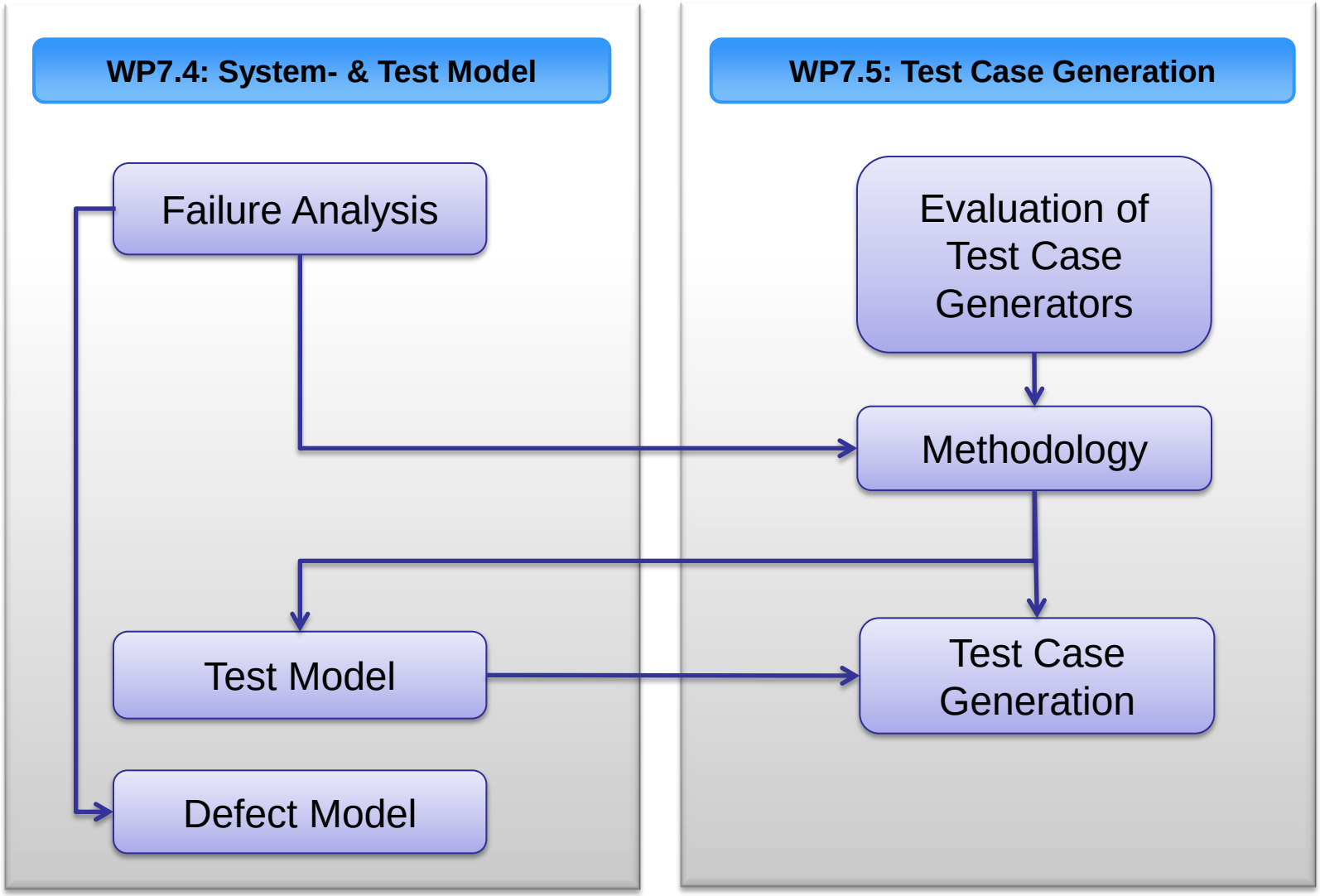
Model-Based Testing

Objectives & Schedule



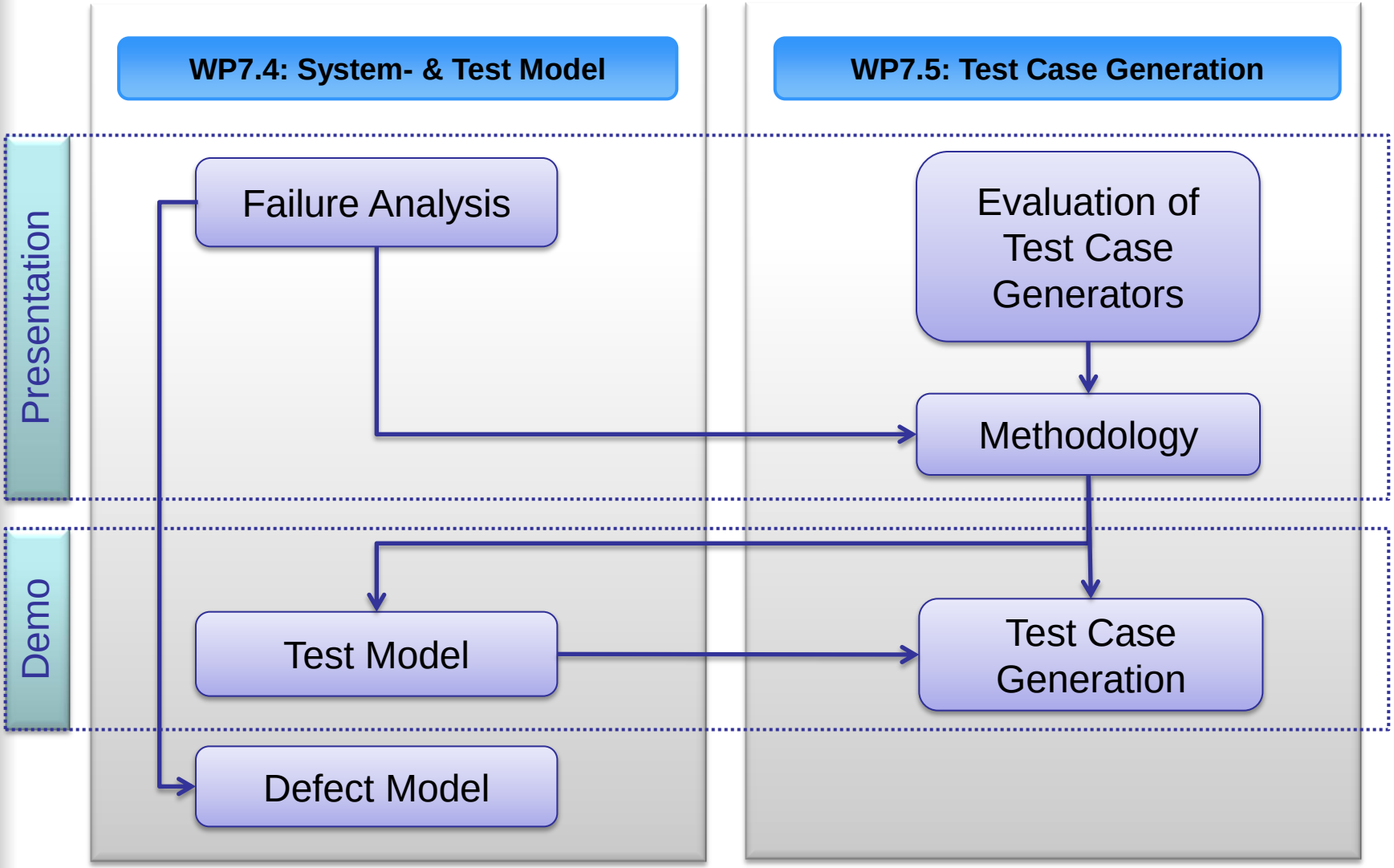
automotive HMI

Focus: Current Work



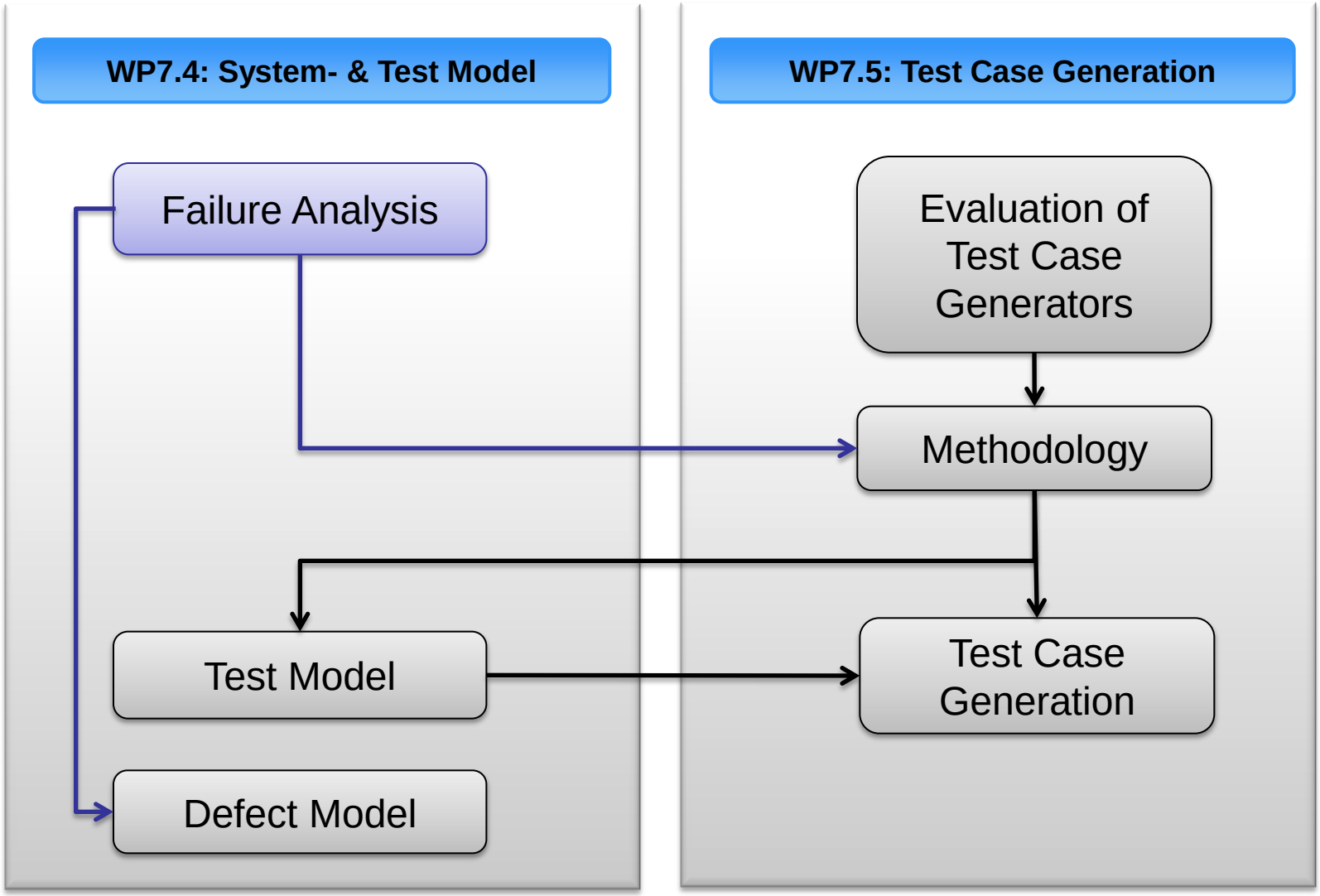
automotive HMI

Focus: Current Work



automotive HMI

Focus: Current Work



Failure Analysis

Goal: Identification and Classification of typical GUI failures

- Take the failure pattern into account when developing the test strategy
- Provides input for the methodology
  - Helps to determine which data has to become part of the test model.
  - Influences the way the test model has to be build so that the test cases needed will be available through generation.
- Results will also be used for evaluation purposes
  - Create a system model that exhibits a realistic failure pattern
  - Check if all failures (quantity & category) are found by the generated test cases.

Approach

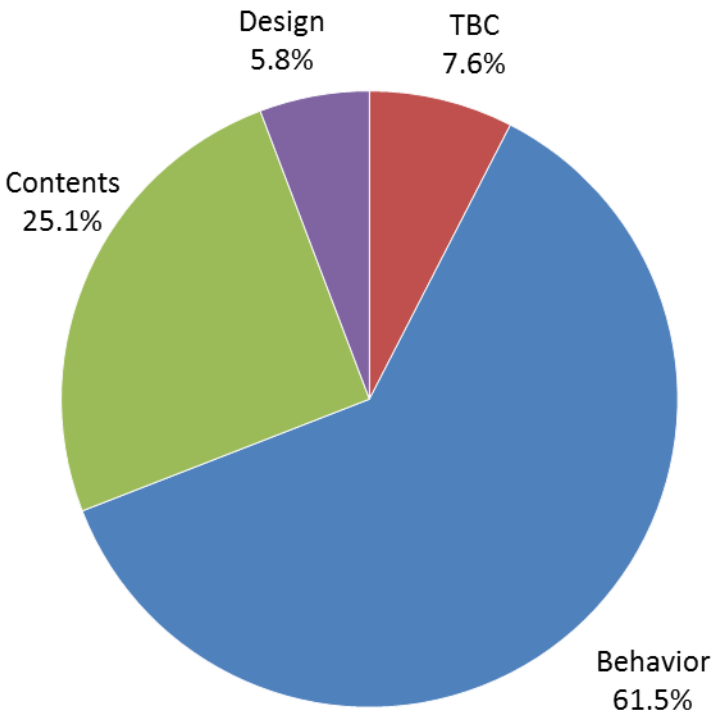
Analyzed data basis

- More than 3000 failure reports (Audi, Bosch and Daimler)
- Representing a broad variety of contexts, such as
  - System Under Test
  - Test strategies
  - Test personnel
  - Test environments
- Clustering failure reports disjunctively
  - ⅓ of data used to develop the basics of the taxonomy
  - ⅔ of data used to fine-tune sub-categories

Classification requirements

1. Hierarchical structure
2. Min 2 and max 5 categories per level
3. Max 10% total percentage on lowest level
4. Class “To Be Categorized” (TBC) which is limited to 10%

Results: Classification & Distribution



Top level follows the Model-View-Controller design pattern

Model [here *contents*] (25.1%)

Holding the actual data
(e.g. text, icons, etc.)

View [here *design*] (5.8%)

Describing how the contents have to be displayed

(e.g. position, color, font, etc.)

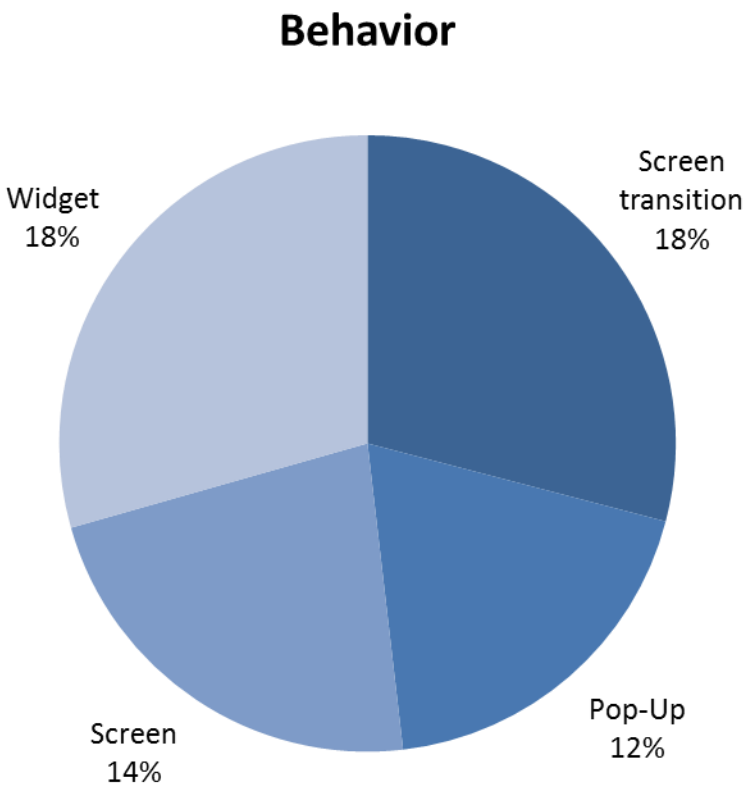
Controller [here *behavior*] (61.5%)

Any kind of logic in response to incoming events
(e.g. menu option iteration, option activation, etc.)

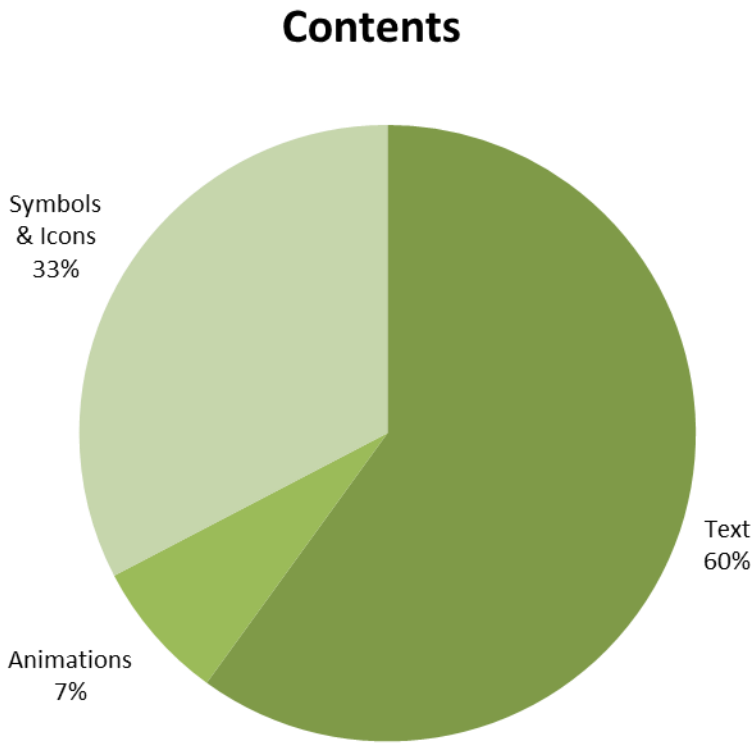
automotive HMI

Failure Analysis

Classification & Distribution



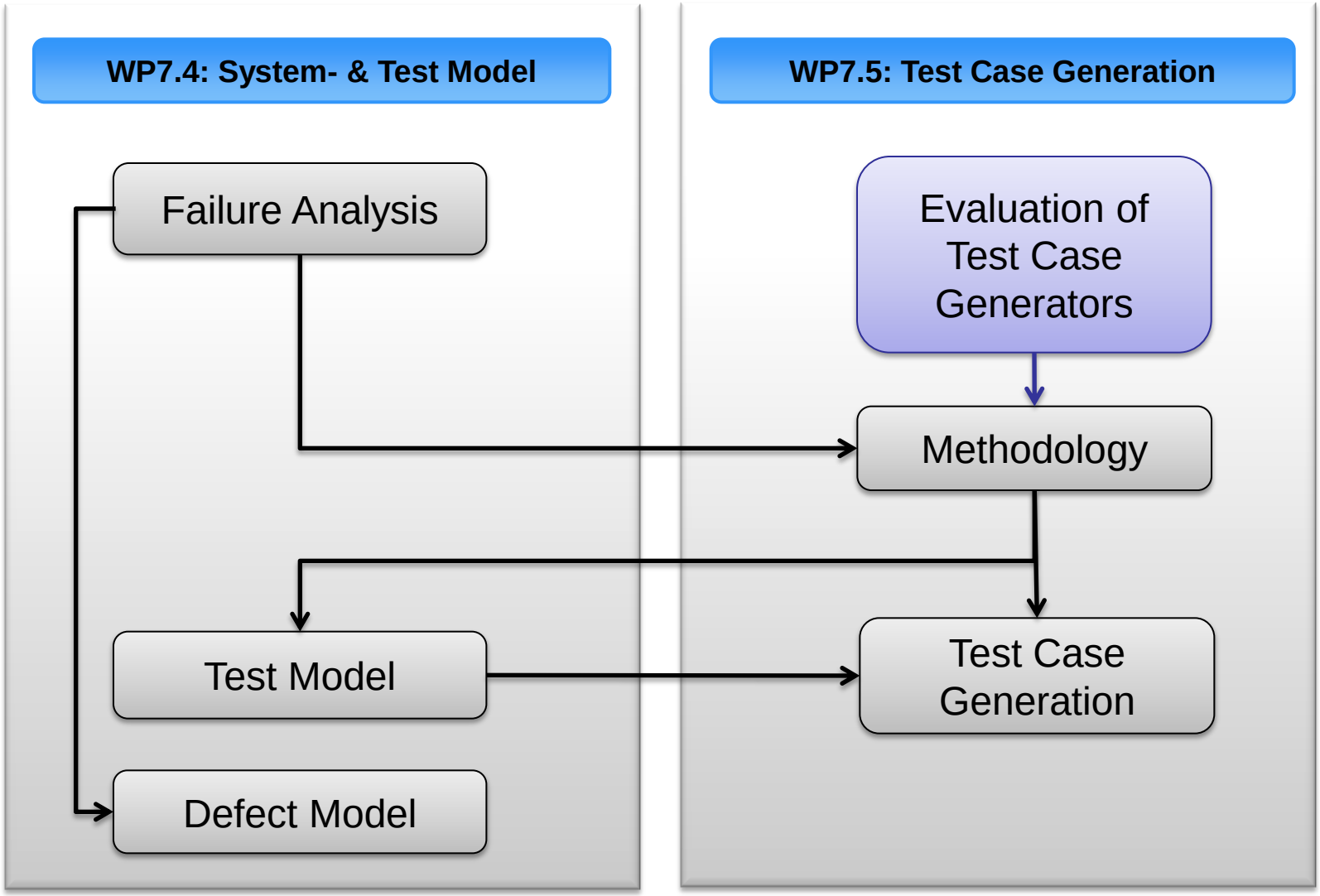
Percentage (total) 61.5%



Percentage (total) 25.1%

automotive HMI

Focus: Current Work



automotive HMI

Evaluation of Test Case Generators

Goal: Evaluate applicability of proprietary Test Case Generators

- A set of 9 Test Case Generators had been evaluated
- The assessment process had 2 steps
  - First check of all products to choose the 2-3 most appropriate
  - Detailed evaluation of the nominated products
- Evaluation based on a set of prioritized criteria
  - Required Operating System
  - Input Formats
  - Test Case Generation
  - Execution Support
  - Required Tool Chain
  - …

automotive HMI

Evaluation Results (1/2)

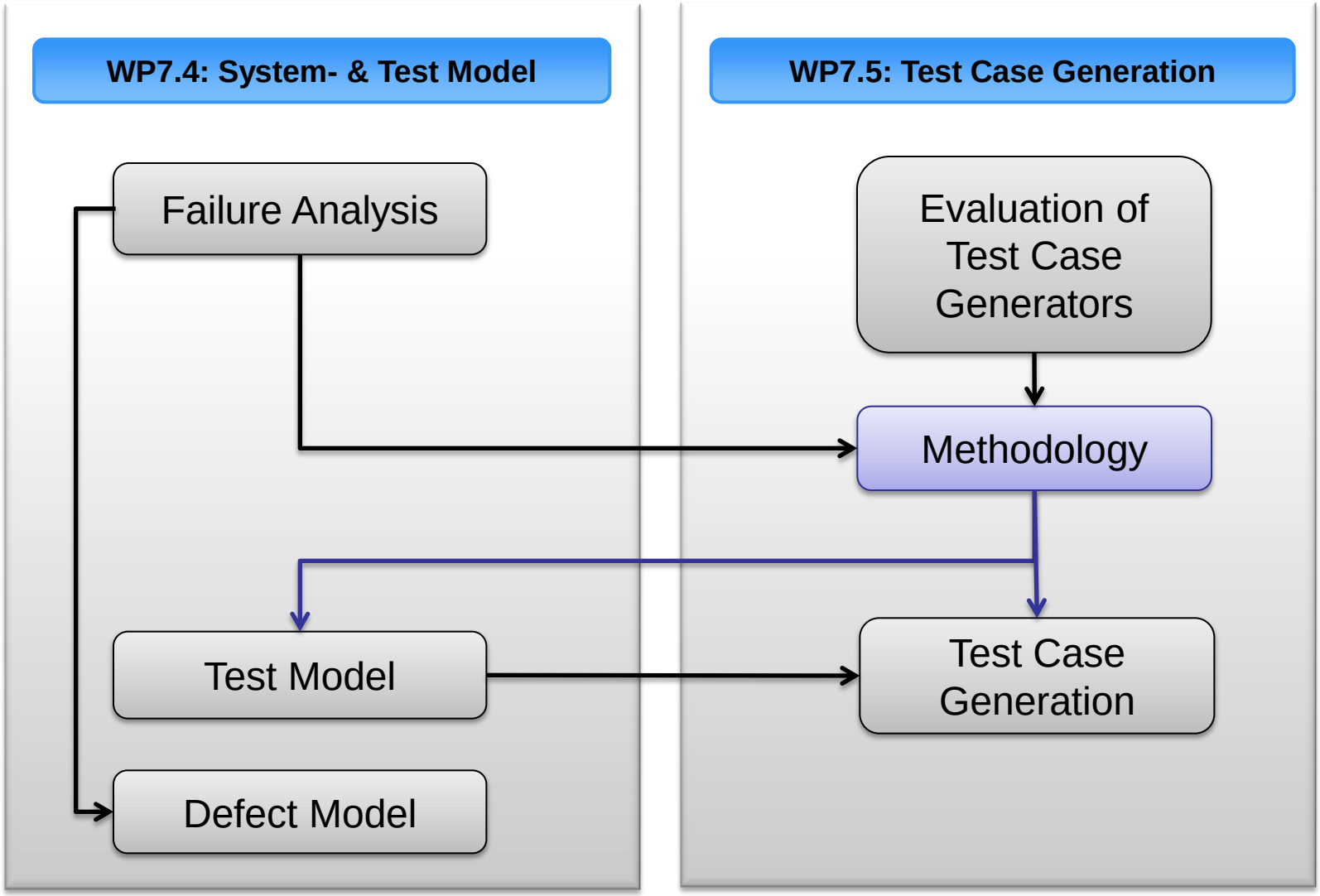
	CQ Designer	Test Designer	MBTsuite	RT Tester
OS	Linux und Windows (x86 und x64) Distributed generation	Linux und Windows	x86 Windows application (x64 Migration announced) No multi core support	Linux und Windows Multi core support Distributed generation
Input	Limited State Charts and proprietary scripting language (QML)	UML (Behavior models, Class- and Object diagrams, OCL)	Hierarchical models incl. Guards Extensions of functionality using Python snippets	Hierarchical models Extension of functionality using first order logic
Generation	Element coverage Usage probability	No adaption of coverage criteria Partial generation	Full path coverage (depth-first search)	Code and branch coverage
Execution	Offline and Online Tests Graphical back tracing of errors	Offline Tests Graphical back tracing of errors	Offline Tests Graphical back tracing of errors	Connection to Hardware-In-A-Loop execution environment
Tool Chain	Enterprise Architect	IBM Rational or Borland Together	Enterprise Architect	Enterprise Architect
Conclusion	⊕ **Infrastructure** ⊕ **Code integration** ⊕ **Use of Tools established tools** ⊖ **Price**	⊕ **Code Integration** ⊖ **Price** ⊖ **Limitation of tool chain**	⊕ **Price & support** ⊕ **Code integration** ⊕ **Use of established tools** ⊖ **Generation cost**	⊕ **Infrastructure** ⊖ **Hardware focus** ⊖ **Uncommon Code** ⊖ **No product yet**

Evaluation Results (2/2)

	Rhapsody ATG	tedeso (MMBT)	MaTeLo
OS	Linux and Windows	Windows XP or newer	Windows, Linux, Windows server
Input	Rhapsody UML System model	UML (class, package, use case, activity, sequence charts)	Usage model (Based on Markov chains)
Generation	Model Element Coverage & Model Code Coverage Incl. cancelation timer	Path coverage, activity and transition coverage, Round Trip, happy path	Usage probability incl. Transition coverage
Execution	Offline Tests Graphical back tracing of errors	No test case execution (testbench optional)	No test case execution Export and Analysis of results
Tool Chain	IBM Rhapsody incl. Rhapsody ATG Plugin	tedeso (+ Testbench)	IBM Doors → MaTeLo Requirements Library → MaTeLo Modell
Conclusion	(+) **Use of system models** (−) **Input limited** (−) **Problem error finding: Test model / system model**	(+) **Modular** (+) **Fast generation** (+) **Open API, adaptability** (i) **Use Case oriented** (−) **No state charts**	(+) **Connection to the specification** (i) **Usage model** (−) **No plugins for current version**

automotive HMI

Focus: Current Work



automotive HMI

Methodology: Different for Audi & Daimler

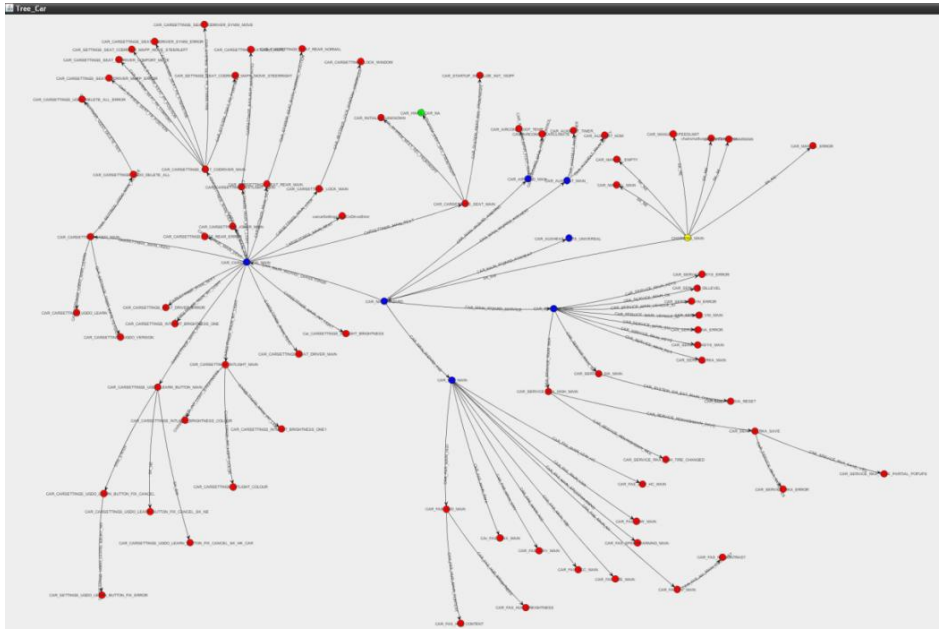
Approach AUDI	Approach Daimler
HMI Model already exists	Development of a dedicated HMI Test-Model
Model is used for system software generation	Model is used for test case generation exclusively
Model is developed and maintained by system development team	Model is developed and maintained by test engineers

Test execution based on HMI system models including dynamic system behavior

Test case generation based on a dedicated HMI test model

Approach
AUDI

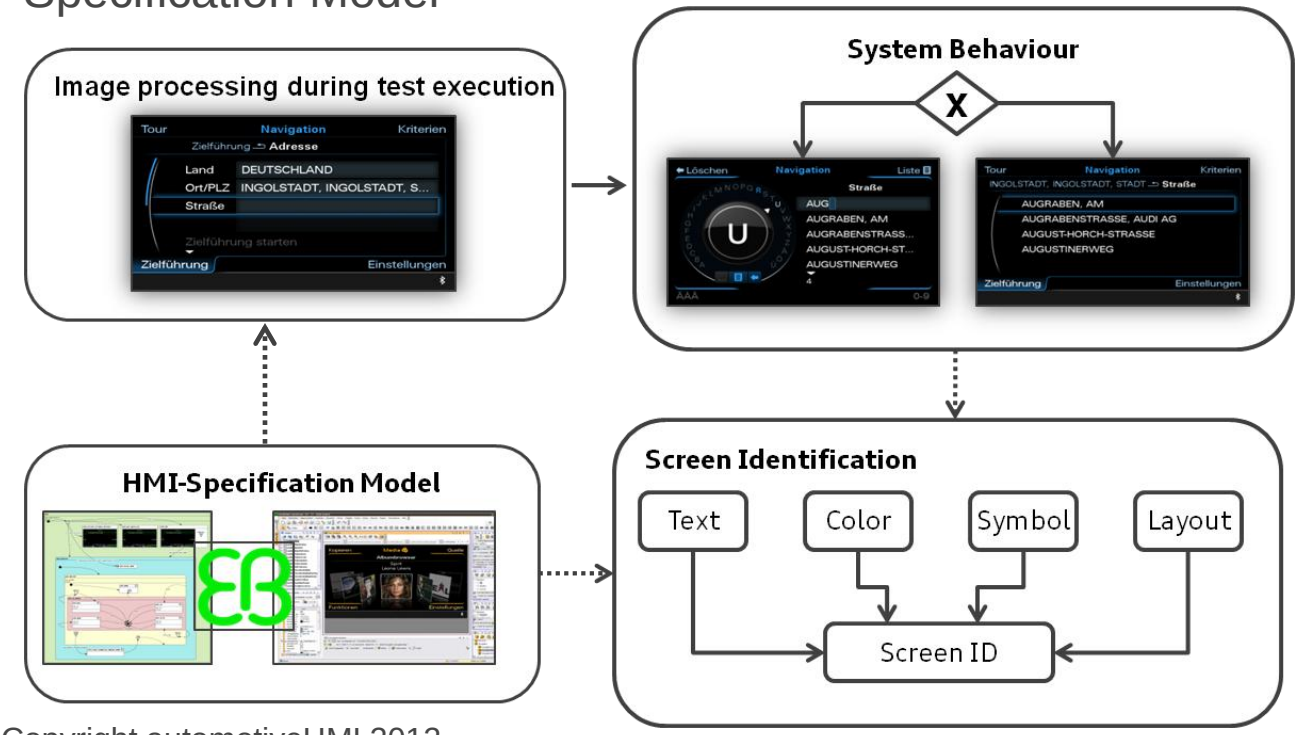
- automated generation of HMI Paths covering specific system features
- generated HMI Paths are extended by dynamic system behavior models to be able to react on system runtime behavior (e.g. media loading state or connected devices)



Example:
generated Navigation Tree for Car Module

Approach
AUDI

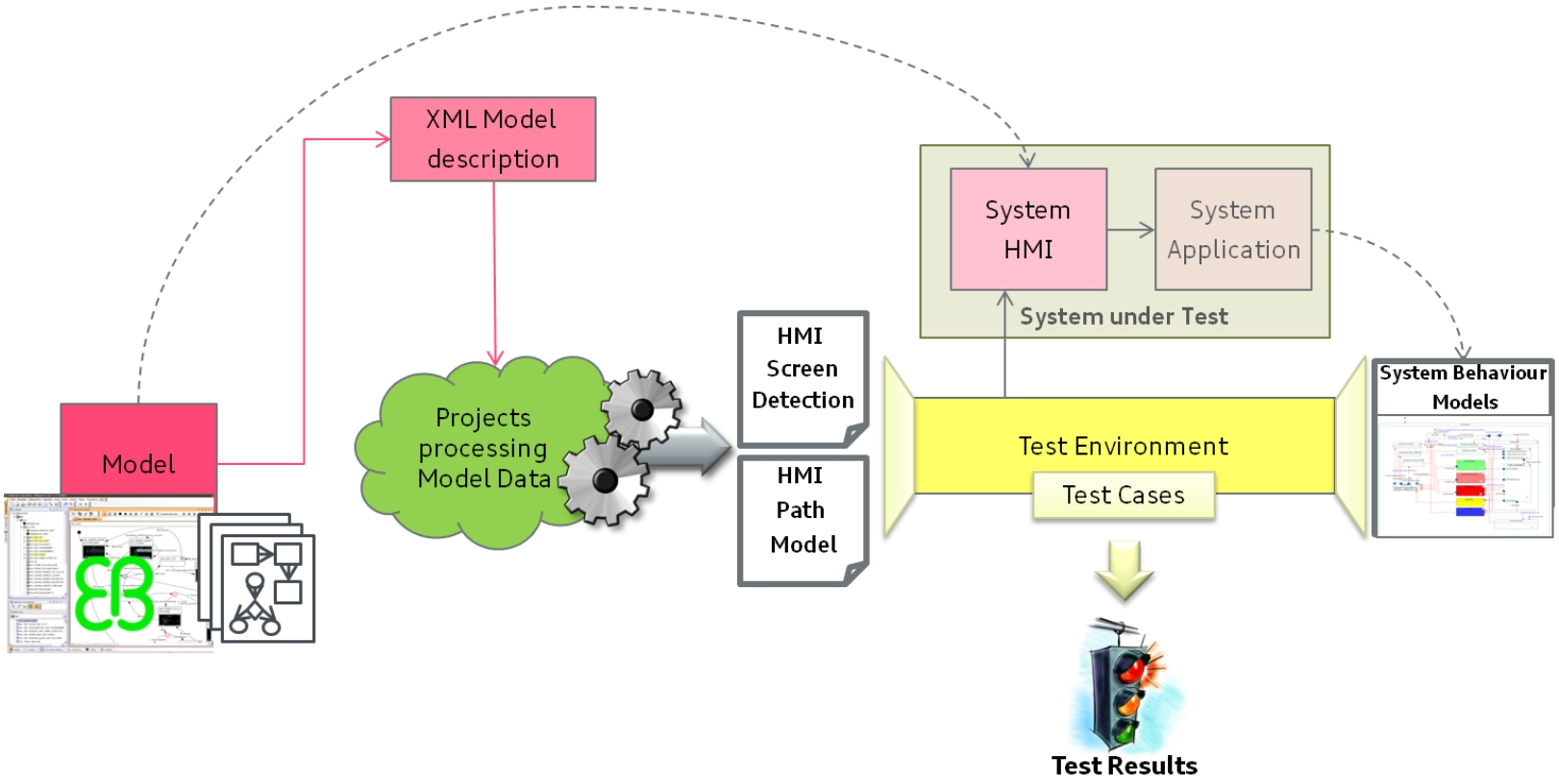
- Dynamic Screen Detection- & Analyze- Module to observe System Screens at runtime (derived from HMI-Specification Model)
- identifies current screen based on HMI Specification Model
- gives possibility to detect HMI deviations compared to the HMI Specification Model



Approach
AUDI

Audi
Vorsprung durch Technik

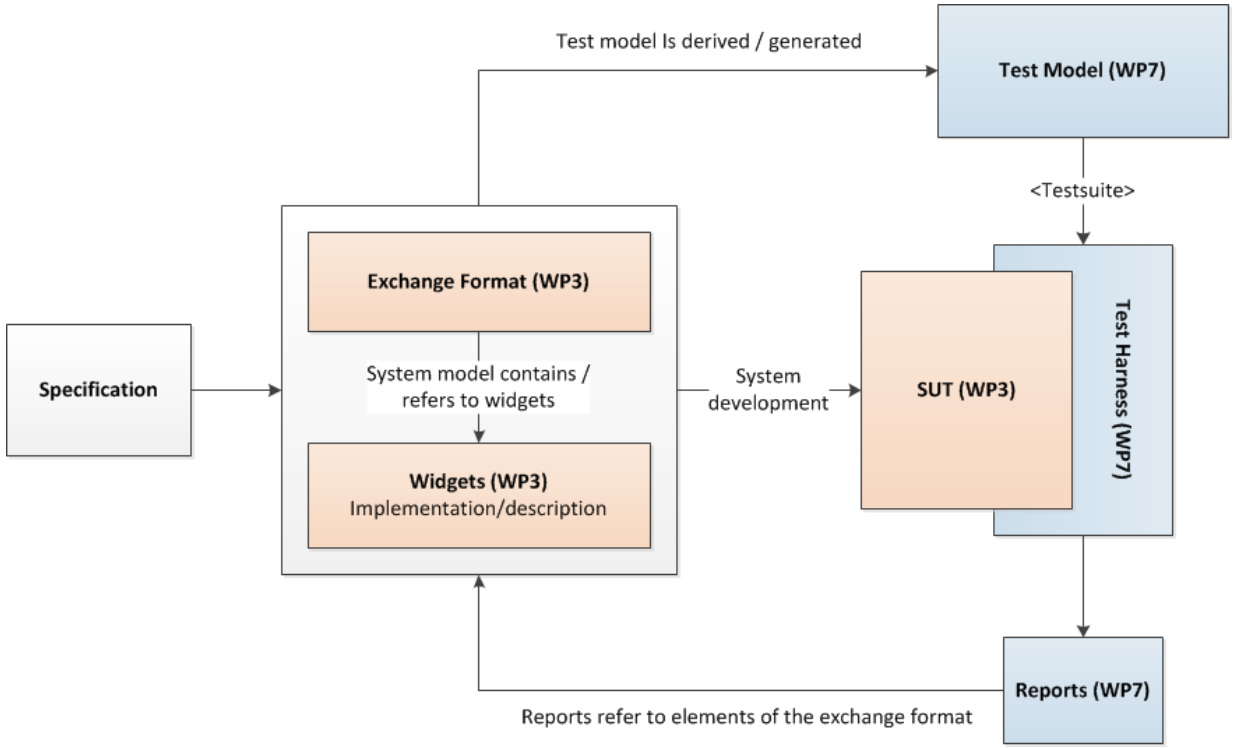
• Test Environment Suite combines the generated HMI Path Models, dynamic HMI Screen Detection- & Analyze- Module and the System Behavior Models to run tests on the system



DAIMLER

Methodology

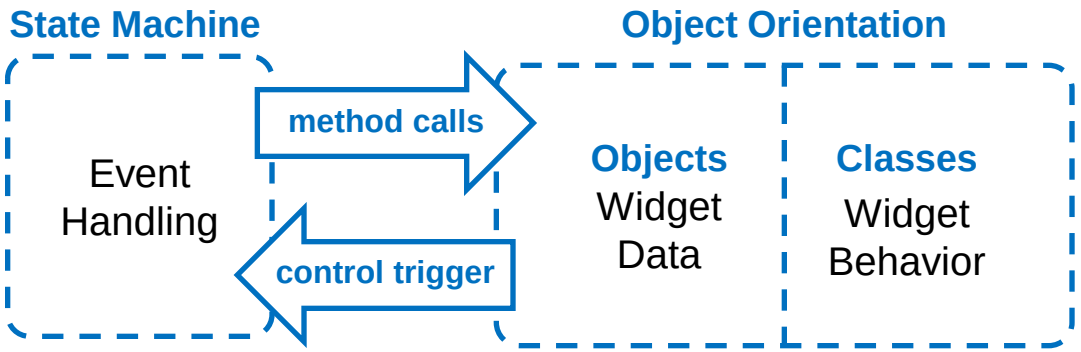
- Focusing behavior failures (Widget & Screen transition)
- Dedicated Test Model based on Exchange Format (WP3)
- Generated test cases are executed automatically



Object Oriented State Machines are used to create the Test Model

- Reactive aspects are captured using State Machines
  - Handling of User- and Middleware-Events
- A set of classes describes the data fields and behavior of the widgets (as basic building elements of the GUI)
  - Such as Buttons and Menus

⇨ Widget-Objects are instantiated and called as part of a State Machine. Usually, they are assigned to a particular state.



automotive HMI

DAIMLER

General Architecture of the Test Model

Screen

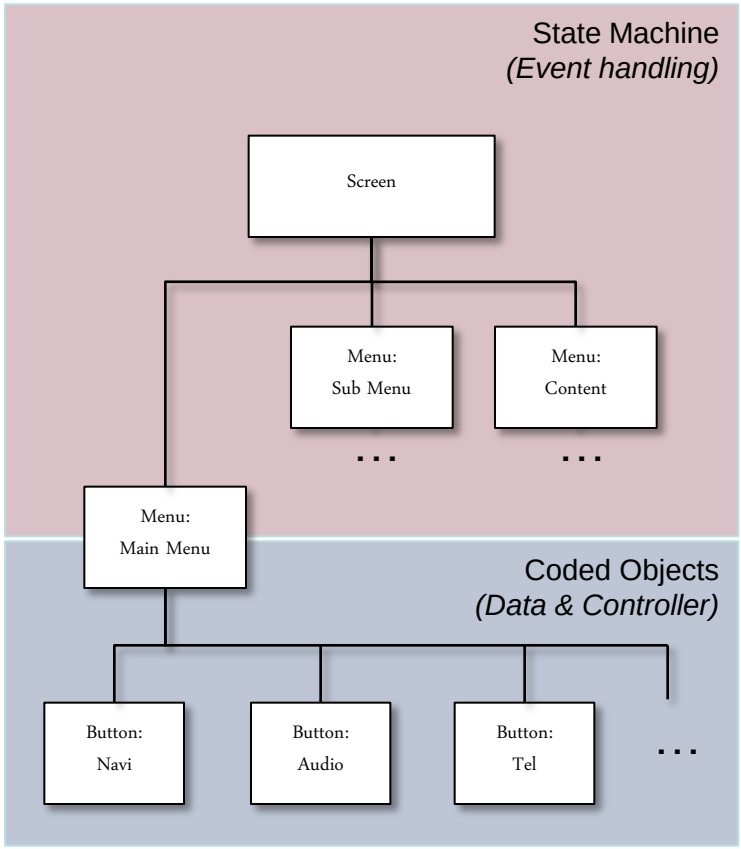
- Focus control
- Menu states
- Method calls

Menu

- Entry behavior
- Iteration logic

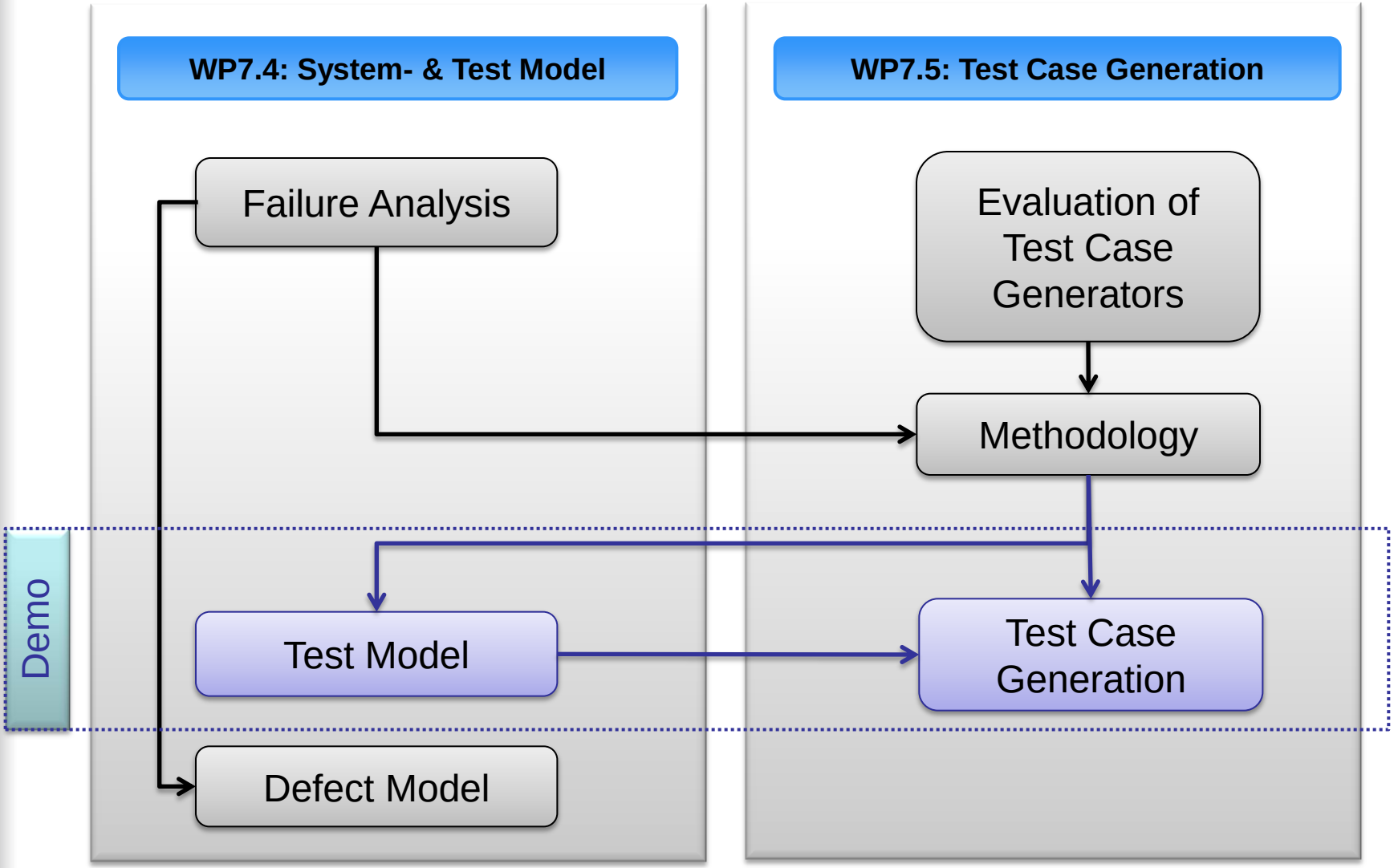
Button

- Availability
- Reference value
- Condition control
- Process trigger



automotive HMI

Focus: Current Work



Conclusion & Next Steps

Conclusion

- Failure analysis to set test focus
- Approach AUDI:
  - Test case execution based on HMI System Models
- Approach Daimler:
  - Dedicated Test Model (Object Oriented State Charts)

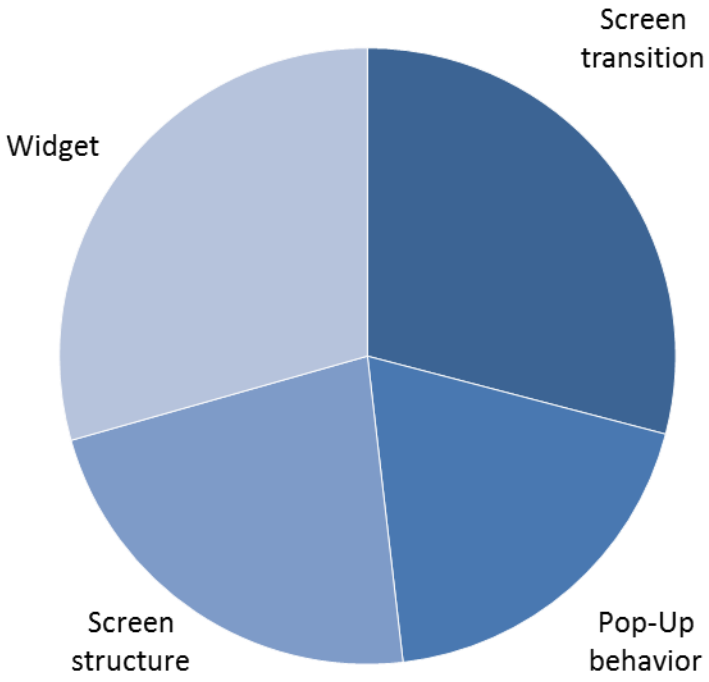
Ongoing & Next Steps

- Further developing the Test Model / Evaluation of different architectures
- Automatic test case execution
- Connection to Exchange Format
  - Test Model derivation
  - Failure Report references

automotive
HMI

Thank you for your attention.

Classification & Distribution – Behavior



Percentage (total) 61.5%

Screen structure (13.8%)

Any logic that determines what widgets the screen contains

Widget (18.1%)

Represents the micro behavior to navigate within screens

Screen transition (17.9%)

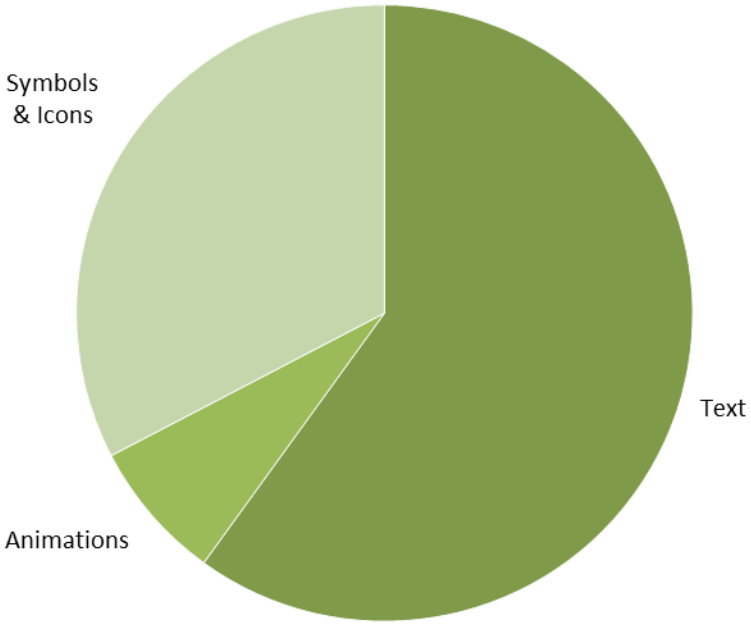
Any logic referring to a change of available menu structure

Pop-up behavior (11.7%)

System or application messages that overlay any content

automotive HMI

Classification & Distribution – Contents



Percentage (total) 25.1%

Contents (25.1%)

Text (15.1%)
The displayed text is wrong, missing, extra or incomplete

(e.g. the label of "Audio" button says "Blind text" instead)

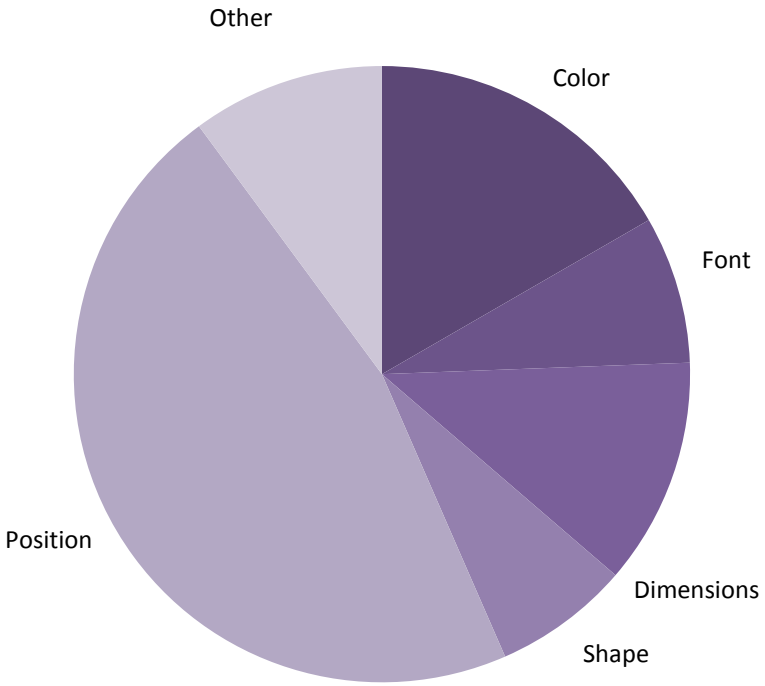
Icons & Symbols (8.2%)
The displayed Icon is either wrong, missing or extra

(e.g. the hang up icon is outdated)

Animations (1.8%)
An animation is either wrong, missing or extra

(e.g. on switching letters in the alpha numeric selector no animation shown)

Classification & Distribution – Design



Percentage (total) 5.8%

Design (5.8%)

Referring how content is displayed

Position *(e.g. a label of a button is centered instead of left-aligned)*

Other *(e.g. wrong clock)*

Color *(e.g. focused color is red instead of orange)*

Font *(e.g. text font is Times New Roman instead of Arial)*

Dimensions *(e.g. a button is higher or broader than specified)*

Shape *(e.g. a button should be displayed with rounded instead of sharp edges)*