Trustworthiness-by-Design and Maintenance at Run-Time

Nazila Gol Mohammadi ▪ FOMSESS 18.02.2016

PALUNO
The Ruhr Institute for Software Technology

OPTET
Operational Trustworthiness Enabling Technologies

SEVENTH FRAMEWORK PROGRAMME

Outline

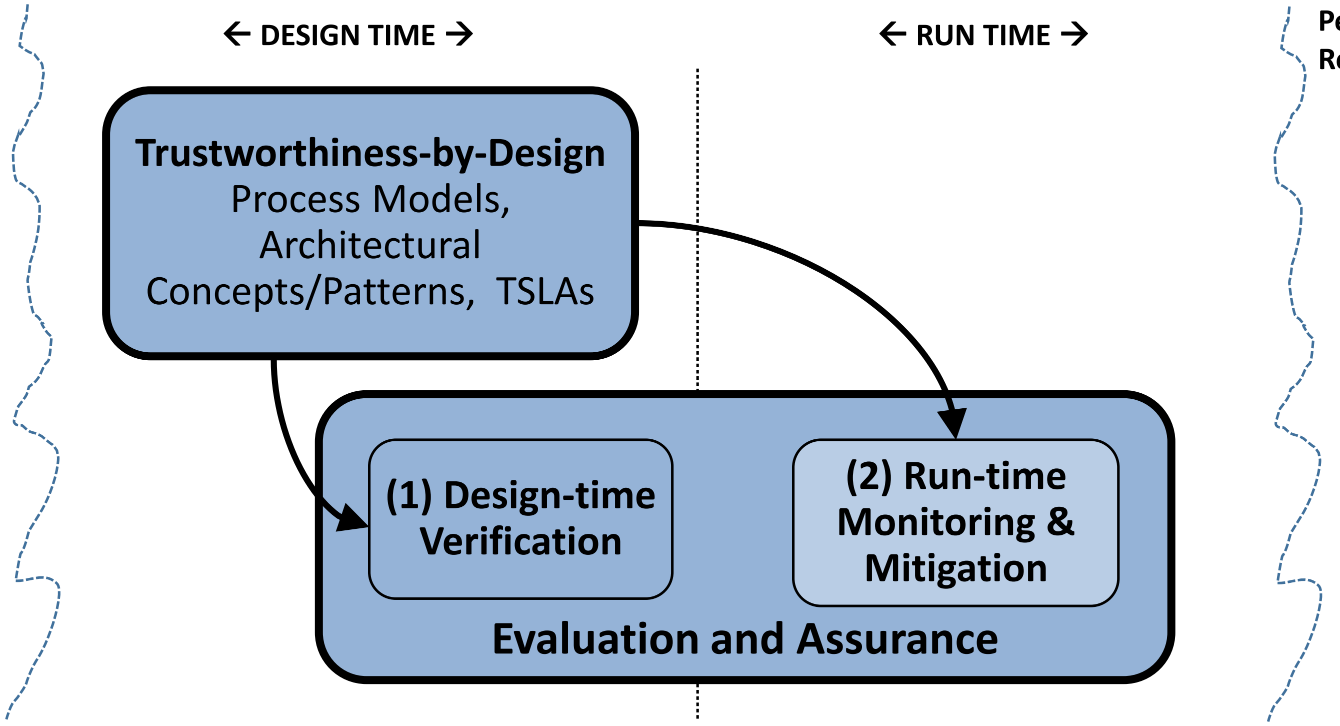
- Background & Motivation
- Contributions
  - Trustworthiness-By-Design
  - Trustworthiness Modeling
  - E2E Trustworthiness Evaluation
- Conclusion & Future Work

Background and Motivation

Motivation

Certification

Prohibitively expensive



Personal Trust Relationships

Scalability issues in Internet setting (e.g., complex service-value chains)

“Trustworthiness-by-Design”

“Trustworthiness Monitoring and Maintenance”

Socio-Technical Systems

“Application, services and platforms, where technology and human behaviour are mutually dependent and strongly influence each other.”

- –“STS include humans, organizations, and their information systems”

[Whitworth 2003, Amit 2011, Baxter 2012]

Characteristics

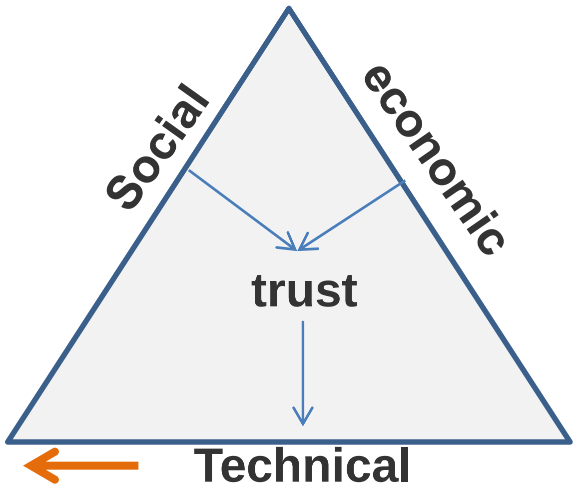
- **<u>Interaction</u> between autonomous participants (Human, organizations as social actors, information systems and software systems as technical actors) with <u>different perceptions and goals</u>**
  - →includes social actors and technical actors

Examples

- Healthcare systems/Patient monitoring systems, Market places, Social networks

Trust and Trustworthiness

- **Trustworthiness of a system is relevant throughout its life cycle**
  - **Design-time activities**
  - **Run-time activities**



provide indicators/
evidence for trustworthiness

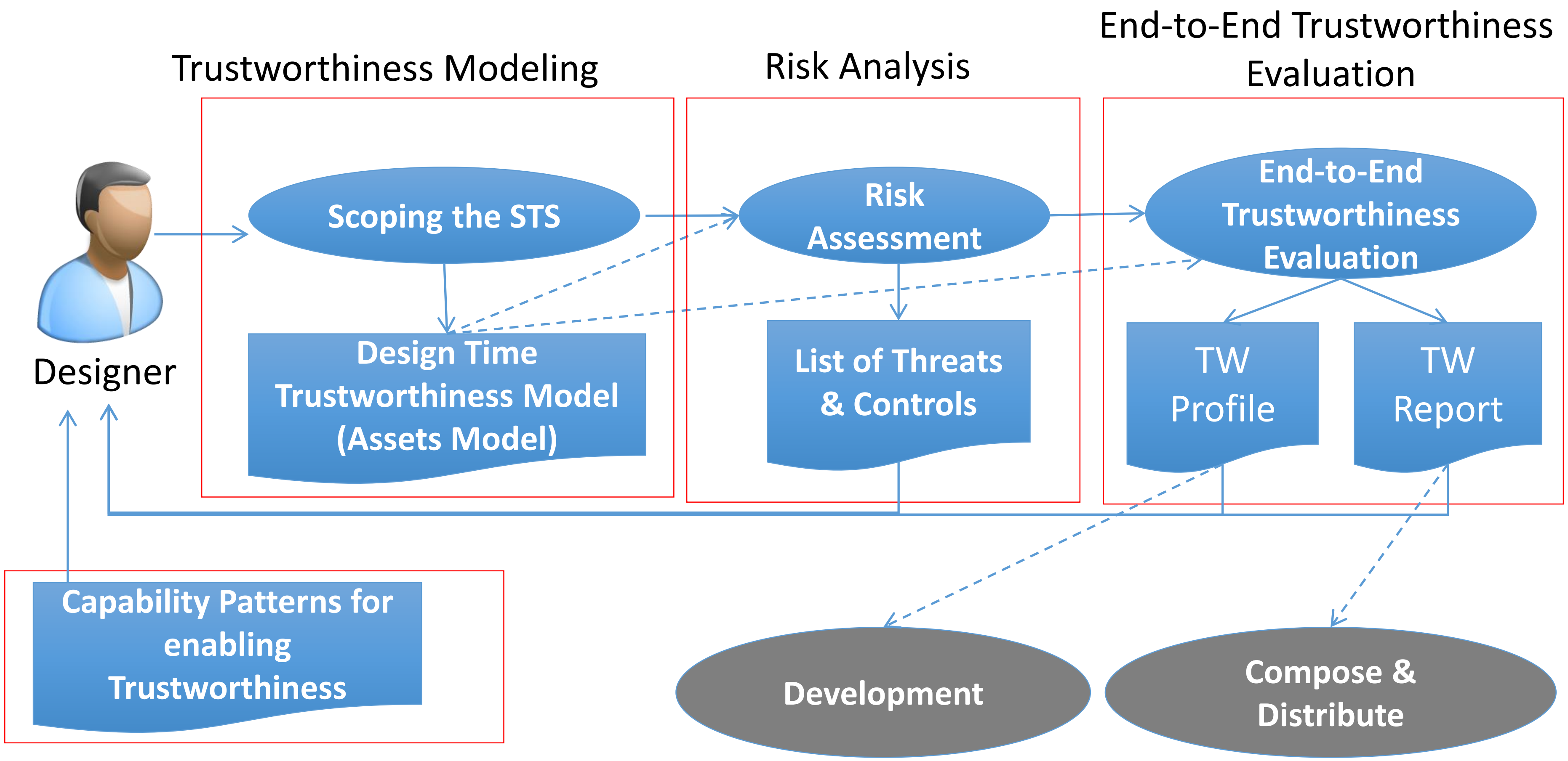
<u>Trustworthiness</u>

Enforced; legally or organizationally guaranteed

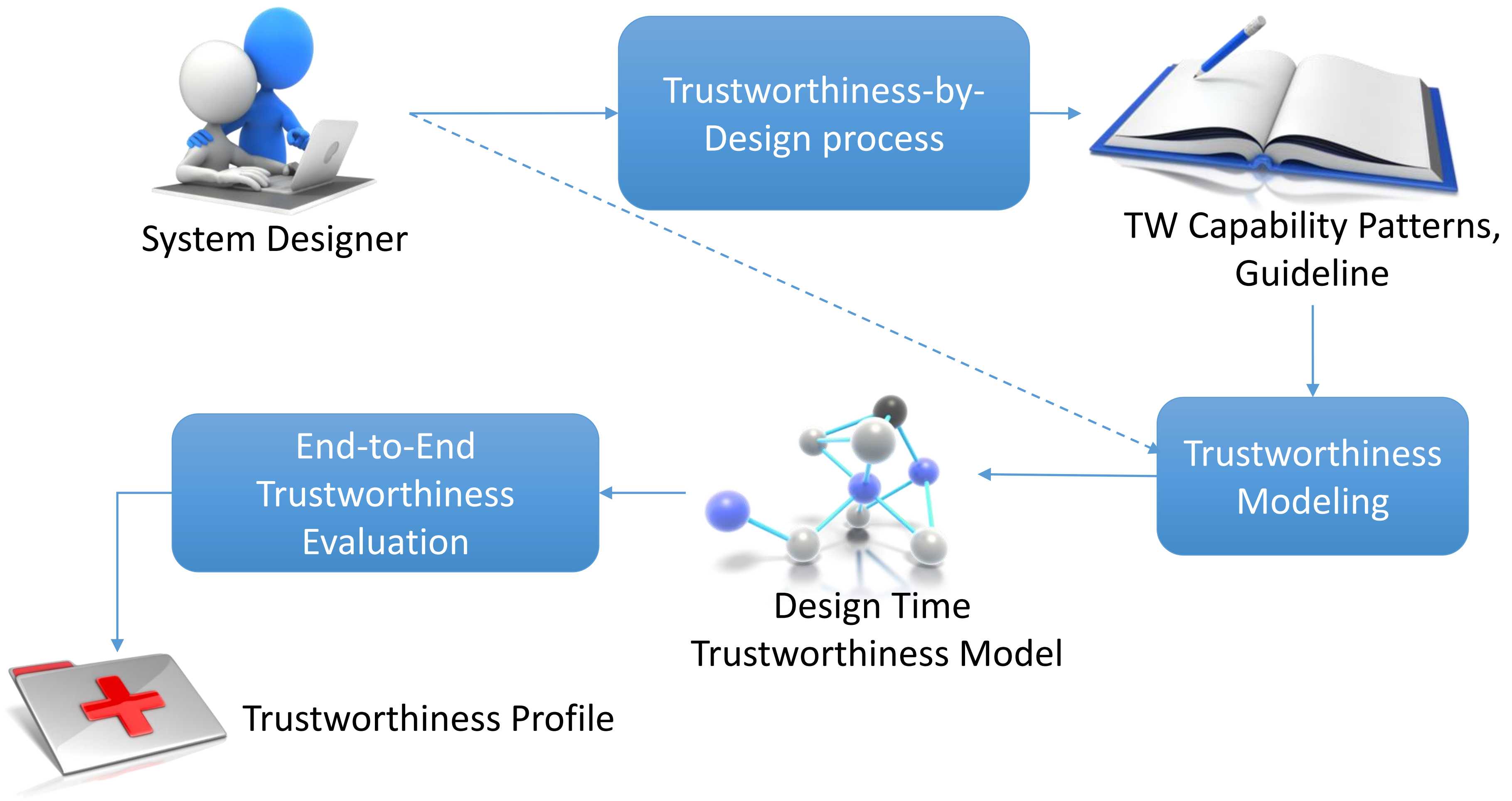
Inspired from [Stefik 2000]

Bringing Trustworthiness into Process

UNIVERSITÄT DUISBURG ESSEN
Open-Minded
PALUNO The Ruhr Institute for Software Technology



Design Phase – Interactions



CONTRIBUTIONS

- Trustworthiness-by-Design
- Trustworthiness Model Editor
- End-to-End Trustworthiness Evaluation

Terminology

- Capability Patterns: Process building blocks represent best development practices for specific disciplines, technologies, or management styles
  - quick assembly of processes based on project/application-specific needs
- Trustworthiness Profile: Determining trustworthiness requirements by specifying expected target values for Trustworthiness metrics
- Trustworthiness Report: Evaluates a <u>certain system composition</u> in terms of:
  - Metric values,
  - Attributes values and
  - An overall trustworthiness value

CONTRIBUTIONS

- Trustworthiness-by-Design
- Trustworthiness Modelling
- End-to-End Trustworthiness Evaluation

Trustworthiness Modelling

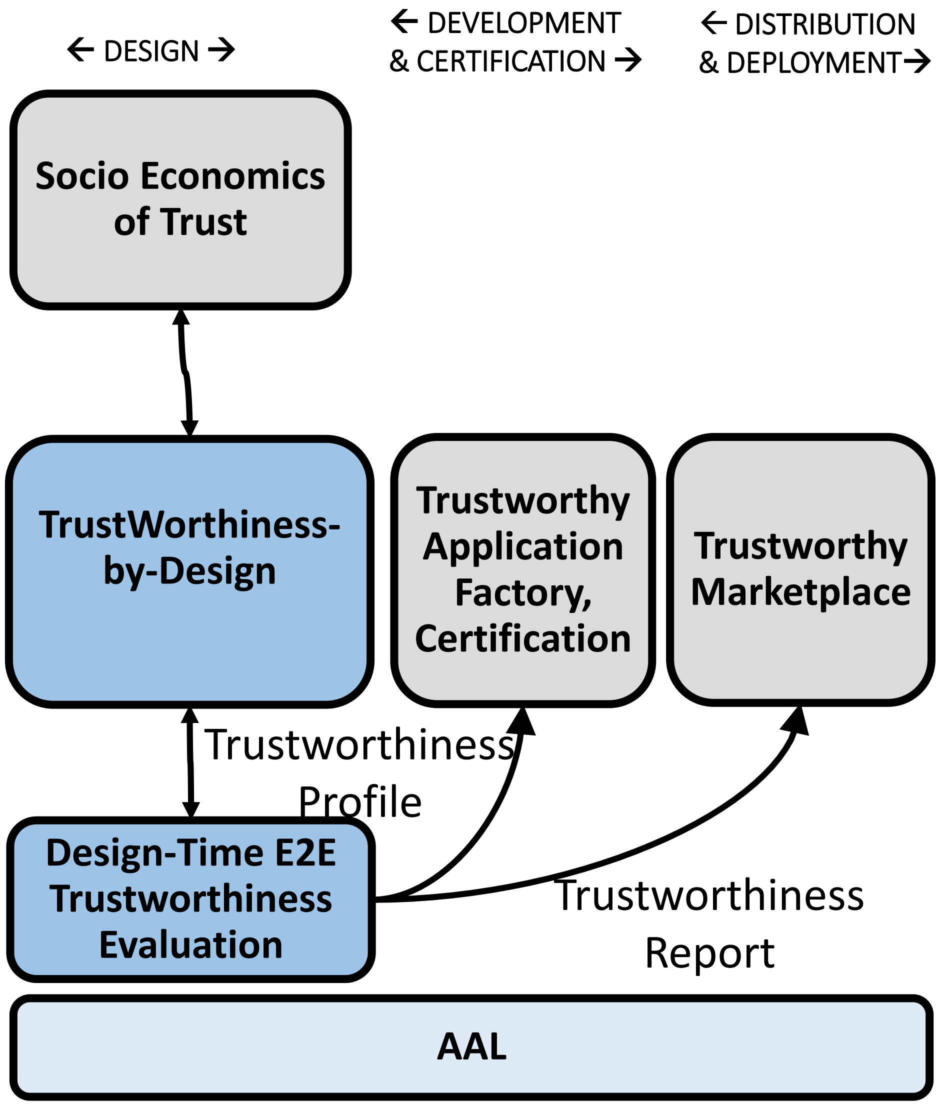
- A tool to model Trustworthy Socio-Technical Systems
  - Uses background knowledge
  - Features:
    - Semi-automated completion of design-time trustworthiness models
    - Fully-automated threat identification and analysis
    - Knowledge sharing –risk analysis report generation

CONTRIBUTIONS

- Trustworthiness-by-Design
- Trustworthiness Modelling
- End-to-End Trustworthiness Evaluation

End-to-End Trustworthiness Evaluation

- Goals & Objectives
  - Quality assurance of Trustworthiness-by Design Process
  - End-to-End Trustworthiness Evaluation Techniques and Tools for the Analytical Assessment of Trustworthinesss

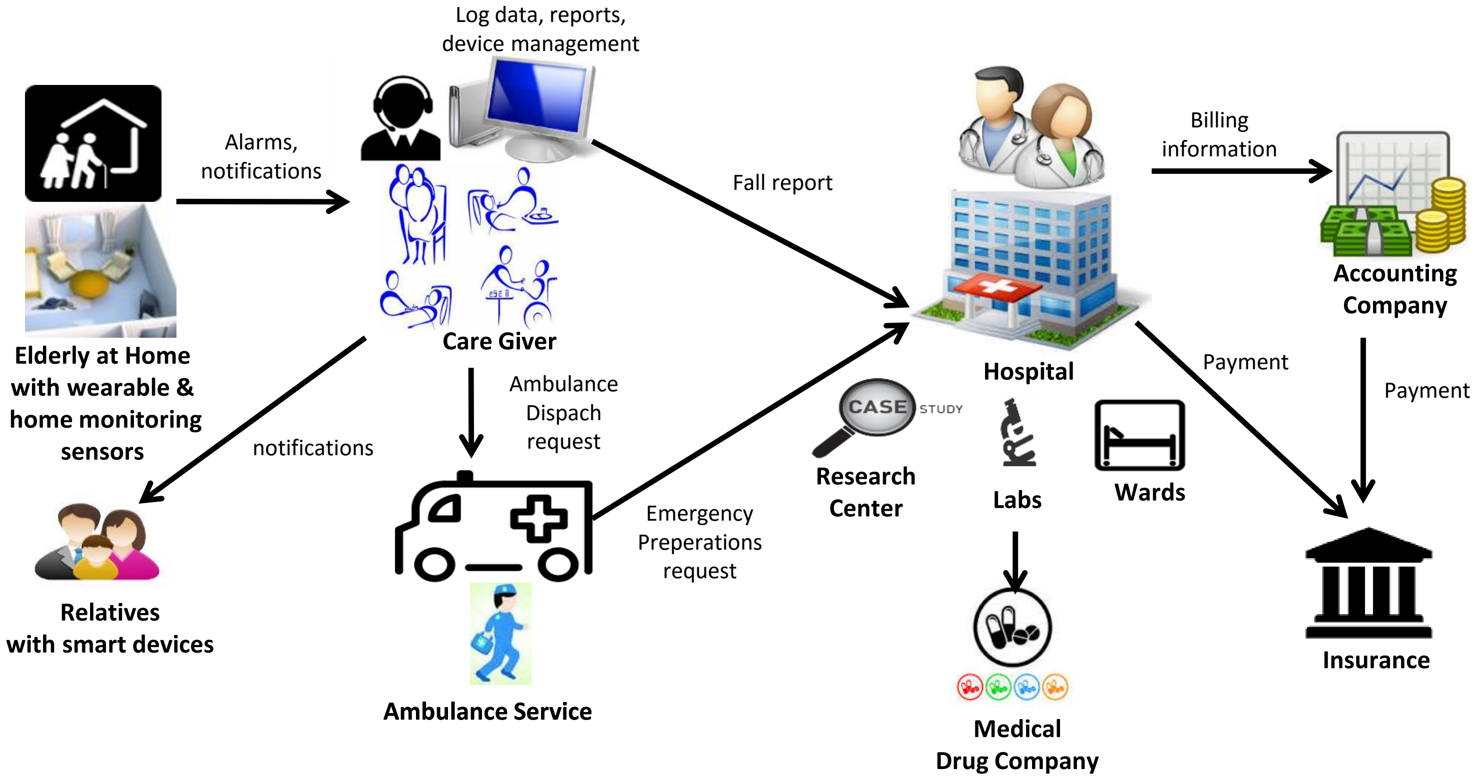


End-to-End TrustWorthiness Evaluation Tool

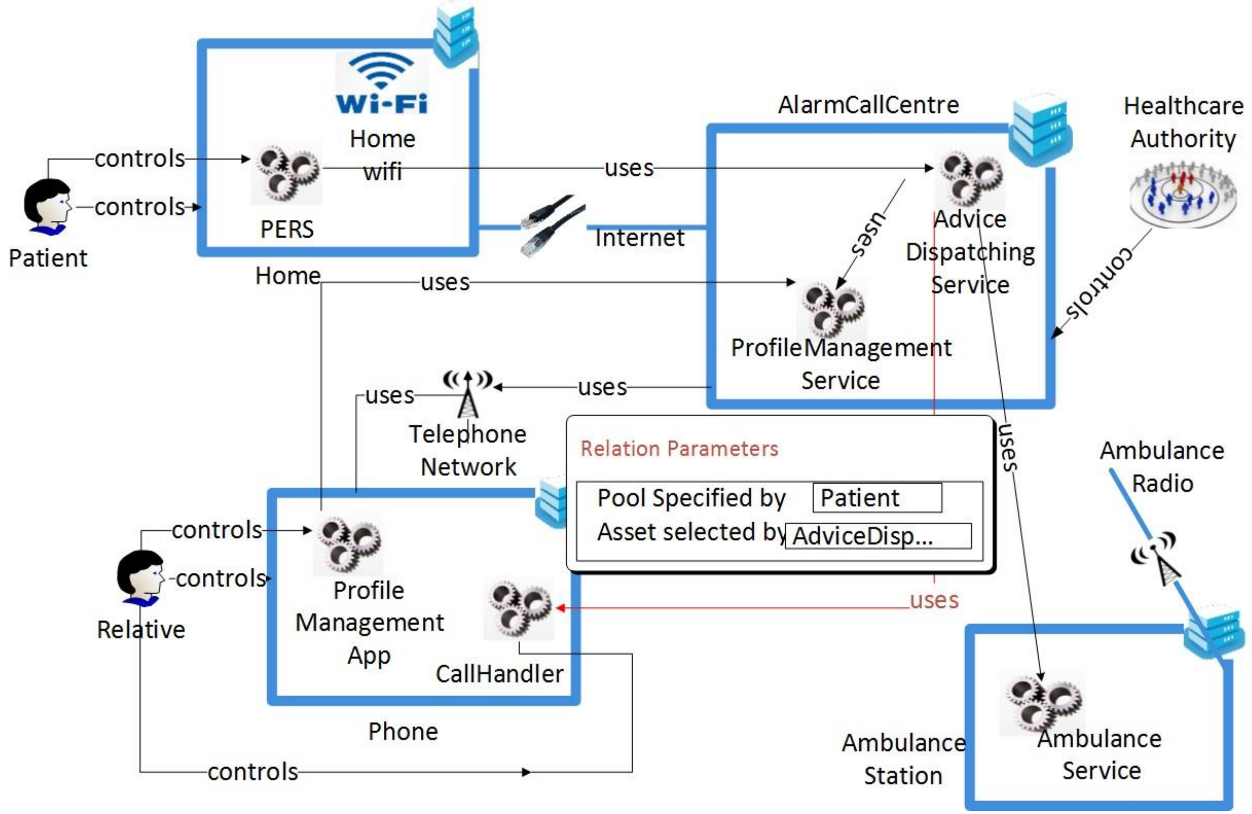
- A tool to Assess and Evaluate the End-to-End Trustworthiness of Socio-Technical Systems
  - Uses computational models for analytical assessment complementary risk assessment approach (threats and possible controls)
  - Features:
    - Requirements determination → Trustworthiness Profile
    - End-to-End Evaluation of the System Composition → Trustworthiness Report
  - Benefits:
    - Supporting designer of STS in documentation, and making design choices

Ambient Assisted Living Use Case

UNIVERSITÄT DUISBURG ESSEN
Open-Minded
PALUNO
The Ruhr Institute for Software Technology



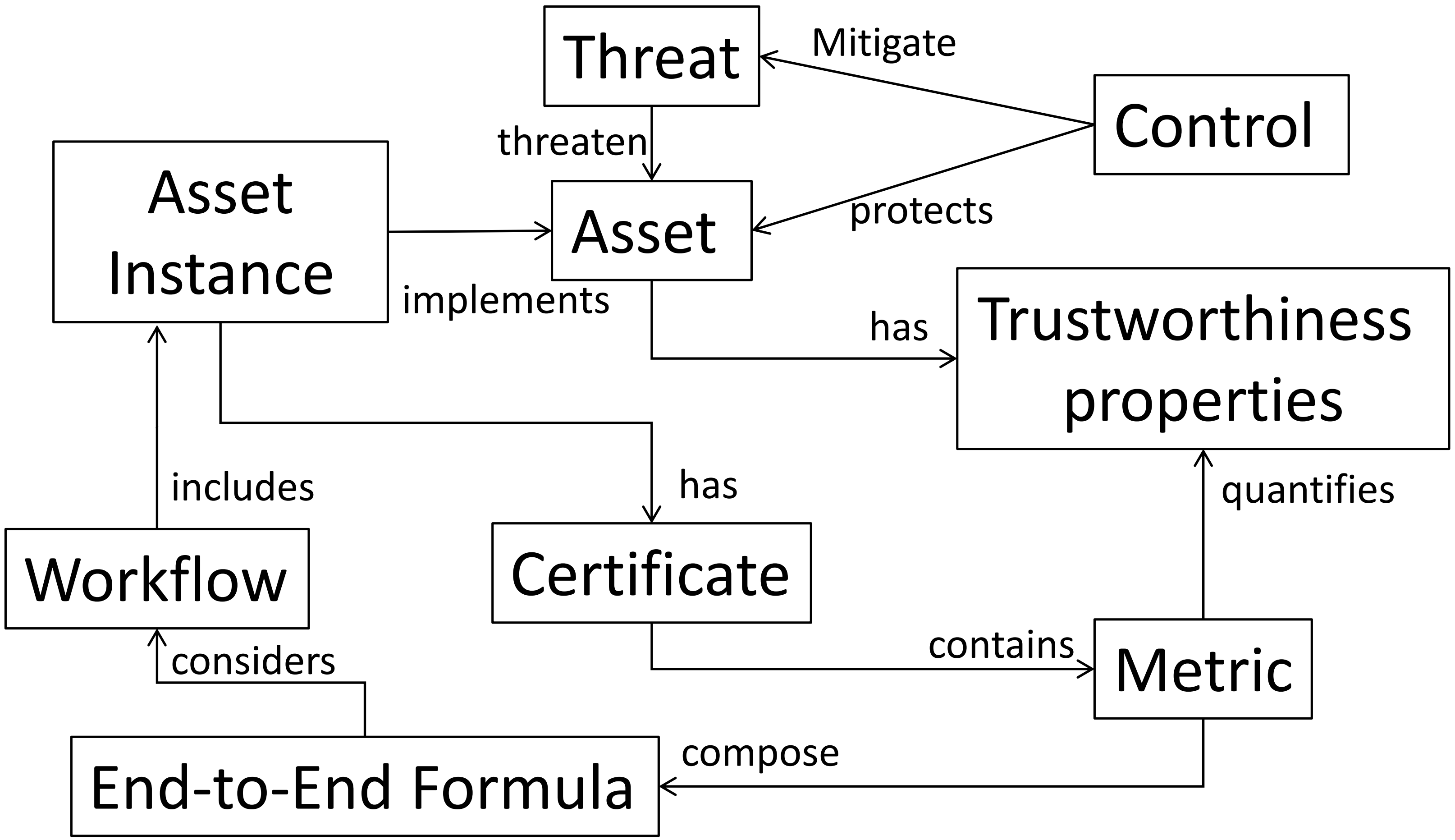
Design-Time Trustworthiness Model for AAL



- Model of Ambient Assisted Living Use Case

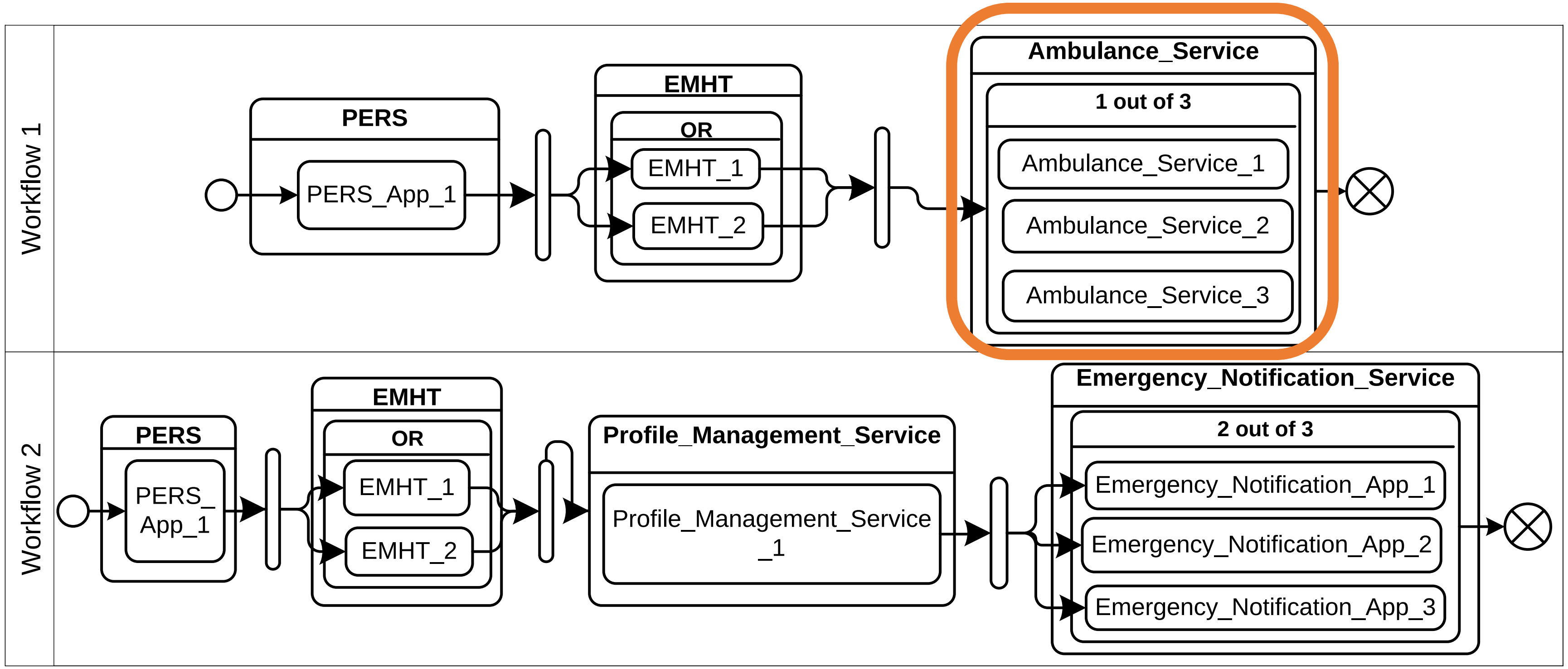
1. [OPTET]

Concepts of End-to-End Trustworthiness Evaluation



Workflows of Using the Exemplary Health Care System

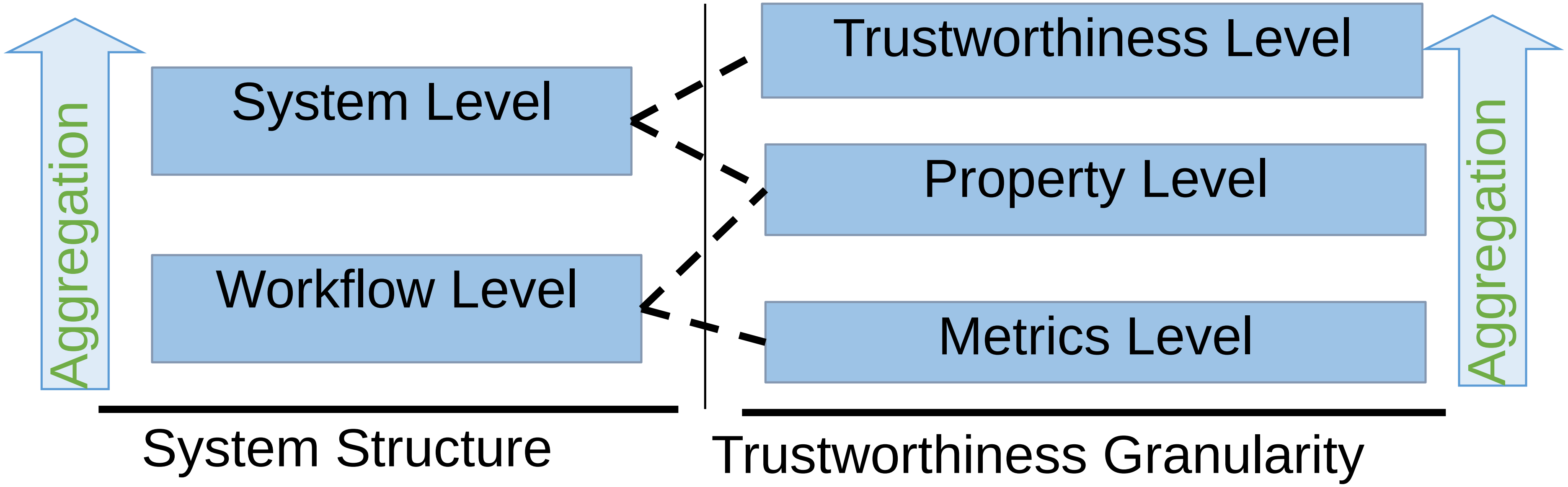
Sequential structures, including **redundant** instantiations of the system assets (as a means to increase trustworthiness)



Our Framework

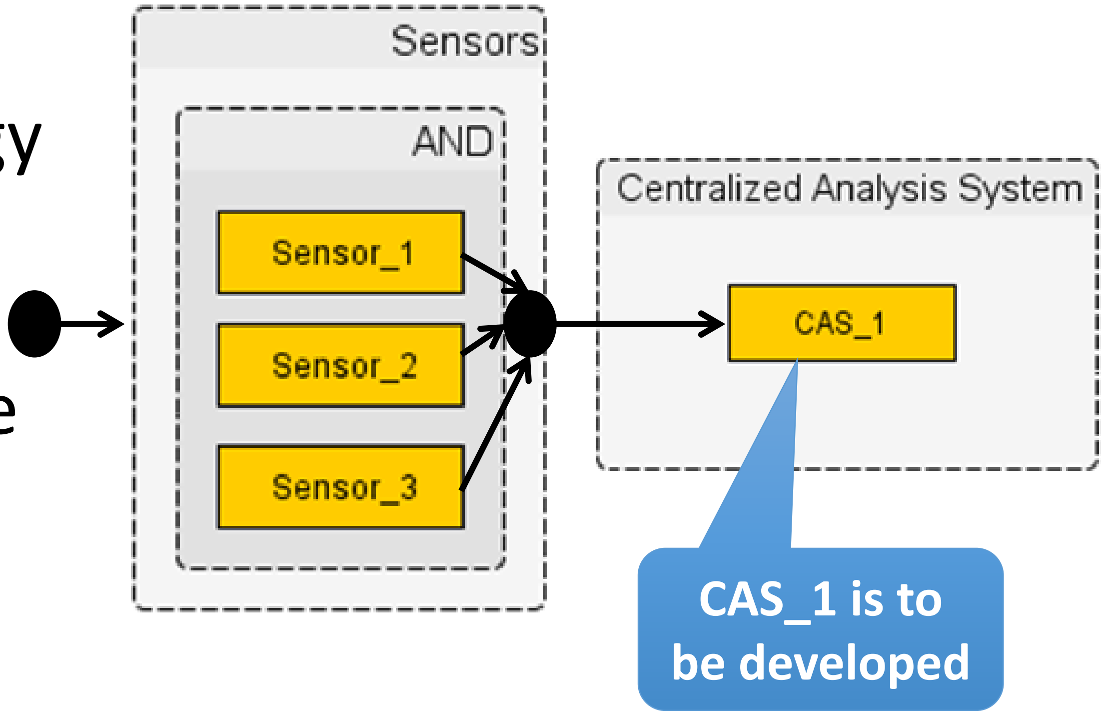
Two dimensions:

- Overall **system structure** (from a set of separate workflows to the complete system)
- Level of **granularity of the end-to-end trustworthiness computation** (from individual metric values to the overall „end-to-end trustworthiness“)



Workflow for End-to-End Trustworthiness Evaluation

- Workflow
  - Structure/topology of the Asset Instances
  - Focus on Software Assets
  - Redundancy Groups
  - Redundancy Type



Trustworthiness Properties and Metrics

Selected Trustworthiness properties

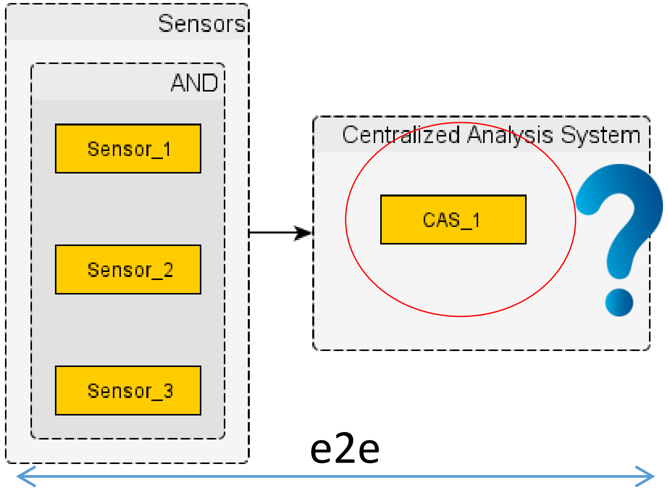
- Response Time
- Accuracy
- Openness
- Availability

Examples of Corresponding Metrics

- Average response time
- Error Rate
- Software Precision
- Documented Public Interfaces
- Functions with Semaphores

End-to-End Formula Composition (1)

- Different kind of Metrics are considered
  - Multiplicative, Concave
  - Higher or Lower values are desired
- Composition of a Formula for calculation of End-to-End Trustworthiness Value for given Workflow



Trustworthiness Profile

- Includes:
  - Relevance to Trustworthiness Attributes
  - Corresponding Metrics to the Trustworthiness Attributes
  - Target/Desired Metric values
- Benefit:
  - A reference specification document for Trustworthiness Requirements

Metric ID	Name	TW Properties	Expected Value %
1108	*Average response time*	Response Time	80%
163	*Documented Public interfaces*	Openness	75%
1104	*Error Rate*	Accuracy	0,5192%
1105	*Software Precision*	Accuracy	70%
230	*Functions with Semaphores*	Availability	98,9583%

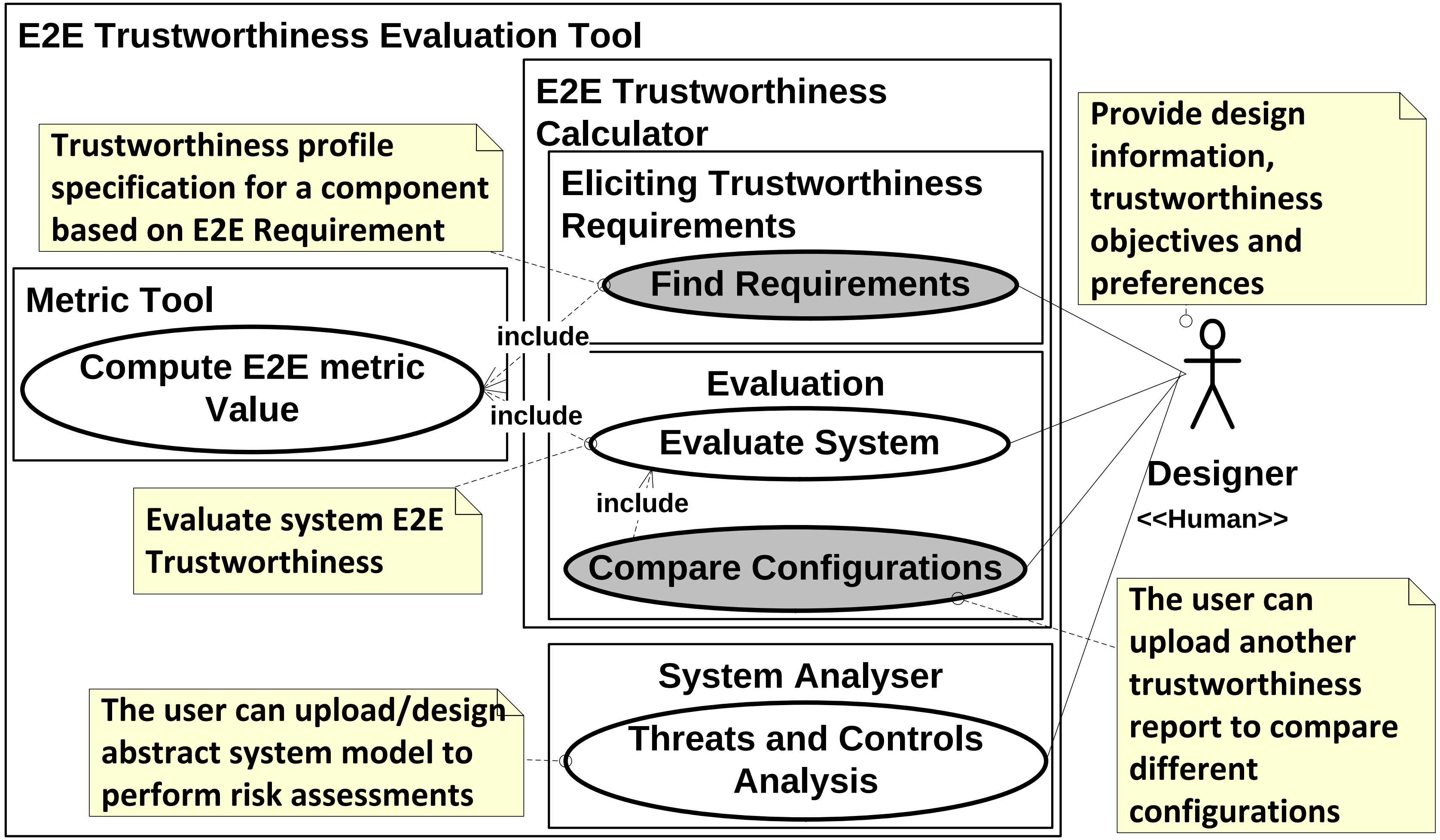
CONCLUSION & FUTURE WORK

Conclusion and Future Work

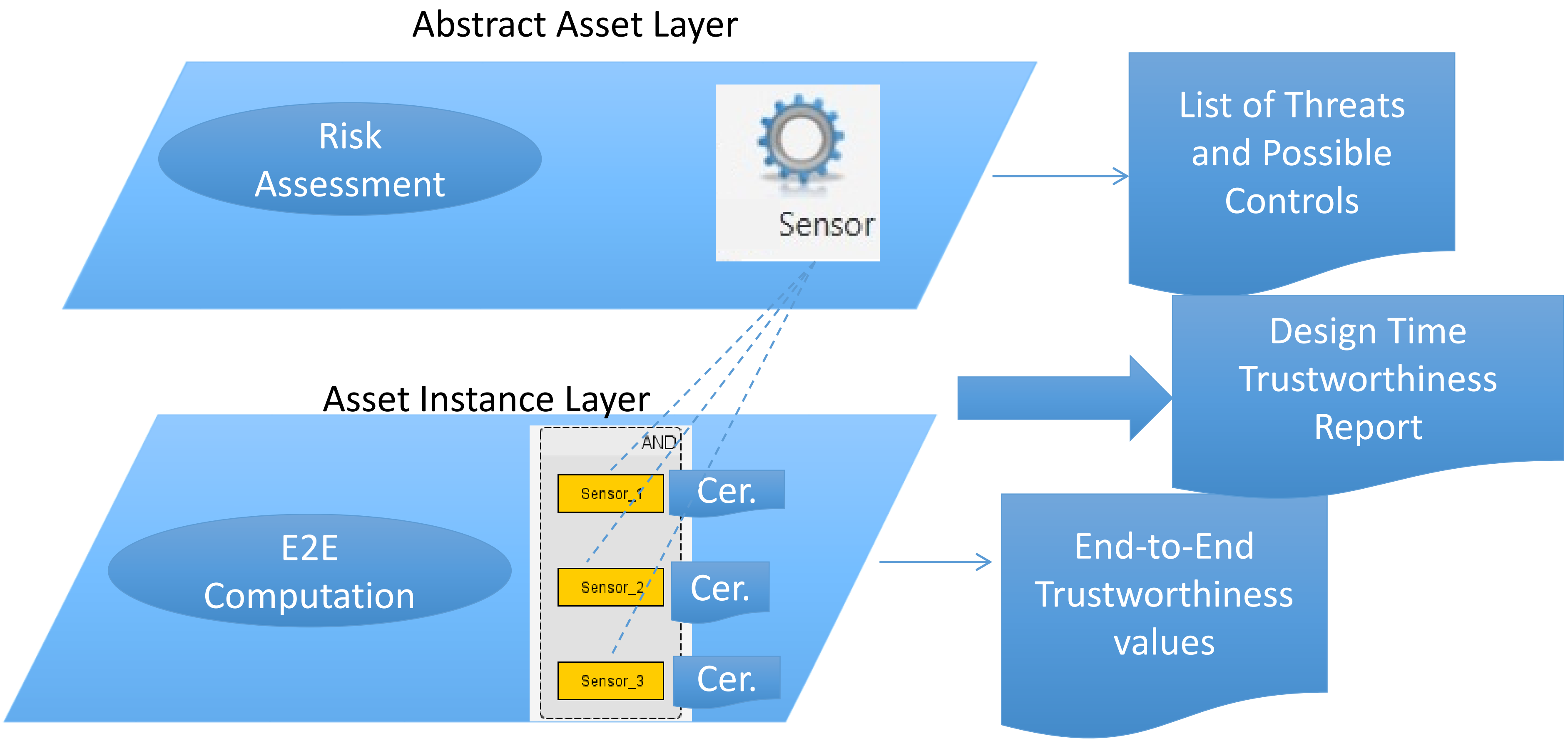
- Contribution: A comprehensive framework for evaluating end-to-end trustworthiness, which
  - is based on, and unifies existing work in the areas of service composition, trustworthiness attributes, and metrics
  - considers various system redundancy structures and metrics
  - guides the definition of end-to-end metrics
  - guides the steps to be taken for aggregating end-to-end values on different levels of granularity
- Fundamental precondition: meaningfulness of the underlying metric values for individual components
- Next steps:
  - Providing more detailed guidance on how to use the framework
  - Investigating the use of risk-based approach complementing the end-to-end calculation

BACK-UP SLIDES

Provided Functionalities in E2E TWE



Risk Assessment and the Computational Approach



Our Framework

Computing the End-to-End Trustworthiness Evaluation

End-to-end formulas for individual metrics can be composed based on the following characteristics:

- Metric type
- Metric target type
- Redundancy type

Creating an end-to-end formula:

1. Apply formula skeletons to the assetS, reflecting their redundancy type.
2. Determine the complete end-to-end metric composing the different asset parts.

Metric Type	Metric Target Type	Asset Category Redundancy Type	Formula of Asset Category j	Formula for E2E Metric
Concave	Higher values	-	$a_j = \min_i m_i$	$e = \min_j a_j$
	Lower values		$a_j = \max_i m_i$	$e = \max_j a_j$
Multiplicative	Higher values	AND	$a_j = \prod_{i=1}^{n} m_i$	$e = \prod_j a_j$
		OR	$a_j = 1 - \prod_{i=1}^{n} (1 - m_i)$	
		K-out-of-N	$a_j = \sum_{i=k}^{n} \binom{n}{i} m^i * (1-m)^{n-}$	
	Lower values	AND	$a_j = \prod_{i=1}^{n} m_i$	$e = 1 - \prod_j a_j$
		OR	$a_j = 1 - \prod_{i=1}^{n} (1 - m_i)$	
		K-out-of-N	$a_j = \sum_{i=k}^{n} \binom{n}{i} m^i * (1-m)^{n-}$	
Additive	Both	-	$a_j = \sum_{i=k}^{n} m_i$	$e = \sum_j a_j$

End-to-End Formula Composition (1)

- Different kind of Metrics are considered
  - e.g., Multiplicative, Concave, Additive
  - Higher or Lower values are desired
- Composition of a Formula for calculation of End-to-End Trustworthiness Value for given Workflow

Introduction and Background
Problem Statement

- Component-Based Software Engineering: Compose systems by orchestrating existing software services
  - Components provided on some online marketplace (e.g., Amazon Web Services Marketplace [10])
  - Individual certificates including different metric values as evidence (cf. [7, 26])
- Challenges:
  - How to consider different **system composition structures or topologies**, such as **redundancy**?
  - How to consider **trustworthiness** as a **multi-faceted** and broad spectrum of system quality properties?
  - How to consider different types of quality/trustworthiness metrics?

→ Metric values have to be **aggregated** in order to determine „end-to-end“ values on appropriate **levels of abstraction**

Fundamentals and Related Work

- **Constructs** for aggregating metric values (cf. e.g. [13, 14]):
  - E.g., sequence, parallel, multichoice-multimerge, loop
- Software quality attributes contributing to trustworthiness (denoted „**trustworthiness attributes**“) and **metrics** [6]
  - Including reliability, usability, confidentiality etc.
- Different **types of metrics** [18]:
  - Additive, multiplicative, and concave

- Lack of a **comprehensive framework** that takes into account versatile system structures, as well as a large set of trustworthiness attributes and metrics

Trustworthiness Model Editor Guideline Example

Elements of Capability Patterns	Content in CP: Identification of Threats and Mitigating Controls	Relevant support by developed Tool: Trustworthiness Model Editor
Role	Risk Analyzer	System designer with the knowledge in Asset modeling and risk analysis and management processes
Task	Identify threats and controls [system scoping, modelling, risk assessments, etc.]	Supporting the tasks for creating the asset models, automated identification of threats and a list of possible controls
Work products	Asset model, Statement of threats and controls	Design Time Trustworthiness Model, threat and controls list based on the model

Trustworthiness Profile

- Elements of Trustworthiness Profile
  - List of the required TWAtrributes with the associated metrics and the expected value for each metric
  - All the information must be put into the TWProfile with the following representation (The TWA name, attributeID, Metric name and metricID are coming from the metric tools):

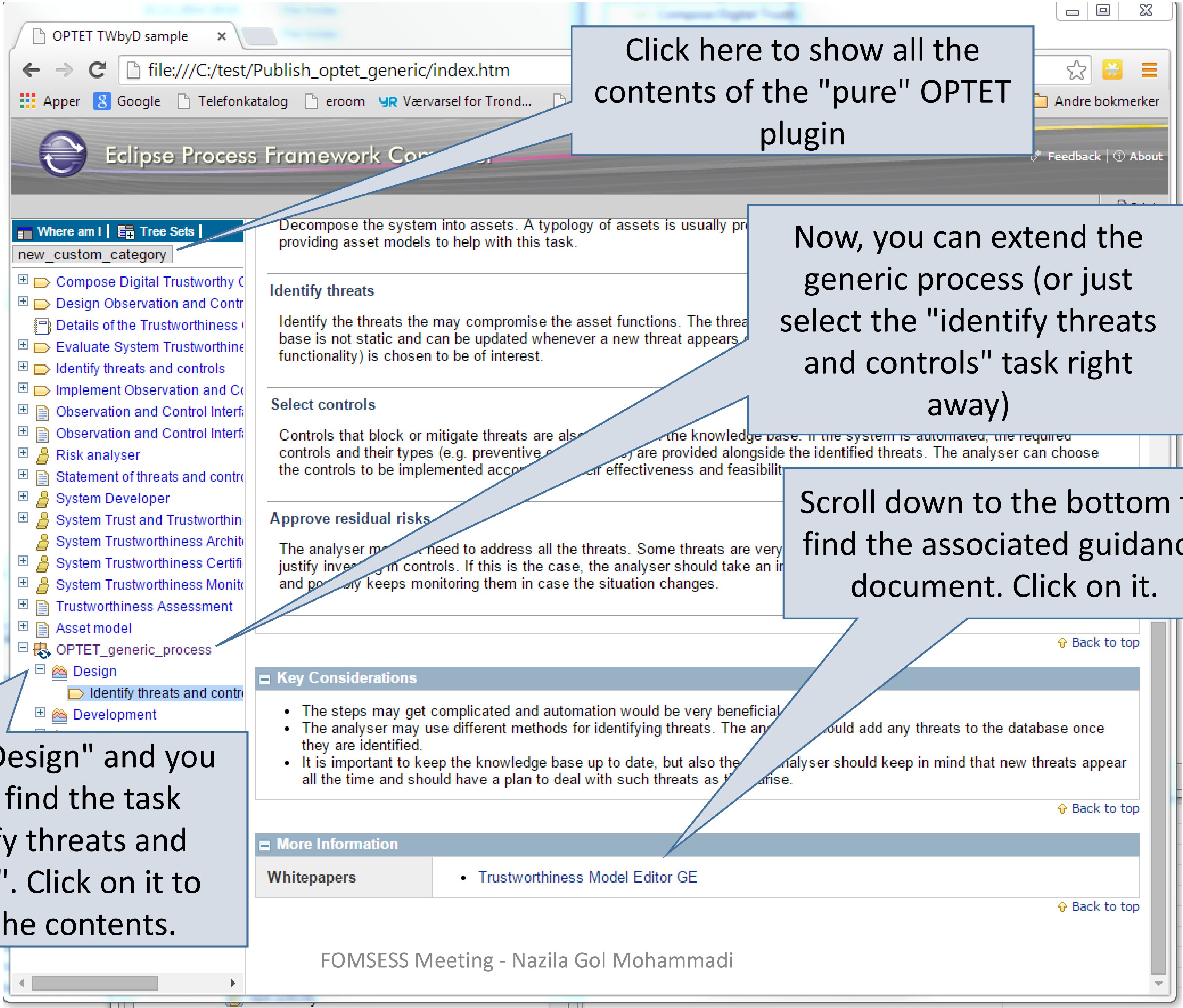
<TWProfile>
<TWAttribute name =" Availability " *attributeID="39" >*
<Metric name=" availability of Software " *metricID="230" expectedValue="1"/>*
</TWAttribute>
<TWAttribute name =" Accuracy " *attributeID="38" >*
<Metric name=" accuracy calculation " *metricID="225" expectedValue="0,6"/>*
</TWAttribute>
<TWAttribute name =" Performance " *attributeID="XX" >*
<Metric name="Performance calculation " *metricID="YYY" expectedValue="0,9"/>*
</TWAttribute>
</TWProfile>

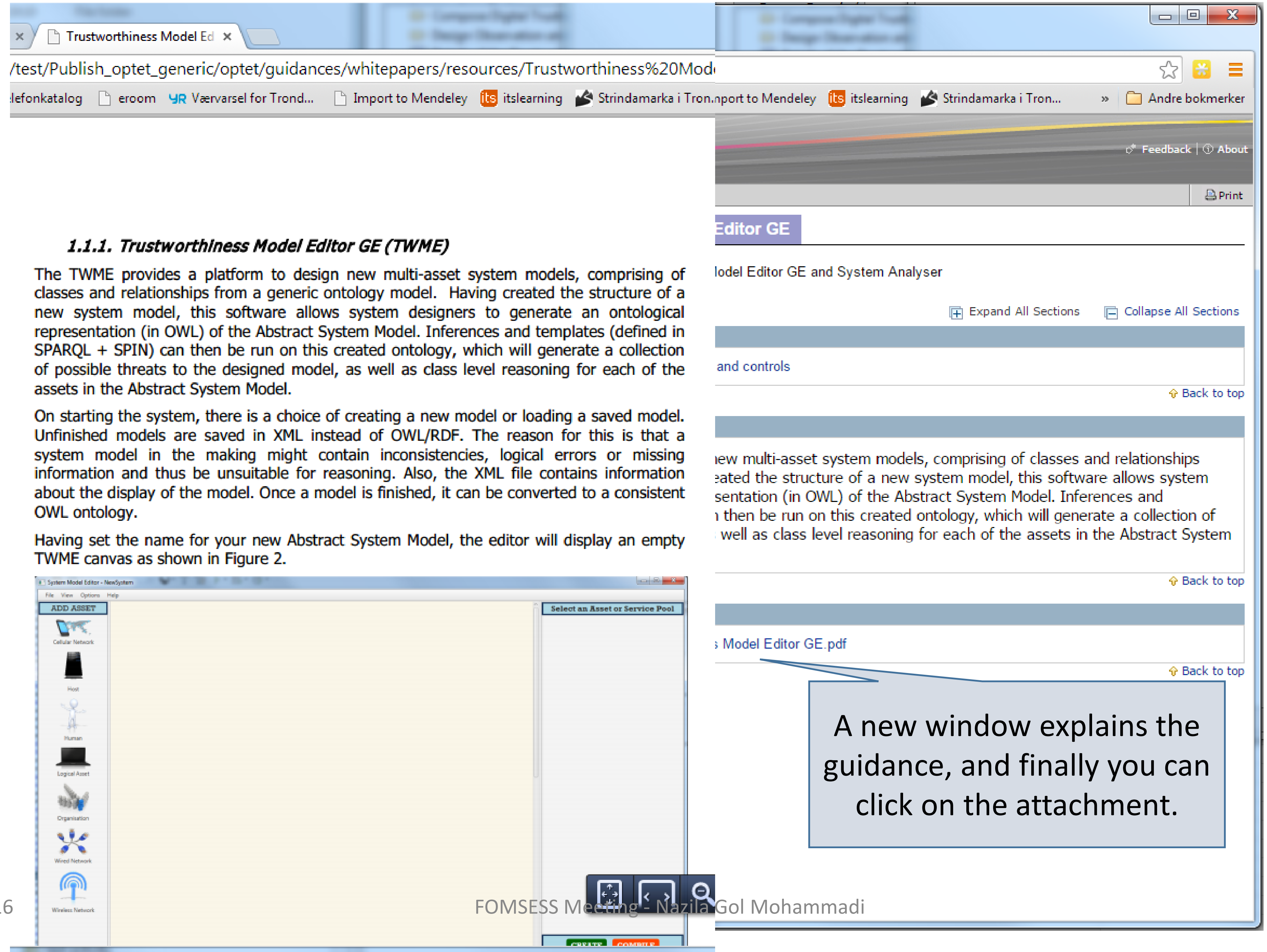
Click here to show all the contents of the "pure" OPTET plugin

Now, you can extend the generic process (or just select the "identify threats and controls" task right away)

Scroll down to the bottom to find the associated guidance document. Click on it.

Extend "Design" and you should find the task "Identify threats and controls". Click on it to show the contents.



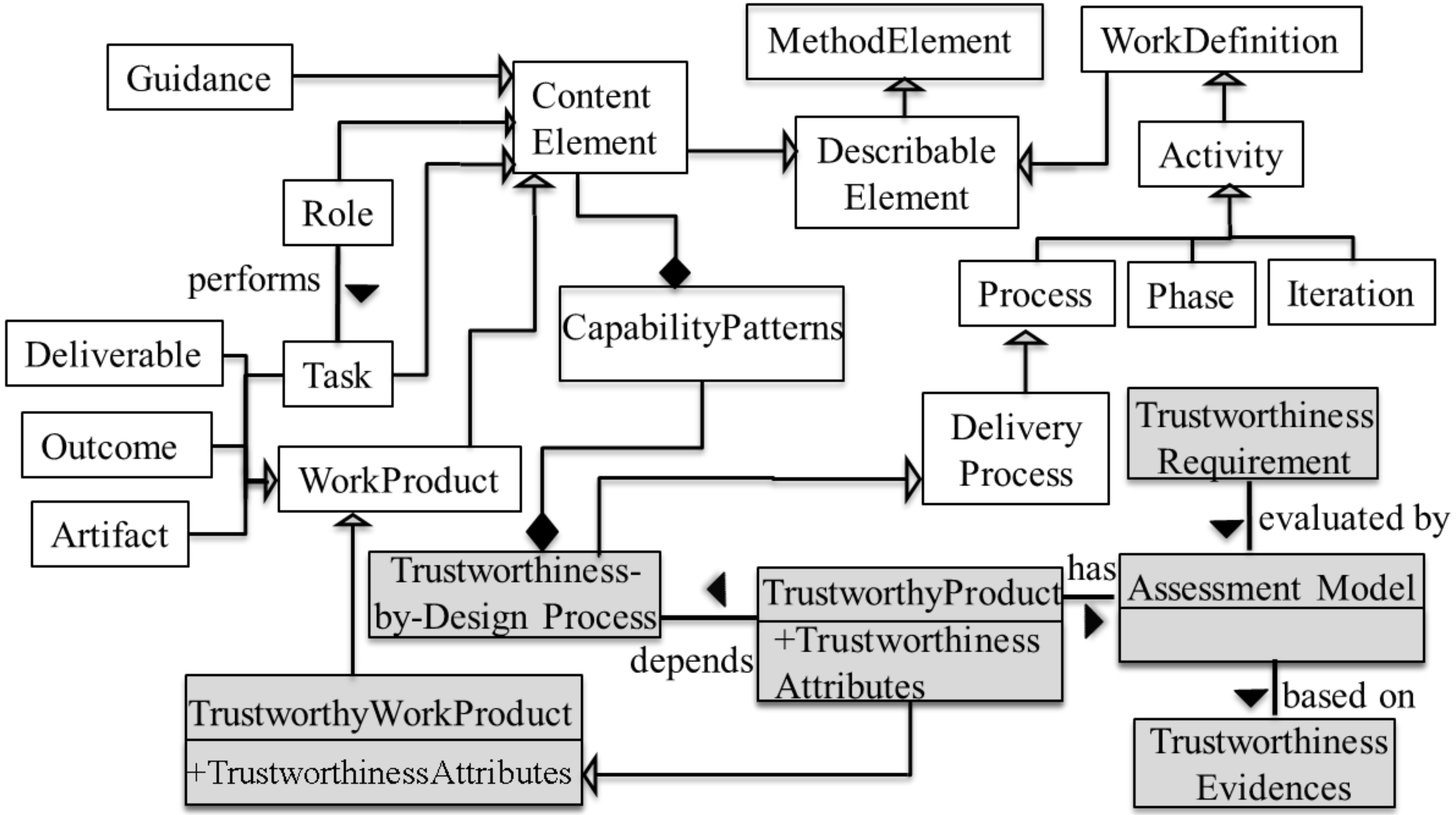


A new window explains the guidance, and finally you can click on the attachment.

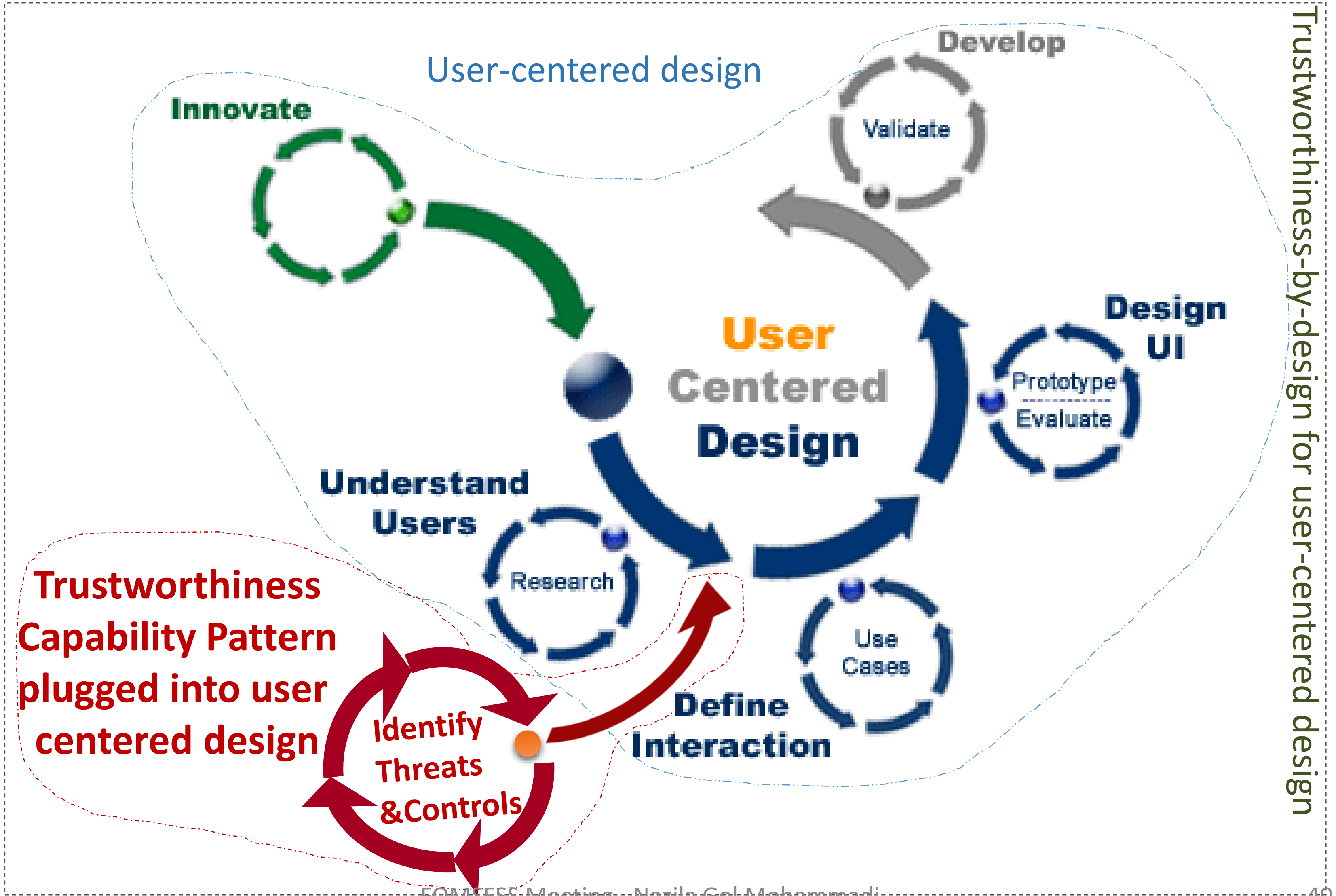
Socio-Economics of Trust

- A generic trust computational model based on system behaviour has been designed, validated and extended
  - Key social & technical factors have been identified resulting in 4 statistically significant trustor segments
  - Each segment reacts differently on system behaviour (successes/failures)
  - Extension: Trust levels are computed for each individual user after a "training period"
- The trust computational model can be used for helping a provider at design-time in finding the combination of system trustworthiness level and price that maximises her profits

Meta-Model for Trustworthiness-by-Design based on SPEM



Example: TWbyD + UCD



Extending Development Methodologies with Trustworthiness-By-Design for Socio-Technical Systems

Introduction and Problem Statement

- Socio-Technical Systems (STS) include humans, organizations, and information systems they use to achieve certain goals [1].
- Trustworthiness in general: assurance that the system will perform as expected or meets certain requirements (cf., e.g., [2]), is an important factor for system adoption.
- Trustworthiness must be considered a multitude of quality attributes, instead of merely focusing on, e.g., security.
- Development methodologies explicitly lack methods addressing trustworthiness evaluation.
- There is a need for generic and reusable trustworthiness process building blocks that can be added to existing software

Our Approach

Extending development methodologies with Trustworthiness-by-Design with taking three steps:

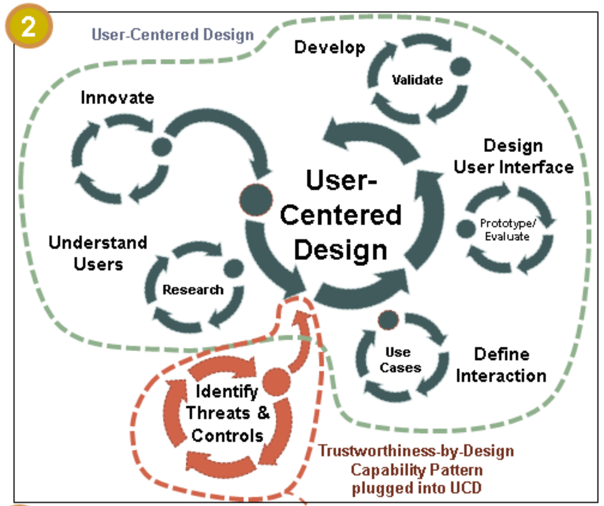
1. We analyze which characteristics of these methodologies are promising to enhance Trustworthiness-by-Design (i.e., to build trustworthy STS), and which indicate improvement potential.
2. For extending the design methodologies, we propose an extension of the Software Process Engineering Meta-model (SPEM) [4], which allows for tailoring certain "trustworthy" process chunks into the process models.
3. We provide Capability Patterns that represent best practices to extend the processes regarding Trustworthiness-by-

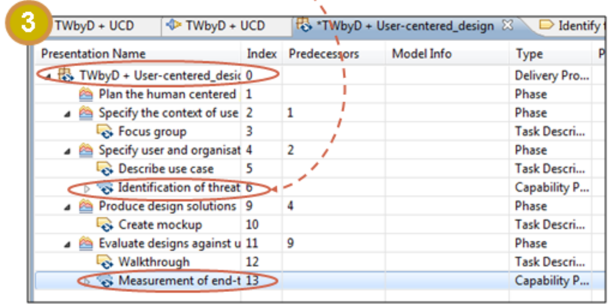
1 Analysis of Development Methodologies → 2 Meta-Model Extensions → 3 Definition of Capability Patterns

Application Example

The analysis results for User-Centered Design (UCD) [3] indicate that it is important to understand which trustworthiness characteristics of the system will enhance stakeholder trust and how system design can help to circumvent any distrust-related concerns. Hence, it is not sufficient to elicit requirements with respect to the way in which people will use the system, as would be done in a standard UCD process. There is an additional need for eliciting and specifying requirements about which trustworthiness attributes will address potential trust issues that might affect the end users. Consequently, we extended UCD by plugging the capability pattern "*Identify Threats and Controls*" into the early process stage of specifying user requirements.

1 Analysis Results for User-Centered Design [3]
Iterative development process, focusing on e.g., user requirements and UI prototyping
Elements interesting for Trustworthiness-by-Design: • is a specialization of incremental development and therefore shares the same trustworthiness characteristics • allows for validating the design throughout the process in order to establish whether the trustworthiness attributes designed into the system appropriately address trust concerns
Improvement Potential: • mismatch between organizational procedures/policies and a more informal or agile process • potentially insufficient documentation (especially concerning non-functional aspects), increasing complexity may cause security flaws • difficult to test and evaluate trustworthiness specifically by the user, who may therefore lose focus on the development





Presentation Name	Index	Predecessors	Model Info	Type
TWbyD + User-centered_desic	0			Delivery Pro...
Plan the human centered	1			Phase
Specify the context of use	2	1		Phase
Focus group	3			Task Descri...
Specify user and organisat	4	2		Phase
Describe use case	5			Task Descri...
Identification of threat	6			Capability P...
Produce design solutions	9	4		Phase
Create mockup	10			Task Descri...
Evaluate designs against u	11	9		Phase
Walkthrough	12			Task Descri...
Measurement of end-t	13			Capability P...

References

Trustworthiness Attributes and Metrics

Selected Trustworthiness Attributes

- Response Time
- Accuracy
- Openness
- Availability

Example of Metrics for selected Trustworthiness Attributes

- Average response time of public software interfaces
- Error rate when logging timestamps
- Software precision when logging timestamps
- Ratio of documented public interfaces
- Ratio of functions that use semaphores or mutexes

End-to-End Formula Composition (2)

R: *Average response time of public interfaces*

$$R_{CAS} = \min(R_{Sensor\_1}, R_{Sensor\_2}, R_{Sensor\_3}, R_{e2e})$$

E: *Error rate when logging timestamps*

$$E_{CAS} = 1 - \frac{1-E_{e2e}}{(1-E_{Sensor\_1})*(1-E_{Sensor\_2})*(1-E_{Sensor\_3})}$$

P: *Software precision when logging timestamps*

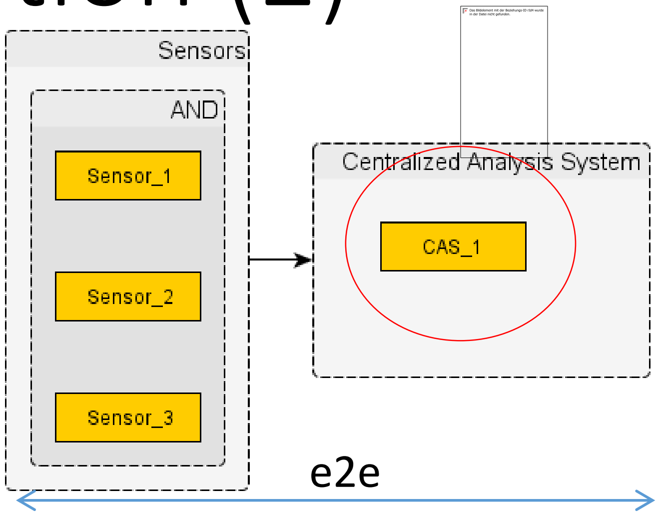
$$P_{CAS} = \min(P_{Sensor\_1}, P_{Sensor\_2}, P_{Sensor\_3}, P_{e2e})$$

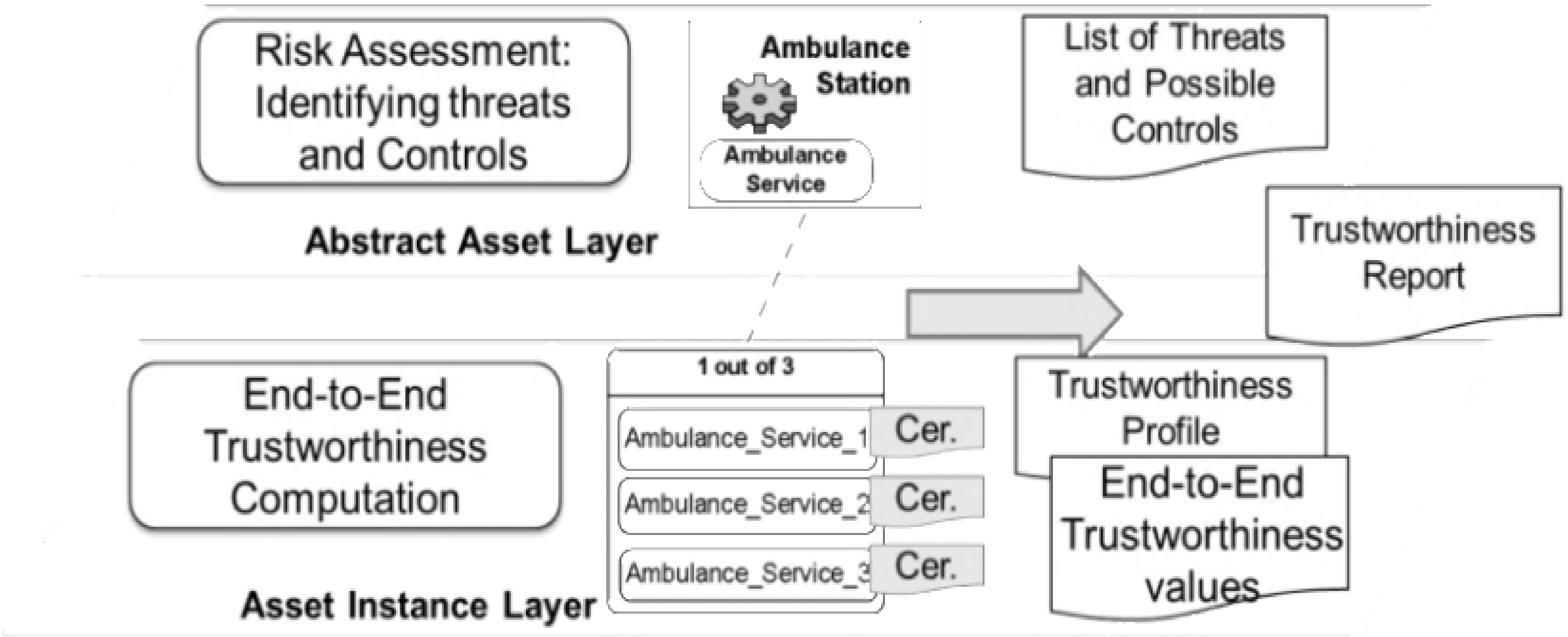
O: *Ratio of documented public interfaces over all interfaces*

$$O_{CAS} = \min(O_{Sensor\_1}, O_{Sensor\_2}, O_{Sensor\_3}, O_{e2e})$$

A: *Ratio of Functions using Semaphors over all functions*

$$A_{CAS} = \frac{A_{e2e}}{A_{Sensor\_1}*A_{Sensor\_2}*A_{Sensor\_3}}$$

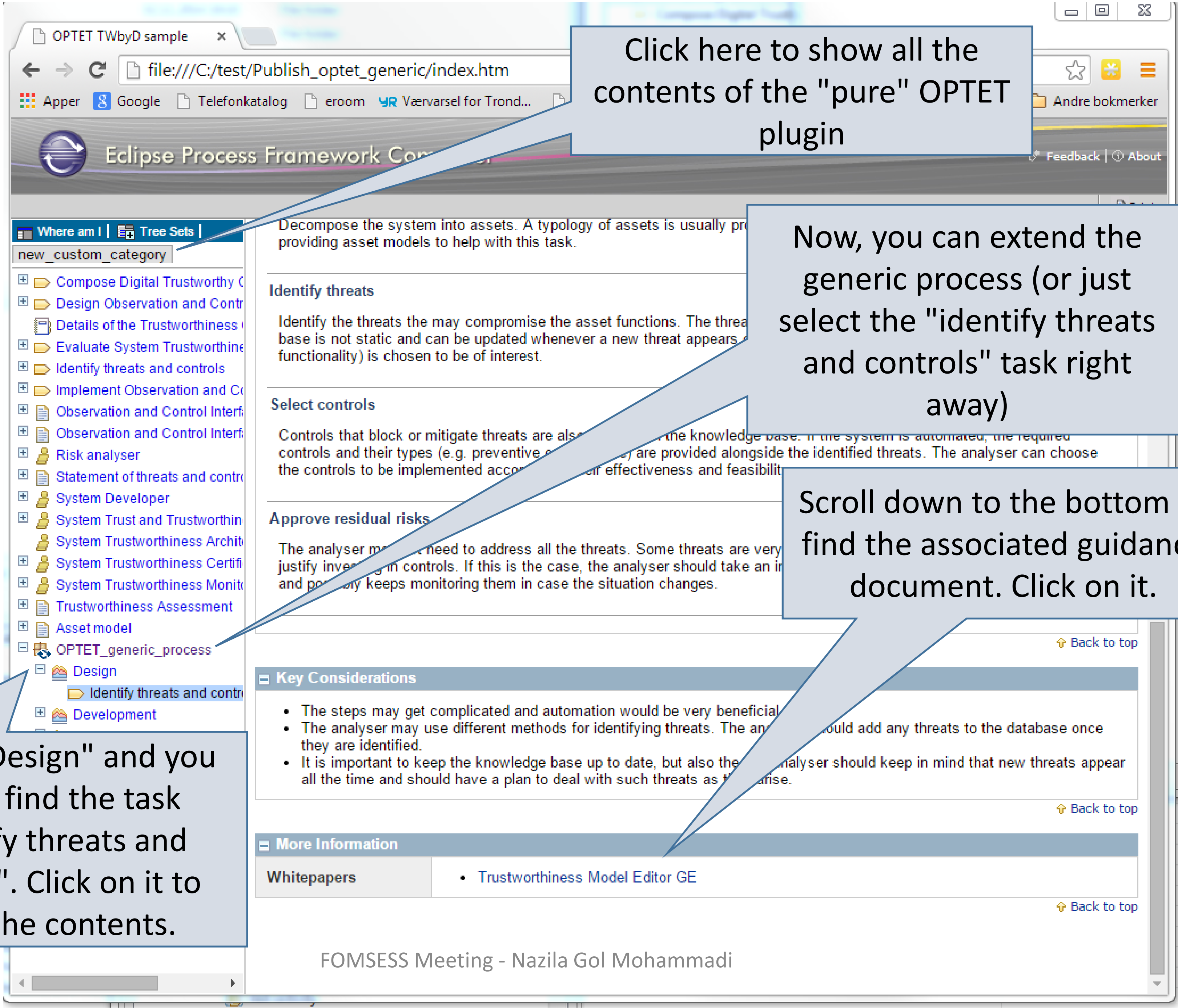




Risk Assessment: Identifying threats and Controls
Ambulance Station
Ambulance Service
List of Threats and Possible Controls
Trustworthiness Report
Abstract Asset Layer
End-to-End Trustworthiness Computation
1 out of 3
Ambulance_Service_1
Cer.
Ambulance_Service_2
Cer.
Ambulance_Service_3
Cer.
Trustworthiness Profile
End-to-End Trustworthiness values
Asset Instance Layer

Trustworthiness Modeling Example

Elements of Capability Patterns	Content in CP: Identification of Threats and Mitigating Controls	Relevant support by developed Tool: Trustworthiness Model Editor
Role	Risk Analyzer	System designer with the knowledge in Asset modeling and risk analysis and management processes
Task	Identify threats and controls [system scoping, modelling, risk assessments, etc.]	Supporting the tasks for creating the asset models, automated identification of threats and a list of possible controls
Work products	Asset model, Statement of threats and controls	Design Time Trustworthiness Model, threat and controls list based on the model

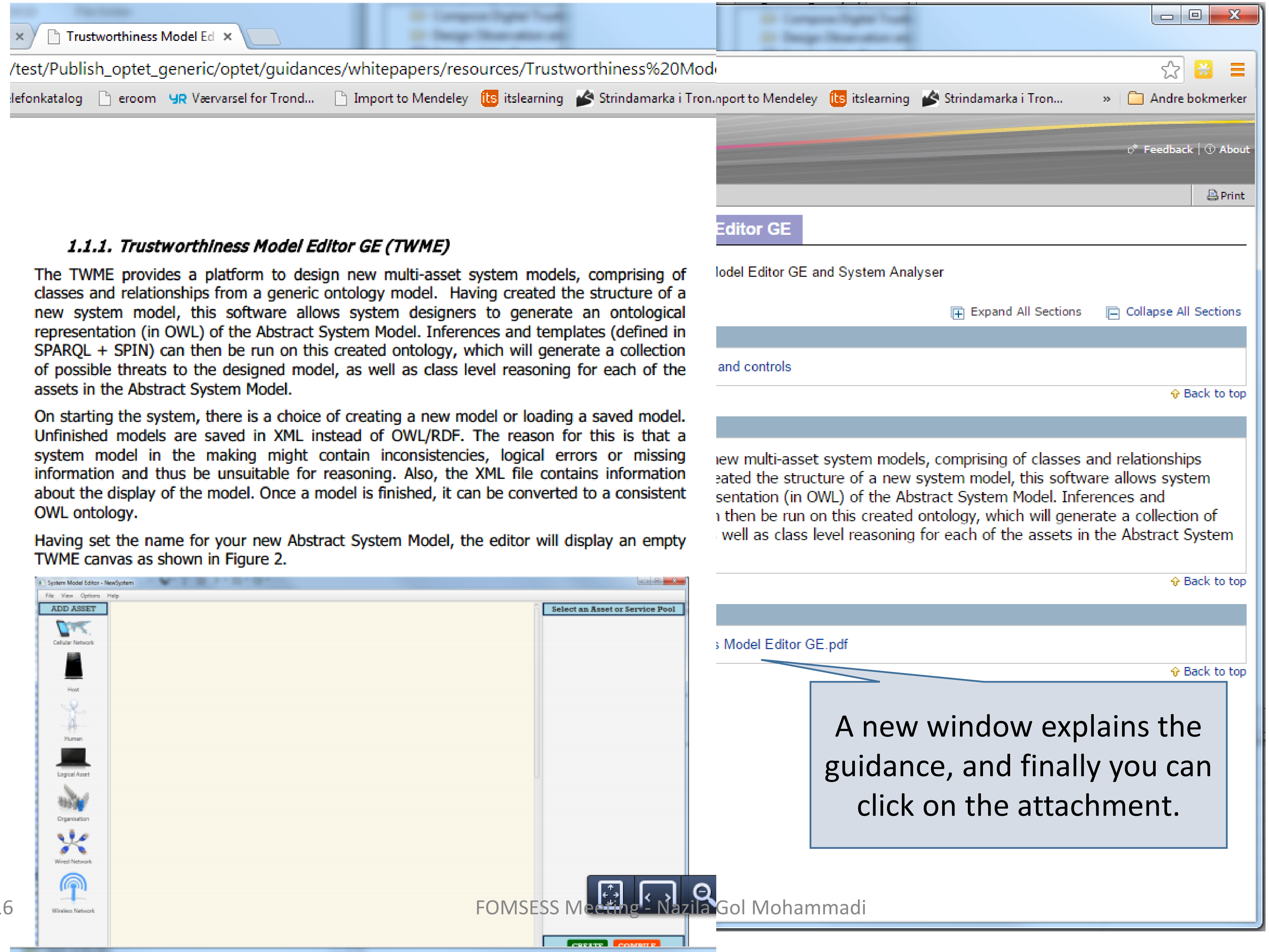


Click here to show all the contents of the "pure" OPTET plugin

Now, you can extend the generic process (or just select the "identify threats and controls" task right away)

Scroll down to the bottom to find the associated guidance document. Click on it.

Extend "Design" and you should find the task "Identify threats and controls". Click on it to show the contents.



1.1.1. Trustworthiness Model Editor GE (TWME)

The TWME provides a platform to design new multi-asset system models, comprising of classes and relationships from a generic ontology model. Having created the structure of a new system model, this software allows system designers to generate an ontological representation (in OWL) of the Abstract System Model. Inferences and templates (defined in SPARQL + SPIN) can then be run on this created ontology, which will generate a collection of possible threats to the designed model, as well as class level reasoning for each of the assets in the Abstract System Model.

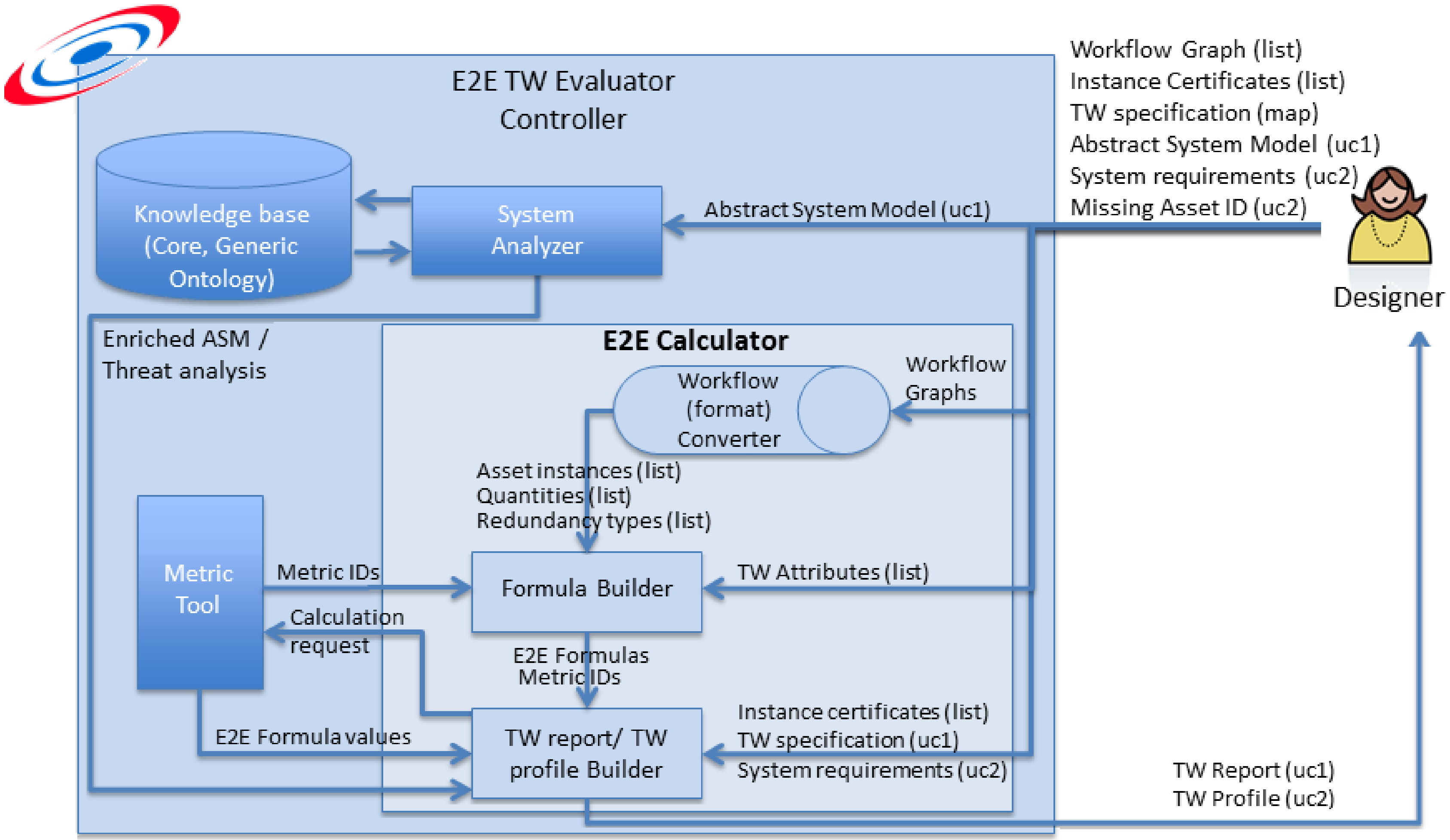
On starting the system, there is a choice of creating a new model or loading a saved model. Unfinished models are saved in XML instead of OWL/RDF. The reason for this is that a system model in the making might contain inconsistencies, logical errors or missing information and thus be unsuitable for reasoning. Also, the XML file contains information about the display of the model. Once a model is finished, it can be converted to a consistent OWL ontology.

Having set the name for your new Abstract System Model, the editor will display an empty TWME canvas as shown in Figure 2.

A new window explains the guidance, and finally you can click on the attachment.

Socio-Economics of Trust

- A generic trust computational model based on system behaviour has been designed, validated and extended
  - Key social & technical factors have been identified resulting in 4 statistically significant trustor segments
  - Each segment reacts differently on system behaviour (successes/failures)
  - Extension: Trust levels are computed for each individual user after a "training period"
- The trust computational model can be used for helping a provider at design-time in finding the combination of system trustworthiness level and price that maximises her profits



* Each Workflow Graph is a yEd generated graph
Graph contains nodes (instance names) and groups for specifying quantities and redundancy type