Affordable, fact-oriented assurance

with OMG standards

Nikolai Mansourov, Djenana Campara
KDM Analytics, Inc.

http://www.kdmanalytics.com

06-12-2010 Austin, TX

Afffordable assurance ?

- Current approaches are too costly, so only few ogranizations can afford them. However there is a lot more organization and even individuals how make decisions to use cyber systems for their operations, each with own definition of what is safety-critical, security-critical or mission-critical. At the current cost of assurance their can not afford it, which means that they accept risks that are unknown to them and that may be too high for them.

- Affordable solutions must be scalable

  - There are two kinds of scalability: technical scalability and human scalability. The later is invovles a systematic and repeatable approach to assurance. The former involves automation.

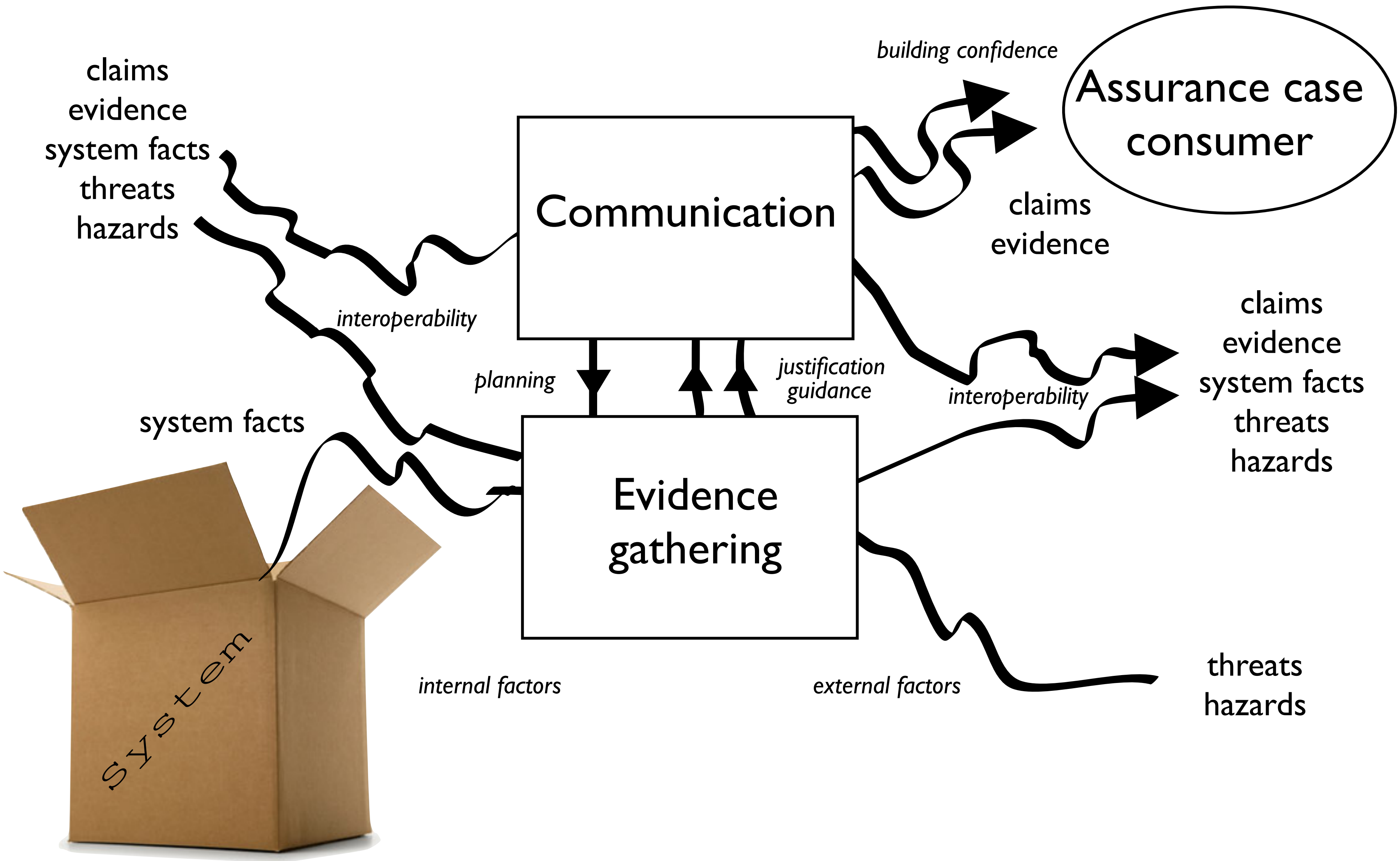
  - Both kinds of scalability can only be achieved through standards. Standards are known to enable economies of scale based on the division of labour.

- So, we must look at the assurance process and identify the opportunities for cooperation, based on exchanges and interoperability.

What is system assurance?

- System performs a mission within a certain operational environment
- There are hazards and threats within the environment that can lead to mishaps and failures
- In order to prevent mishaps and failures, countermeasures are added to the system
- But how do we know that the countermeasures are effective against the known threats and hazards?
- System assurance is about making *justified claims* about the effectiveness of the countermeasures against threats and hazards. Claims are supported by evidence.

System assurance: knowledge-intensive product



Knowledge exchanges in system assurance

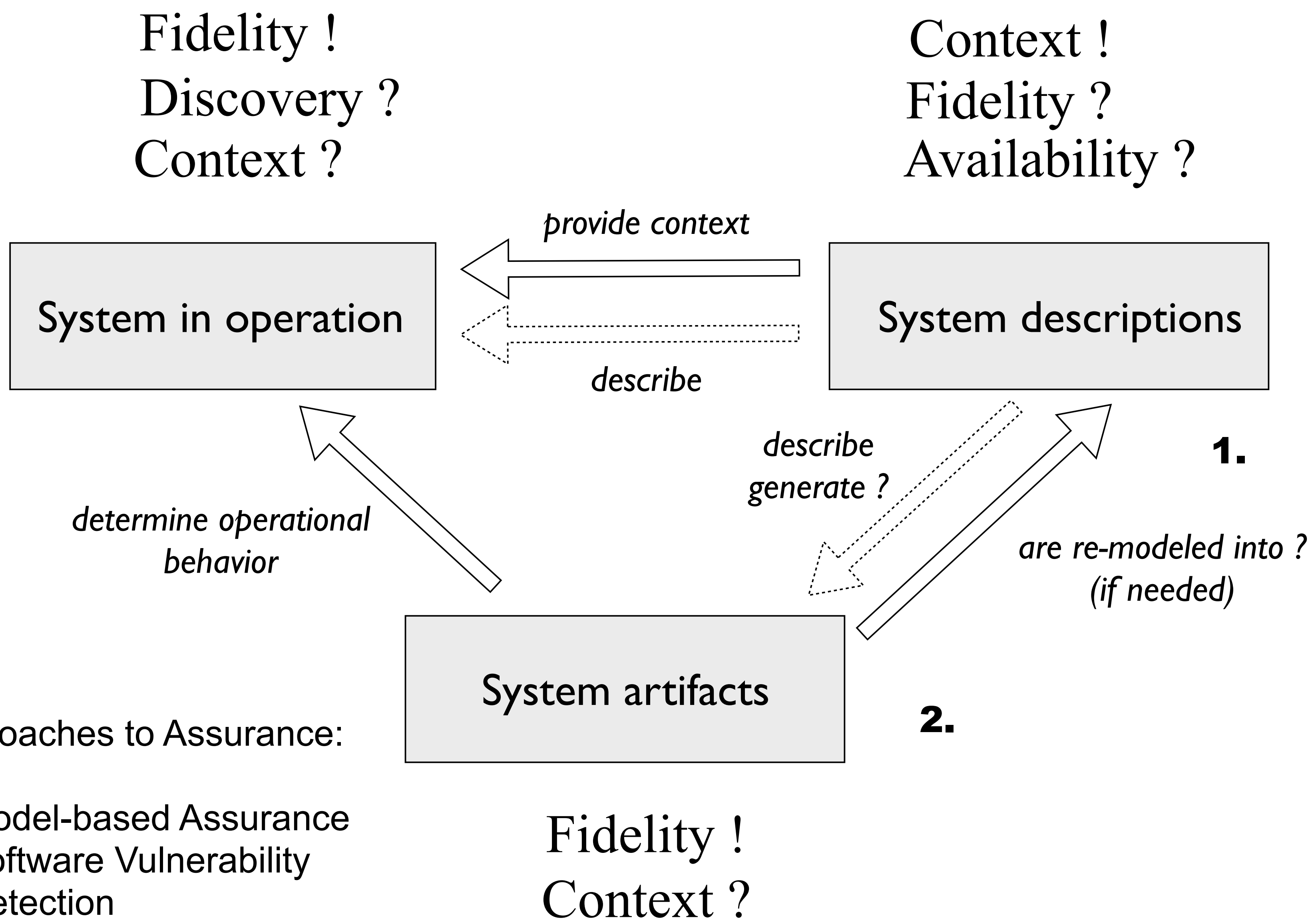
- System assurance involves two key processes
  - evidence gathering
    - collection of the evidence from the system life cycle
    - system analysis
    - analysis of evidence
  - communication
    - clear, comprehensive, defendable argument that explains the evidence
    - development of the assurance case is driven by existing evidence
    - assurance argument provides guidance for evidence collection

Fact-oriented assurance

- Fact-oriented involves the following:
  - Facts are *assertions* that are considered to be elementary to be understood and agreed upon without the need for further justification. Facts involve assertions of *existence* of certain objects, *characteristics* of objects and assertions of certain *relations* between these objects.
  - Evidence is the collection of relevant facts. Evidence needs to be gathered among the miriads of facts that can be known.
  - Fact-oriented assurance develops claims based on the available facts. On the other hand, the assurance argument helps planning the evidence gathering, which helps focus on only those fact-finding activities that support the assurance argument
  - Fact-oriented also has a certain technical meaning: all knowledge items are uniformly treated as facts (objects and relationships), which facilitates their integration. Facts are stored in a physical repository

What are the facts?

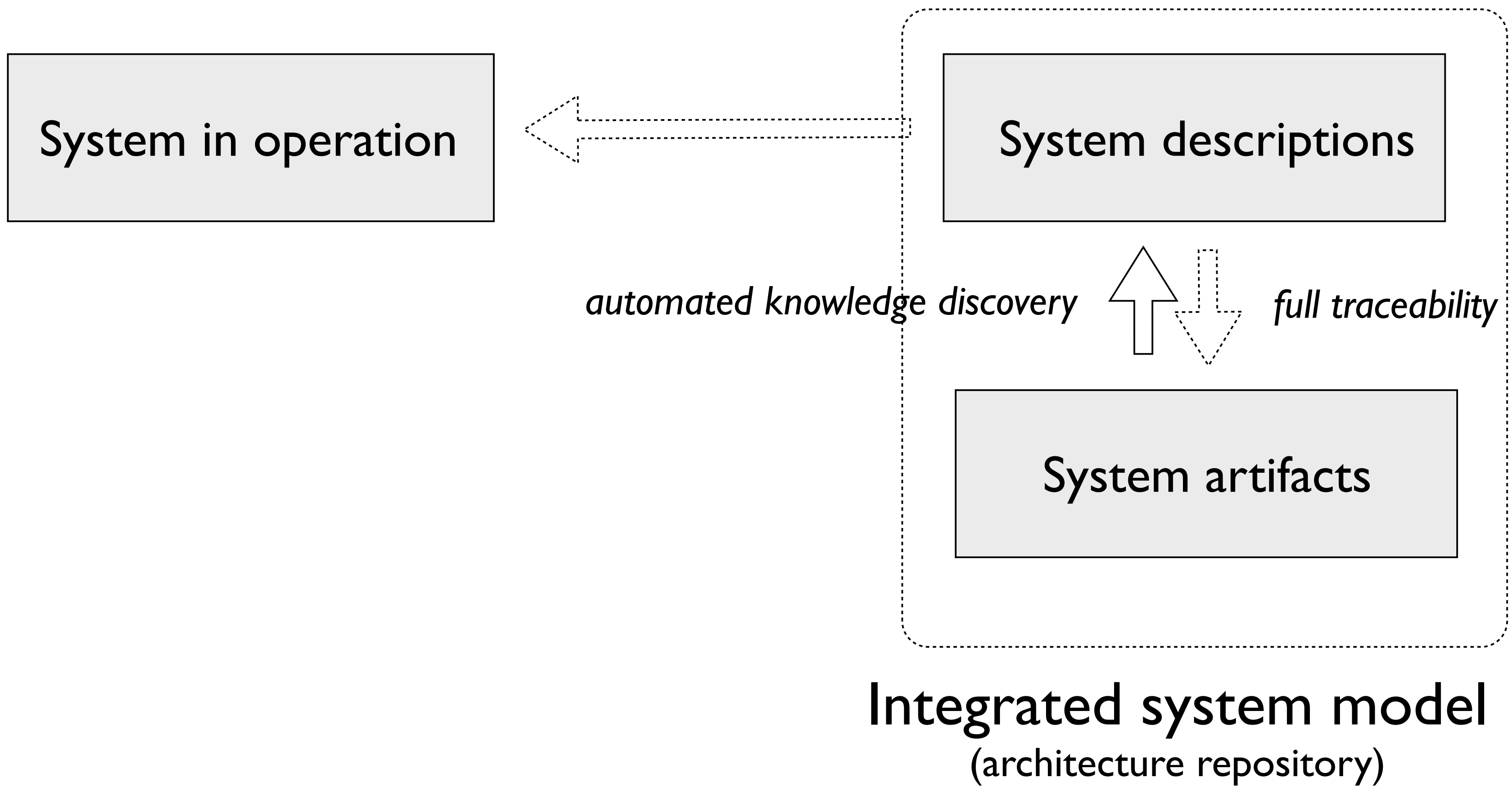
- System in operation involves event occurences. Operational facts usually involve snapshots of behaviors
    - Assurance is focused at the operational facts, as mishaps and incidents are operational events
- System artifacts determine the event occurences during the operations. For cyber systems the majority of the artifacts involve code. Artifacts of a mechanical system may involve pipes, valves, gauges, etc. Artifacts of systems involving human actors are rule books, etc.
- There are also various system descriptions, including blueprints, models, etc. System descriptions involve multiple viewpoints of the system of interest.



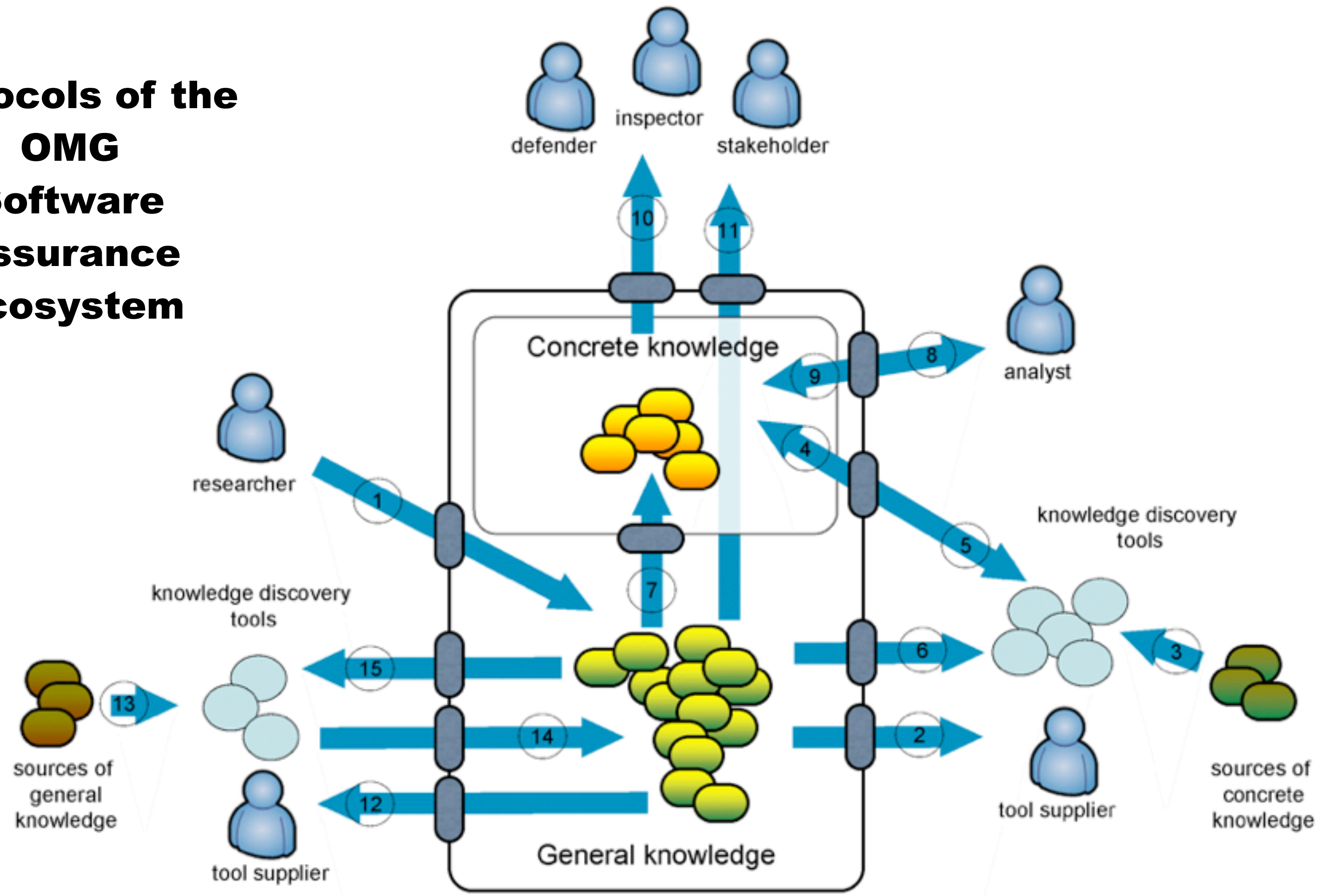
Approaches to Assurance:

1. Model-based Assurance
2. Software Vulnerability Detection
3. Fact-oriented Assurance

Fact-Oriented Assurance



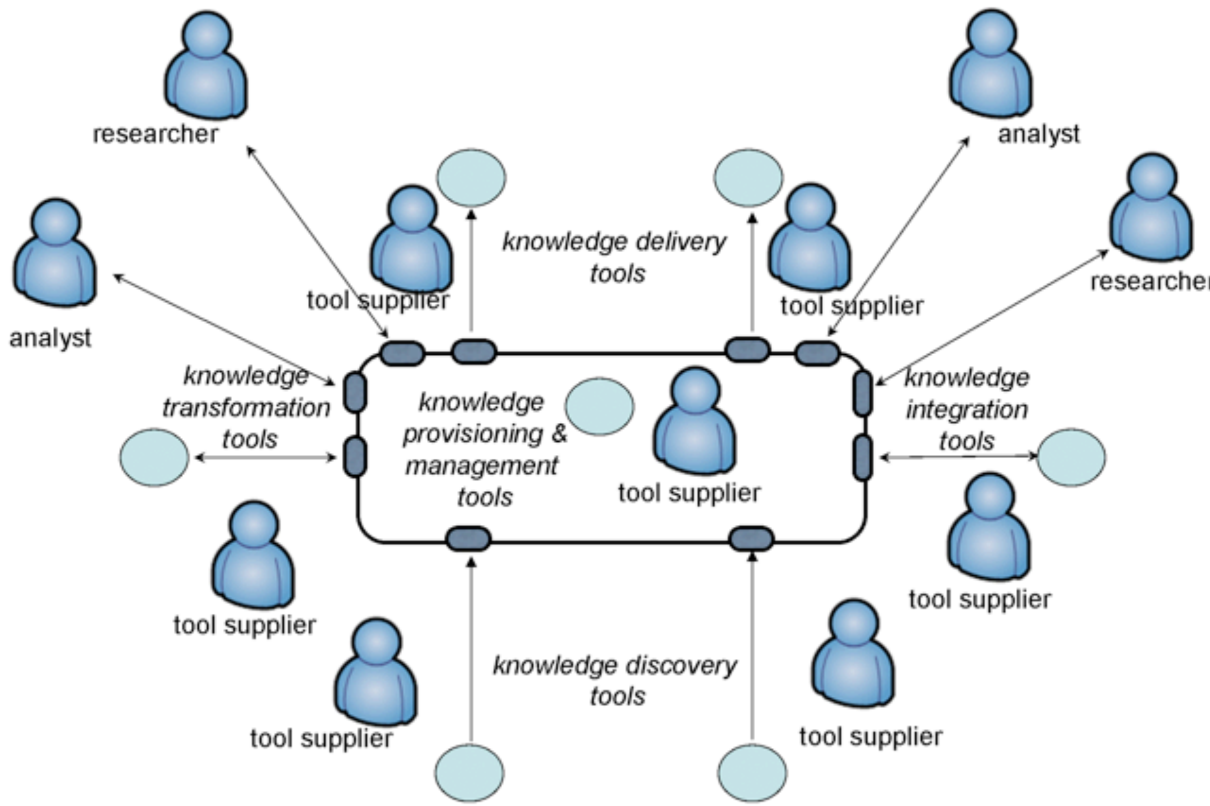
Protocols of the OMG Software Assurance Ecosystem



Protocols of the OMG Software Assurance Ecosystem

- *Argumentation Metamodel (ARM)*: standard protocol for exchanging assurance arguments
- *Software Assurance Evidence Metamodel (SAEM)*: standard protocol for managing and exchanging evidence
- *Knowledge Discovery Metamodel (KDM)*: standard protocol for exchanging system facts
  - *Now also ISO/IEC 19506*
- *Semantics of Business Vocabularies and Rules (SBVR)*: standard protocol for exchanging vocabularies and precise statements
- *Threats and Risk Metamodel*
  - work in progress

Protocols of the
OMG
Software Assurance
Ecosystem
enable
exchange of
machine-readable
content
and automation



Requirement:

FPR_UNO1.1 **Unobservability**: The system shall ensure that any users/subjects are unable to observe any operation on any object/resource by any other user/subject.

Noun concepts:

- System
- User/subject
- Object/resource
- Operation

Verb concepts:

- System *involves* object/resource
- System *involves* operation
- User/subject *performs* operation *on* object/resource
- User/subject *observes* operation

information *flows from* $user_1$ *to* $user_2$

Sample Facts:

```
system('clicks2bricks').
involves_resource('clicks2bricks','personal information of Bill').
involves_resource('clicks2bricks','help page 127').
involves_operation('clicks2bricks','employee request').
involves_operation('clicks2bricks','open page request').
user('Joe'). user('Frank').
performs('op001','Joe','employee request',personal information of Bill').
performs('op002','Frank','open page request','help page 127').
observes('Frank','op002','op001').
```

Sample Verbalization: *System clicks2bricks involves personal information of Bill*

Claim is formalized but there is a semantic gap to the software artifacts

FPR_UNO1.1 **Unobservability**: The system shall ensure that any users/subjects are unable to observe any operation on any object/resource by any other user/subject.

System

User/subject

Object/resource

Operation

System *involves* object/resource

System *involves* operation

User/subject *performs* operation *on* object/resource

User/subject *observes* operation

information *flows from* user$_1$ *to* user$_2$

Second tied concepts close the gap to software artifacts:

Partition

Activity

Information item

System *has* partition

User/subject *is associated with* partition

partition *has* activity

activity *performs* operation *on* object/resource

activity *follows* activity

activity *writes to* information item

Information item *is a record of* operation

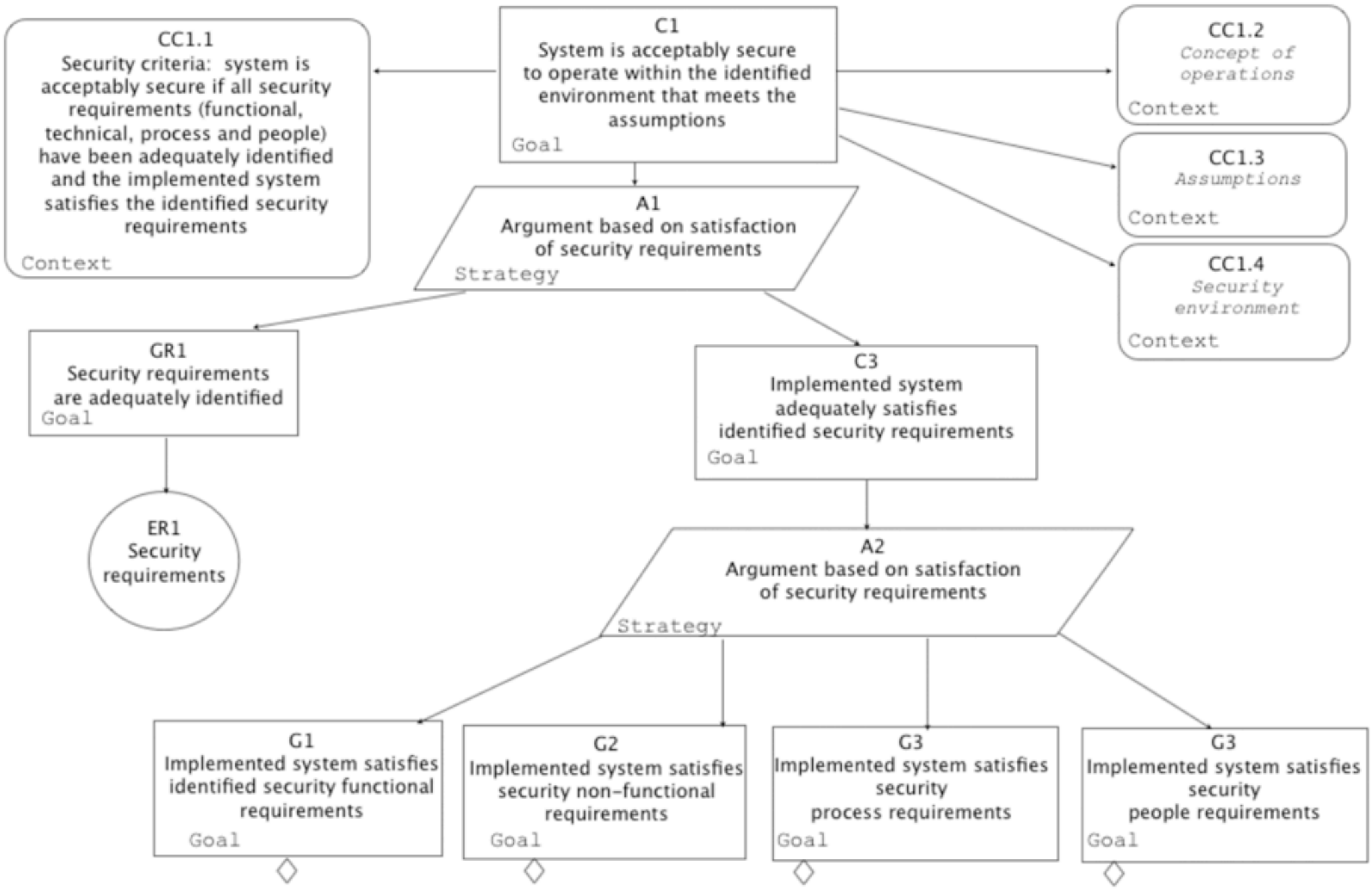
information item *is observable by* partition

activity *discloses* information *to* partition

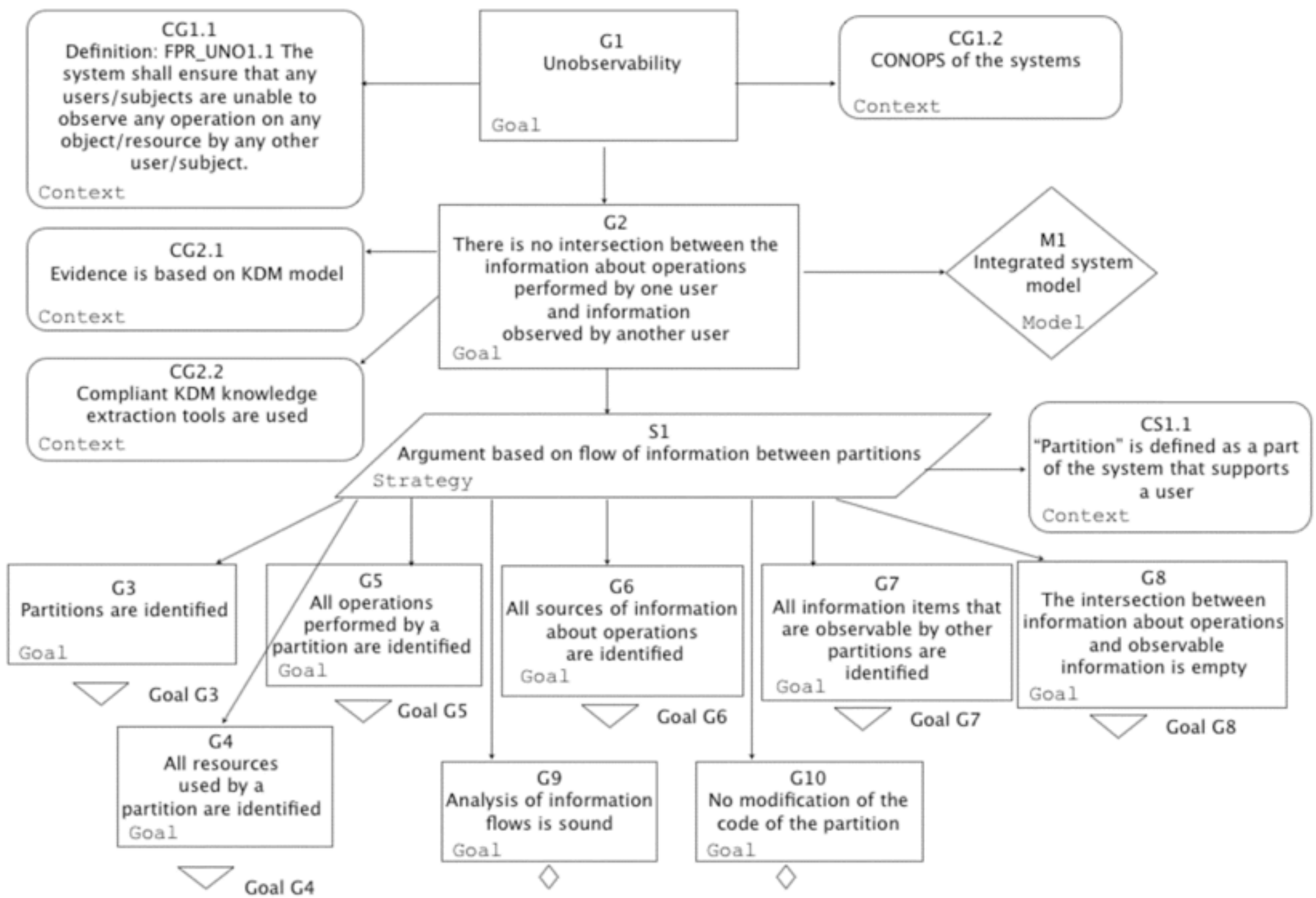
information item *flows from* partition$_1$ *to* partition$_2$

Common vocabulary is a contract; the key to vocabulary refinement is to have a standard vocabulary of system facts

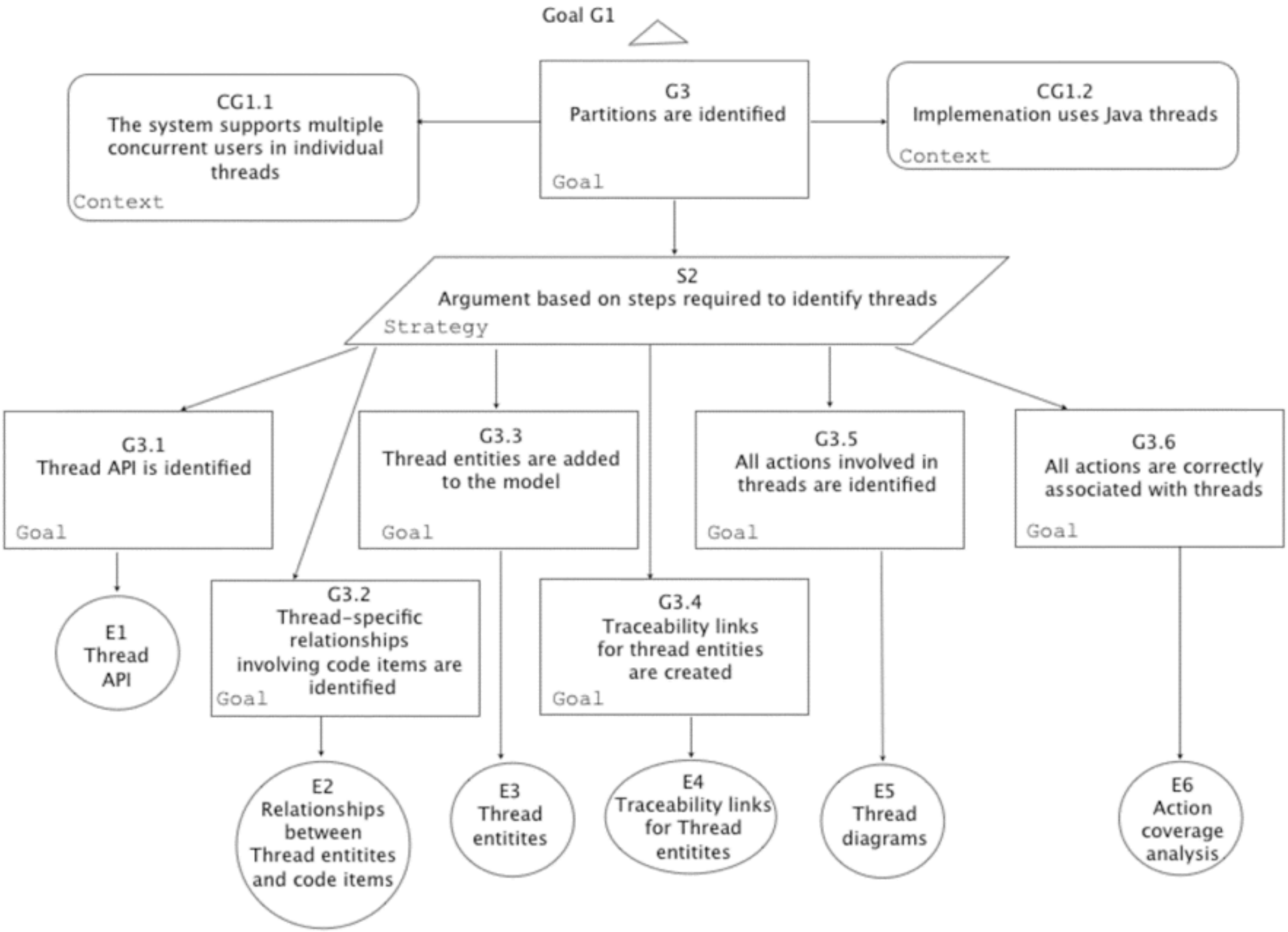
Top level Assurance Case



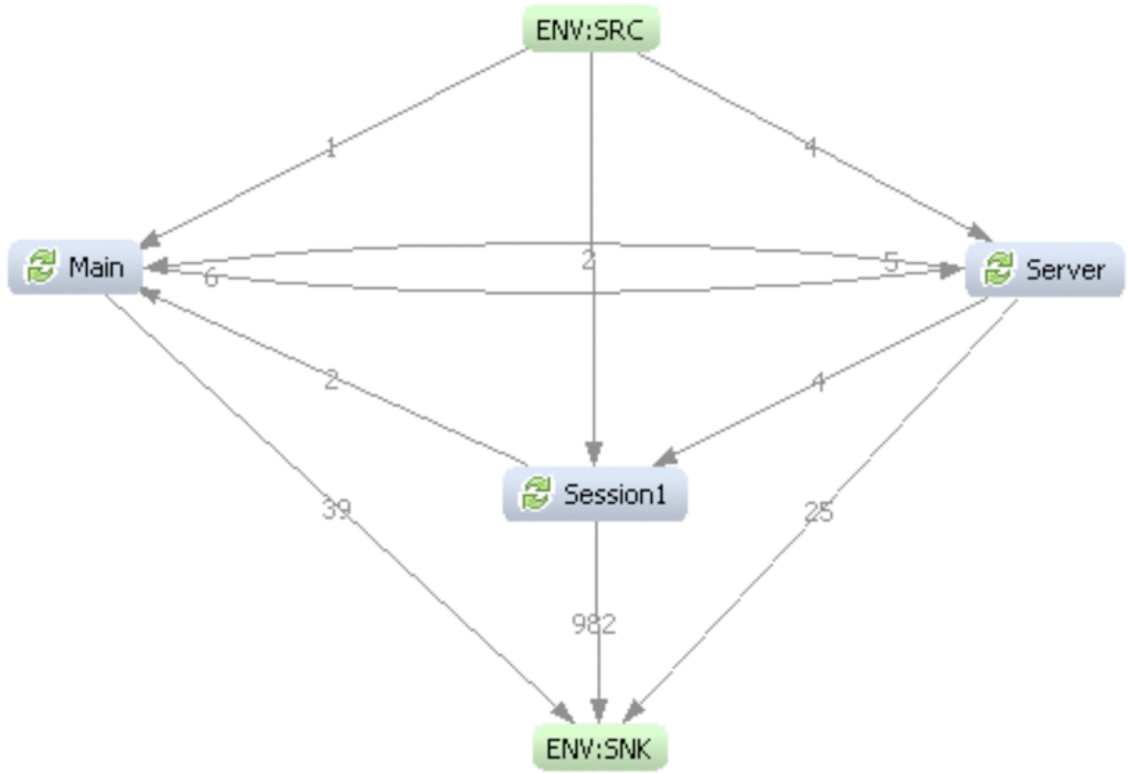
Assurance Case for Unobservability



Decomposition of Claims bridges the gap to available facts

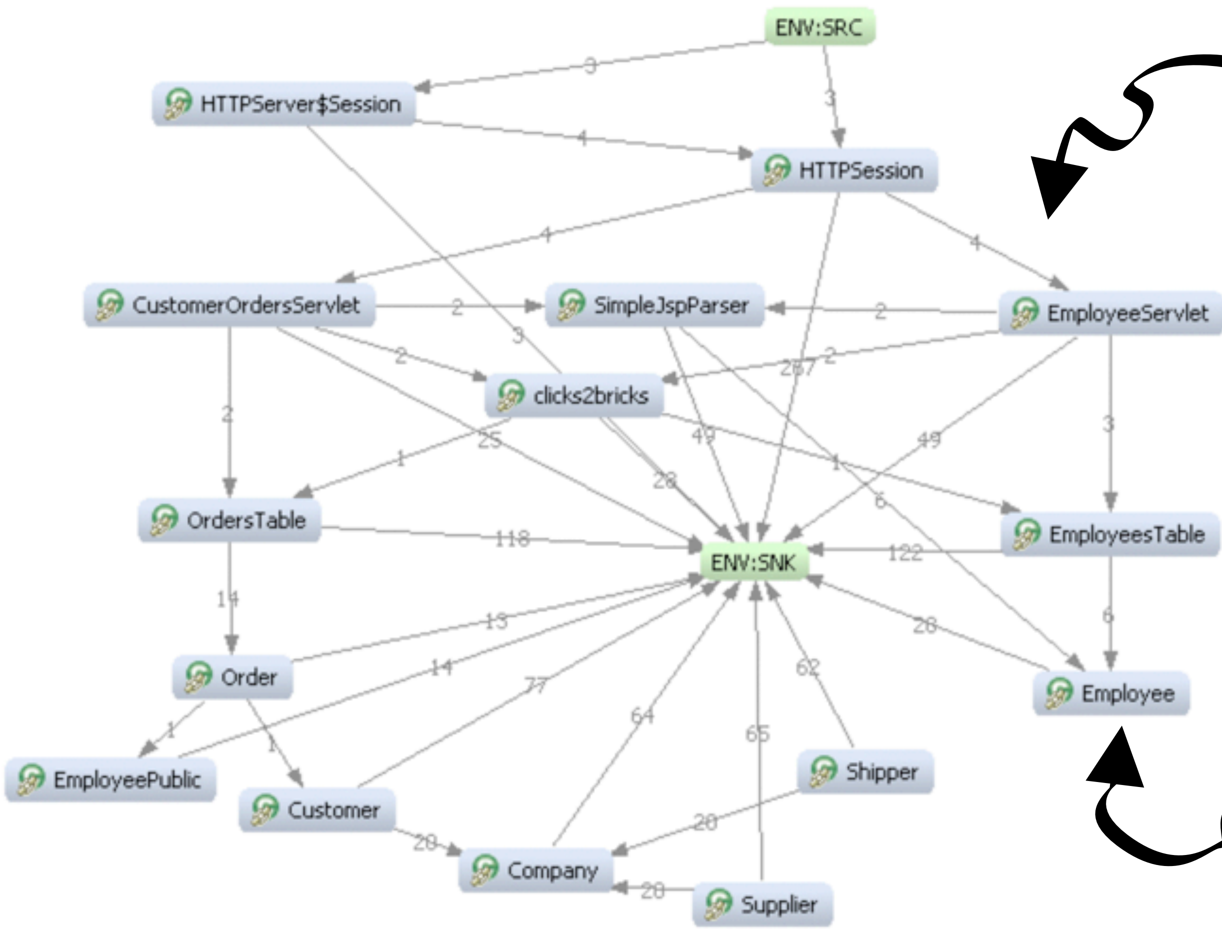


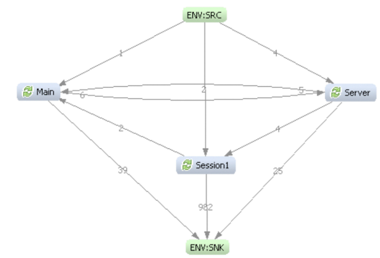
Thread entities (KDM view)



Since KDM is a standard, KDM facts of the system of interest can be discovered independently of the Unobservability claims. The standard-based KDM fact repository can be reused for different assurance claims as well as other maintenance and evolution activities

KDM views provide traceability down to code



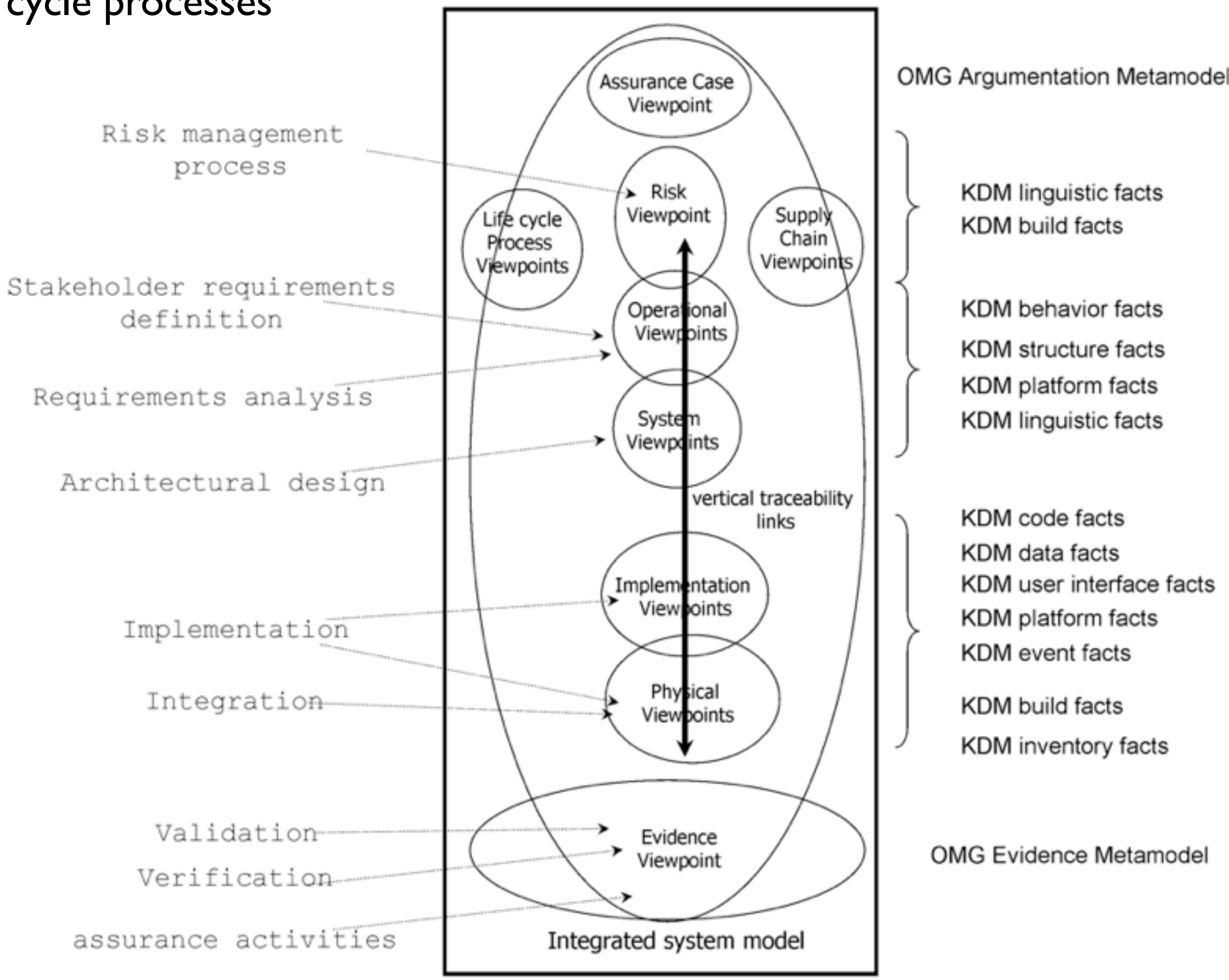


Traceability chain

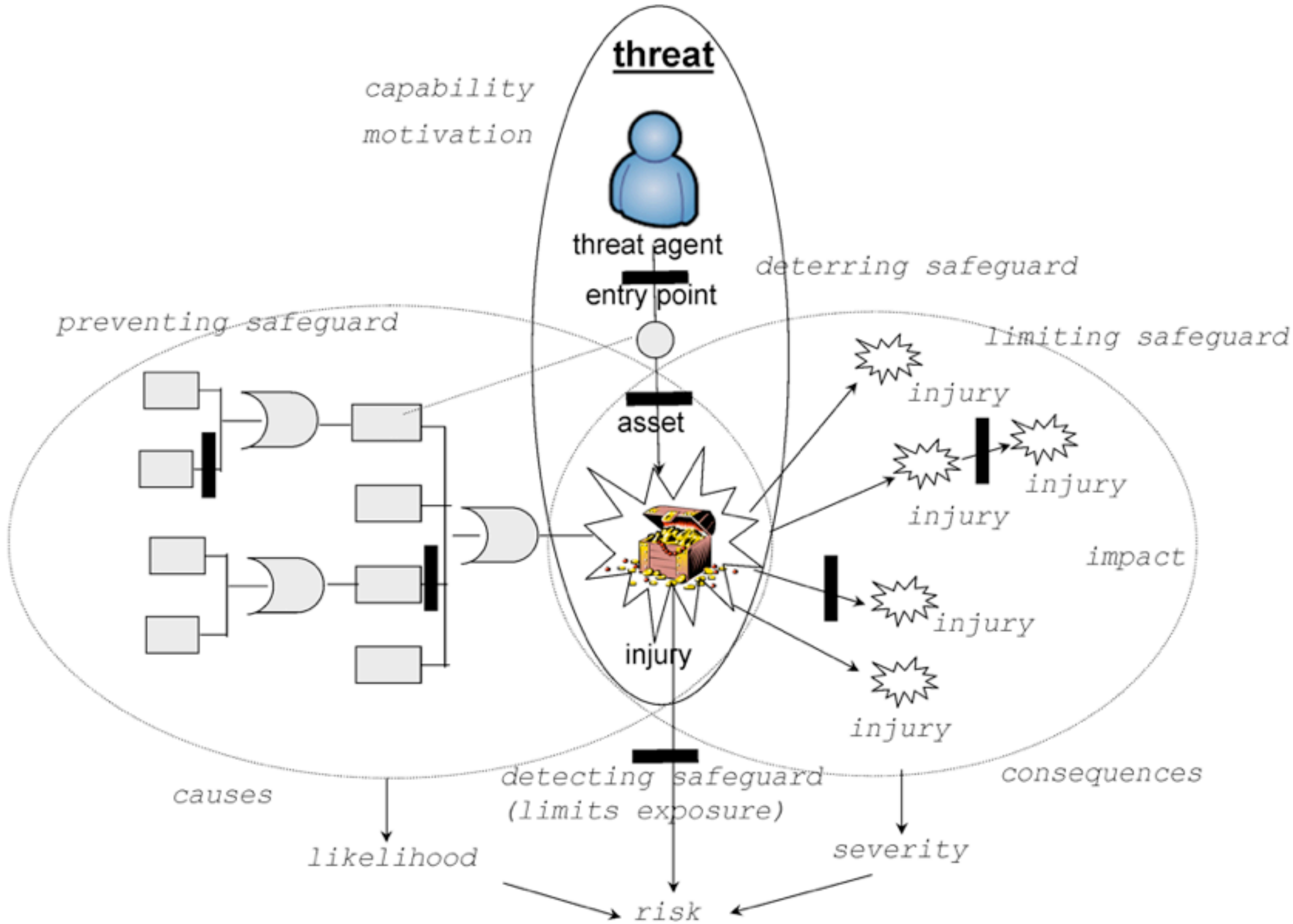
Elements of Management
Coverage
Closed world

KDM views and Assurance

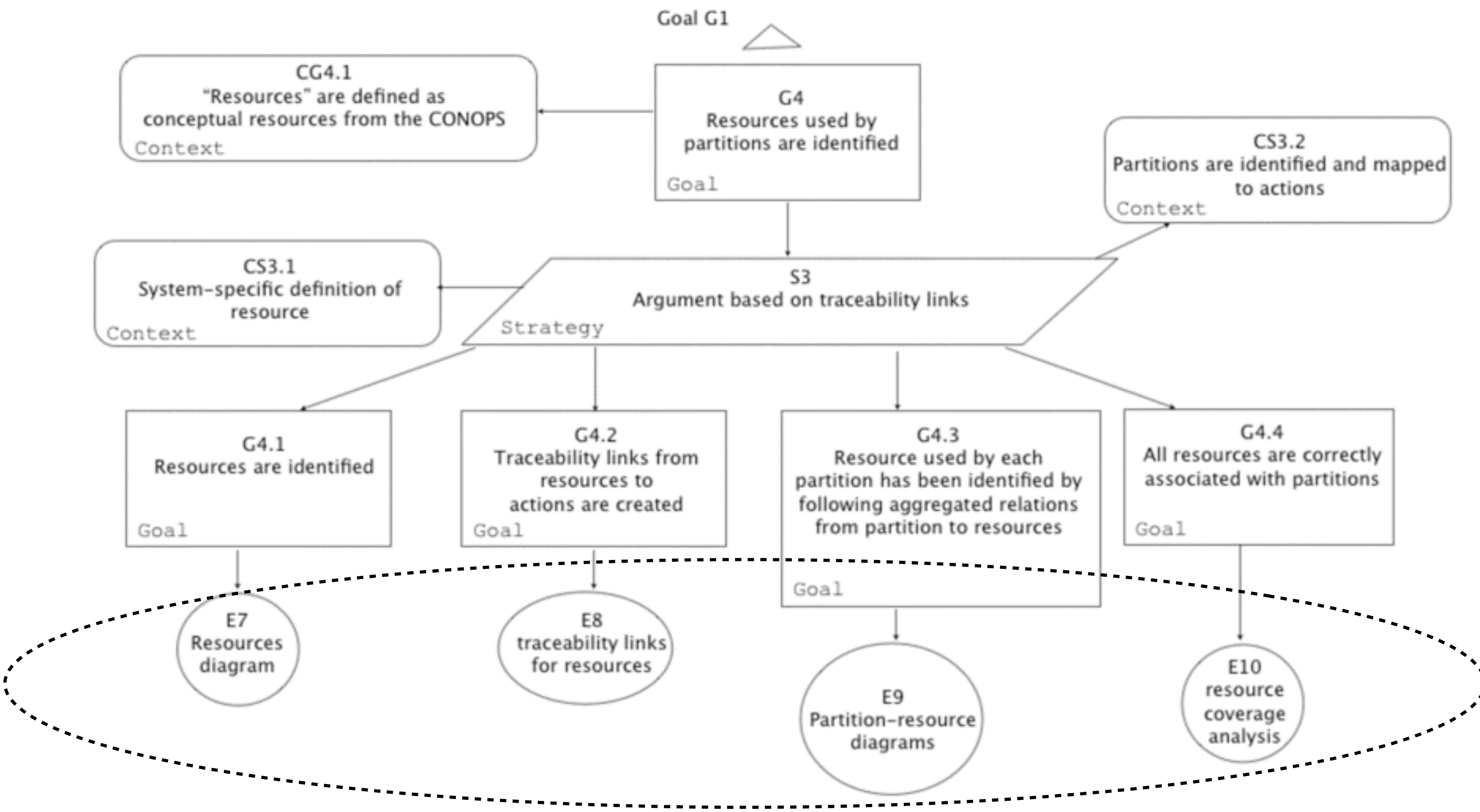
System life cycle processes



Assurance Case supports Risk Management

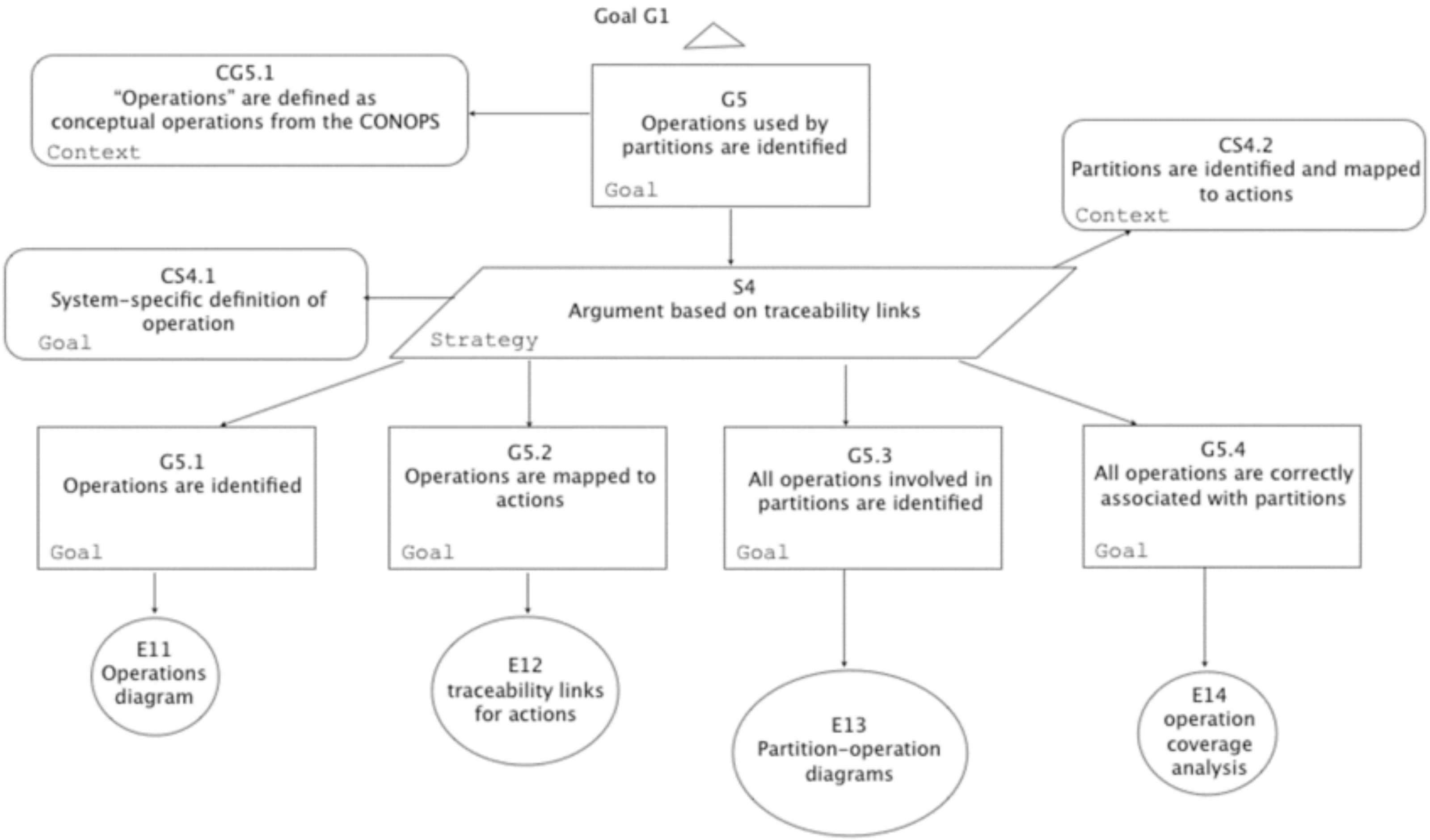


Unobservability Assurance Case (cont'd)

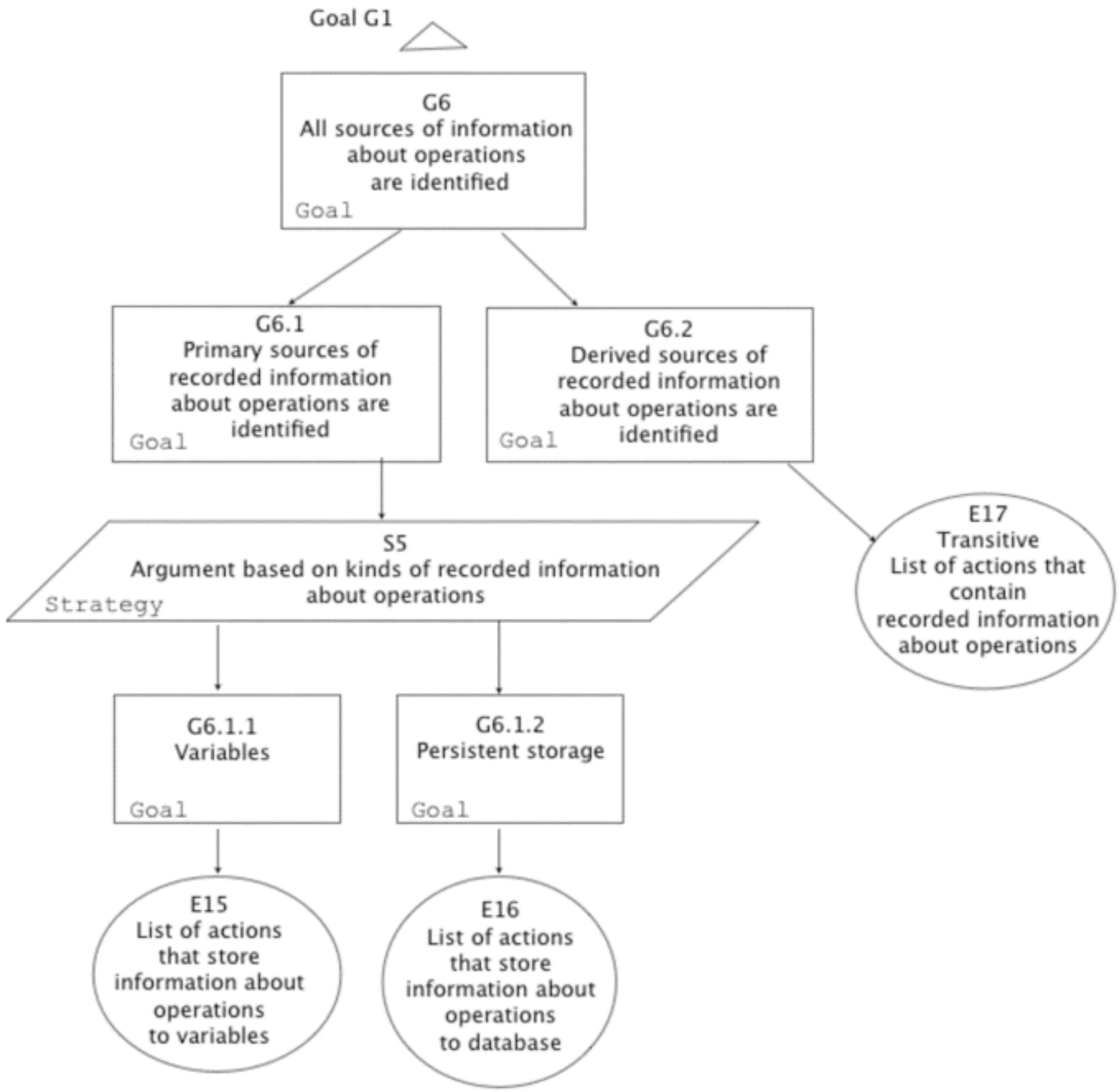


Facts available in the KDM repository (directly or indirectly)

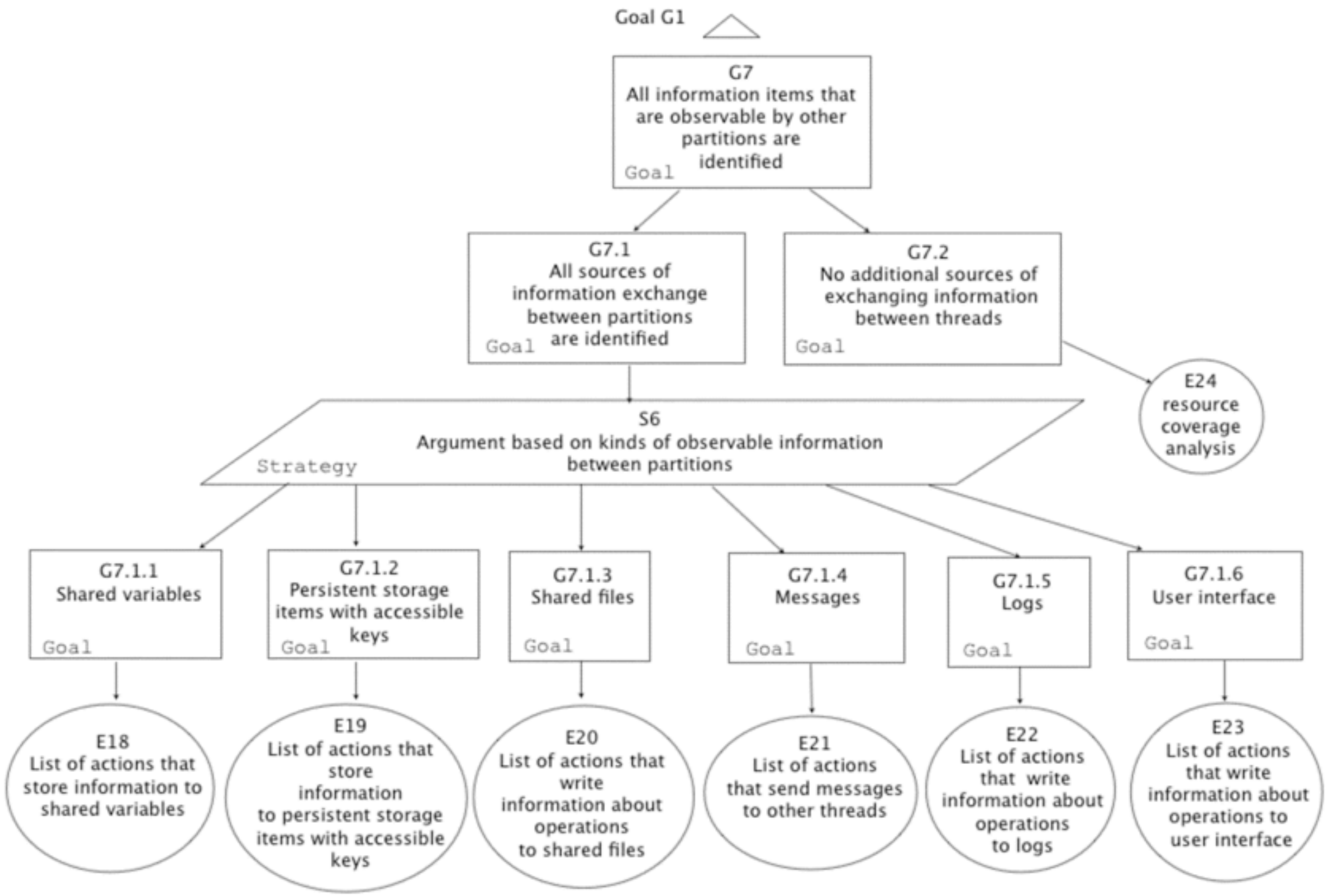
Unobservability Assurance Case (cont'd)



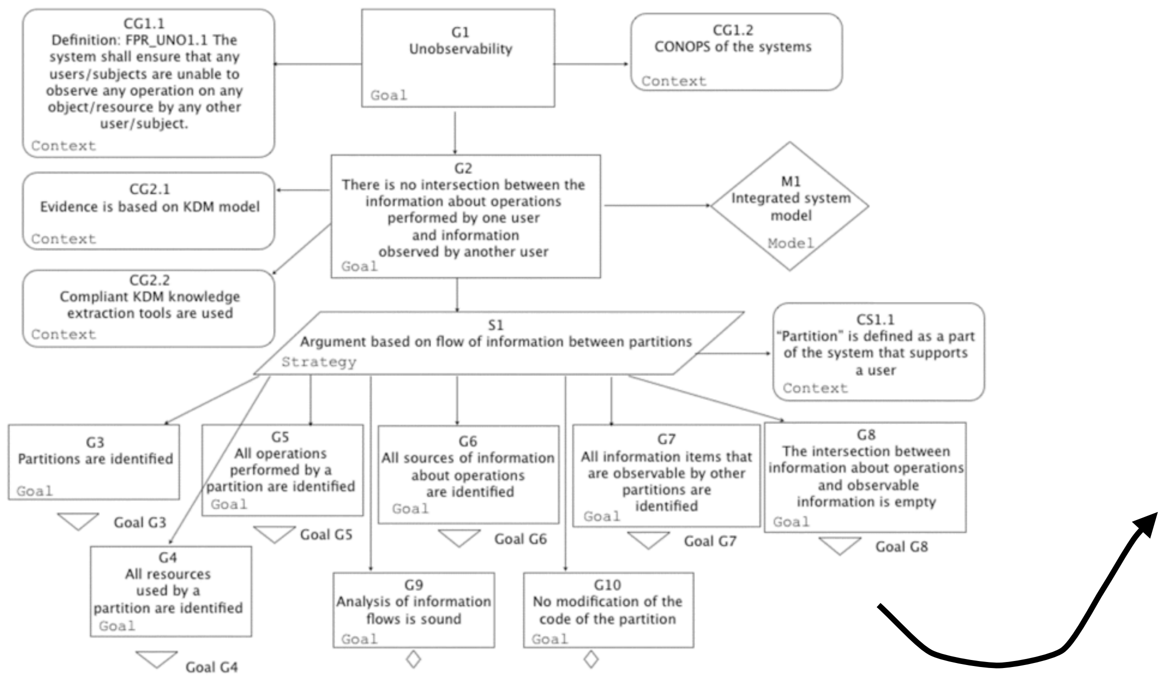
Unobservability Assurance Case (cont'd)

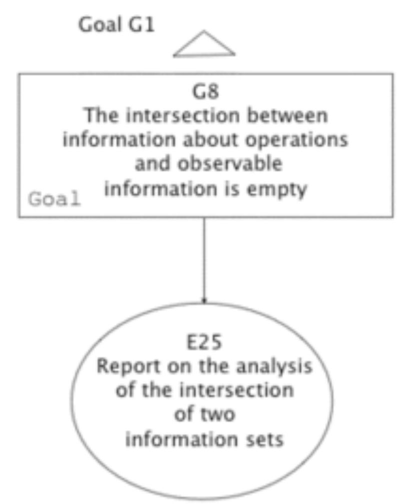


Unobservability Assurance Case (cont'd)



Unobservability Assurance Case (cont'd)



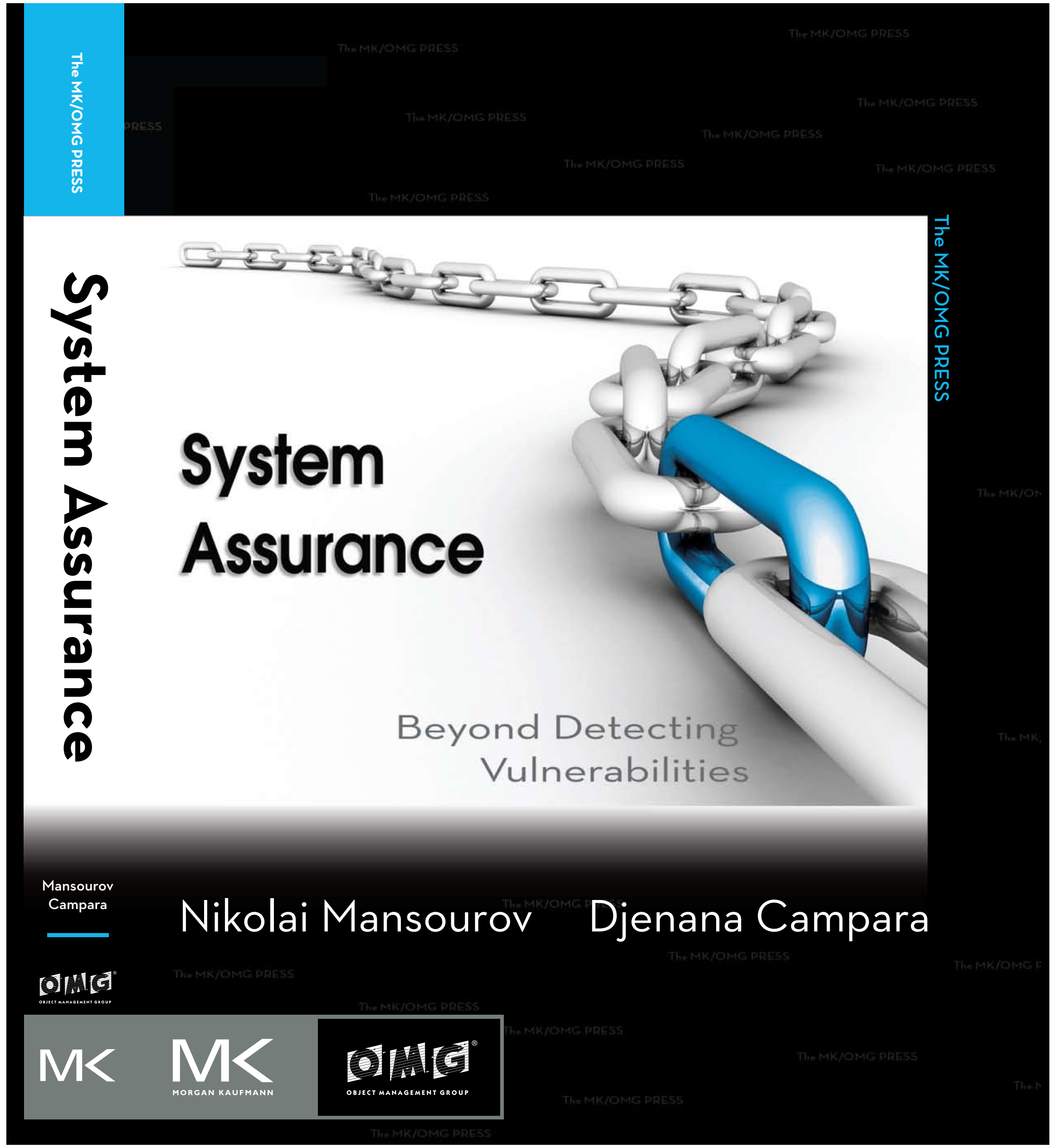


This claim actually generates the verdict based on the analysis of facts collected by the previous steps. When there is sufficient evidence to justify the "no flow between partitions" claim, this generates confidence in the effectiveness of the countermeasures against the observation risk. This confidence is propagated up the claim tree and is combined with the confidence in supporting claims

Conclusions

- OMG protocol stack for assurance knowledge focuses on common semantics and natural language
  - claims, arguments, assumptions, context
  - evidence
  - system facts
  - threats, risk, countermeasures
- Meaningful exchanges in assurance are fairly fine grained
- Entire arguments are represented as facts and linked to evidence
- Management of evidence links as facts
- Uniform, normalized fact-oriented environment industrializes knowledge exchanges in software assurance
  - separates produces and consumers of assurance knowledge
  - allow independent development of assurance tools
  - allows accumulation and exchange of patterns
- Economies of scale

New book!
Now available
at
amazon.com
and
bookstores
near you



http://www.kdmanalytics.com