Affordable, fact-oriented assurance with OMG standards

Nikolai Mansourov and Djenana Campara
KDM Analytics, Inc.
{nick,djenana}@kdmanalytics.com

Abstract

The OMG fact-oriented approach to assurance allows full traceability between the system facts and the assurance claims, so that the system facts can be gathered by automated tools and used as evidence to justify assurance claims. OMG Software Assurance Ecosystem defines a stack of standards to enable interoperability for tools, services and machine-readable content for assurance of existing software based systems.

1. Introduction

> The greatest improvement in the productive powers of labour, and the greater part of the skill, dexterity, and judgement with which it is any where directed, or applied, seem to have been the effects of the division of labour.
>
> **Adam Smith, The Wealth of Nations**

Affordable solutions are needed to achieve the assurance necessary for security-critical, safety-critical, and mission-critical components and systems. There are more such systems as the common wisdom may suggest, as each system stakeholder has own threshold of what is considered critical. Current assurance solutions can be afforded by few organizations for which criticality is defined in terms such as "loss of life" or "damage to the national security" - which includes governments, and the so-called "critical infrastructure" organizations, large banks, etc. Other types of organizations even individuals also use systems, and face certain risks often with dire consequences, however many of them can not afford adequate assurance and end up accepting risks that are too high for their operations.

Software based systems (or cyber systems) present a significant challenge for assurance assessment due to their complexity. Very rarely does a software development project start from a clean slate. More than half of all code written today is based on enhancements and maintenance of existing functions. More often than not, new features or functionality on top of an existing code base is a large, complex piece of software that has evolved into conflicting or challenging designs that often resists evolution and/or bug fixes. Furthermore, market consolidation brings the challenge of integrating these components into net-centric systems due to the mergers and

acquisition activities, making an existing system even larger, more complex, and harder to comprehend. The system's structure includes interconnected software components that were developed using different methodologies, a variety of technologies, and under varying constraints and assumptions. At the same time, the documentation of such systems is never up-to-date and the information available is largely folk- or judgment-driven in nature and difficult to access. The only thing trusted to be an up-to-date source is the code itself; however, comprehension of such a code is further obscured by uncontrolled use of multiple programming languages. This causes the lack of control over software architecture, leading to erosion of the initial structural concepts of the system, increasing a system's complexity even more.

Heavy reliance on COTS/Open Source products further impacted by globalization trends where modern software development and supply chain support are becoming increasingly distributed worldwide. This means that assessing the security posture of systems needs to go beyond evaluating the software application; it needs to include evaluation of the software supply chain, the development process, and pedigree of the development team to address growing concerns regarding the ability of an adversary to subvert the software supply chain and insiders' attacks.

Majority of current assurance practices are relatively informal and very subjective. Due to difficulties caused by development trends and systems' complexities, they focus primarily on evaluating the development process and a product's documentation rather than machine-readable system artifacts. Formal proofs are laborious and costly and often subjective because of the disconnect from the system artifacts, making it prohibitively expensive for most systems to be assessed in this way. For all these reasons, it is becoming increasingly difficult to establish or verify whether or not software is sufficiently trustworthy, and as complications will likely continue to evolve, a new approach needs to be developed to provide credible, repeatable, and cost-efficient ways for assessing a system's trustworthiness and for managing system's risks; an approach that will still give us systematic and methodical system coverage in identifying and mitigating vulnerabilities.

When discussing the cost of assurance it is important to view assurance as a knowledge intensive *product*. In order to produce assurance in an affordable way, we need to understand the operations involved in the assurance process and focus on the interoperability that enables division of labour in producing assurance. Assurance knowledge should be treated as *content* that is produced and consumed throughout an assurance project. Exchanges of knowledge related to the assurance of the system of interest must be done in machine-readable format in order to industrialize assurance and increase the number of organizations that can afford adequate assurance. Knowledge exchanges in assurance can work for the same system as well as across families of

systems. Efficient handling of knowledge content across multiple assurance projects involving the same system guarantee that knowledge of the system of interest does not dissipate once discovered, but is reused in subsequent assurance projects. Assurance knowledge can be accumulated for entire families of systems or operational environments. For example, the common knowledge of threats for certain operations within a particular environment can be accumulated and shared between multiple systems, both new and existing. Once the standard protocols for exchanging assurance content are available, automated tools can be build to facilitate discovery, transformation and management of the assurance facts.

Understanding the detailed flow of knowledge within an assurance project and standardization of the assurance content contributes to effective and efficient modular construction and certification of assured systems from assured components. In other words, the affordable assurance is layered assurance, and the question is what is appropriate granularity of layers in assurance that result in efficient flow of assurance knowledge in the context that involves multiple participants.

1.1. The Object Management Group

The Object Management Group (OMG) is uniquely positioned to standardize assurance knowledge. OMG is an international, open membership, not-for-profit computer industry consortium. OMG Task Forces develop enterprise integration standards for a wide range of technologies, including: Real-time, Embedded and Specialized Systems, Analysis & Design, Architecture-Driven Modernization and Middleware and an even wider range of industries, including: Business Modeling and Integration, C4I, Finance, Government, Healthcare, Legal Compliance, Life Sciences Research, Manufacturing Technology, Robotics, Software-Based Communications and Space. OMG membership includes hundreds of organizations, with half being software end-users in over two dozen vertical markets, and the other half representing virtually every large organization in the computer industry and many smaller ones. Dozens of standards organizations and other consortia maintain liaison relationships with OMG. OMG is an ISO PAS submitter, able to submit our specifications directly into ISO's fast-track adoption process. Several OMG's standards are already ISO standards and ITU-T recommendation.

OMG has been successful in the past developing knowledge exchange standards for modeling new systems, including cyber systems. OMG's modeling standards, including the Unified Modeling Language (UML) and Model Driven Architecture (MDA), enable powerful visual design, execution and maintenance of software and other processes, including IT Systems Modeling and Business Process Management.

However there is also a vast amount of highly functional, operational software representing enormous commercial value deployed in organizations around the globe. Existing systems are

defined as any production-enabled software, regardless of the platform it runs on, language it's written in, or length of time it has been in production. These entrenched software systems often resist evolution because their strategic value and ability to adapt has diminished. Common examples of such factors are a system's inability to be understood or maintained cost-effectively, inability to interoperate or dependence on undesired technologies or architectures. To address the challenges of existing cyber systems, the OMG's Architecture-Driven Modernization (ADM) Task Force was formed. The standards developed by the ADM Task Force enable understanding, maintenance and evolution of existing software systems.

The System Assurance Task Force shares the interest in existing complex software-based systems with the ADM Task Force, because system assurance also involves understanding of existing software systems. The standards developed by the OMG System Assurance Task Force enable exchanges of assurance knowledge leading to the establishing of the Software Assurance Ecosystem.

1.2. The OMG Software Assurance Ecosystem

An *ecosystem* refers to a community of participants in knowledge-intensive exchanges involving an explicit and growing shared body of knowledge and corresponding automated tools for collecting, analyzing, and reporting on the various facets of knowledge, as well as a market where participants exchange tools, services, and content in order to solve significant problems. The essential characteristic of an ecosystem is establishment of knowledge content as a product. An ecosystem involves a certain communication infrastructure, sometimes referred to as "plumbing," based on the number of knowledge-sharing protocols provided by a number of knowledge-management tools [5].

The purpose of the ecosystem within the assurance community is to facilitate collection and accumulation of assurance content, and to ensure its efficient and affordable delivery to the defenders of cyber systems, as well as to other stakeholders.

2. Software Assurance

"Assurance" can be also referred to as "system assurance" or "software assurance", when emphasizing the importance of vulnerabilities in the software itself. The nature of assurance is to provide a reasonable, justifiable answer to the question of whether the *countermeasures* implemented in the target system adequately mitigate all *threats* to that system. This answer can be called the *risk posture* of the system (this is not a common term because various communities focus at certain subsets of risks, e.g. security posture, safety posture, etc.). Evaluating the risk posture requires detailed knowledge, in particular, the knowledge of the factors internal to the

system, such as the system boundaries, components, access points, countermeasures, assets, impact, policy, design, etc., as well as the factors that are external, such as threats, hazards, capability and motivations of the threat agents, etc. Vulnerability can be described as a situation where a particular threat is not mitigated, and there exists a possibility of a mishap, waiting for a particular random event or for a motivated attacker to discover this situation, and execute the attack steps.

Producing assurance related to the risk posture of the system is not easy due to uncertainty factors. There is a great deal of uncertainty associated with any external factors, but there is uncertainty associated with the internal factors too, because our knowledge of the complex target system is not perfect. There may be some uncertainty about the impacts of the mishaps to the business and even about the validity of the security policy. In a slightly different sense, there is uncertainty about the behavior of the system, but this uncertainty is only related to the current state of our knowledge, and can be removed (at a certain cost) by analyzing the system deeper.

Assurance is a systematic discipline that focuses on how to provide justification that the risk posture is adequate. Assurance studies methods for justifying the risk posture of systems. System assurance approach develops a clear, comprehensive, and defensible argument for risk posture of the system, supported by evidence.

Assurance justifies the decision to use the system for a given mission. Assurance is a direct way to build confidence in the system. Deficit of assurance can be directly transformed into the requirements for additional countermeasures. System assurance deals with uncertainty through a special kind of reasoning that involves the architecture of the system of interest. Instead of a never- ending process of seeking more and more data to eliminate some fundamental uncertainty, system assurance uses the “defence-in-depth” considerations to address the unknowns. From the system assurance perspective, deficiencies in the argument indicate defects in the defense. Particular points in the architecture of the system that prevent building a defensible assurance argument are the candidates for the engineering to fix. The new, improved mechanisms will directly support the assurance argument and, at the same time, improve the risk posture of the system.

While evidence of risks, produced by detecting vulnerabilities, can also be transformed into countermeasures, it does not build confidence in the system, because the absence of vulnerabilities makes only a weak indirect case about the risk posture of the system. The key difference of the system assurance approach and traditional risk assessment is that assurance directly justifies the claims about the risk posture of the system.

3. Claims, Evidence and Code

Assurance is the justified belief that the risks are acceptably low. From the system engineering perspective, this means that the risks are made acceptably low, by 1) knowing what they are, and 2) engineering adequate countermeasures. Assurance claims are supported by evidence of the understanding of risks and the effectiveness of the countermeasures in mitigating these risks. Assurance of a system of interest involves communicating this knowledge as a clear, comprehensive and defendable assurance case to the system stakeholders.

The OMG Assurance Ecosystem include the following standards:

- Argumentation Metamodel - a common protocol for developing and exchanging assurance arguments [7];
- Evidence Metamodel - an common protocol for gathering and exchanging assurance evidence [10];
- Knowledge Discovery Metamodel (KDM) - a common protocol for exchanging system facts [2,8].

The central artifact for assurance is the so-called assurance argument - a collection of claims (often - countermeasure effectiveness claims) and contract for the types of evidence that would be considered sufficient to justify the claim. The structure of the assurance argument, i.e. the decomposition of the claims into subclaims can be made visual [1,3,11] as illustrated in Figure 1. Here the top claim C1 is argued using the strategy A1 based on the two subclaims CR1 and C3. Element ER1 describes what kind of evidence is considered adequate to justify claim CR1 (at the given level of assurance). Figure 1 provides a simplified assurance claim. Instead of the end-to-end the claims related to the knowledge of risks and their mitigation by specific countermeasures the claim is reduced to the claims related to the knowledge of requirements and their satisfaction by the system.

The OMG Argumentation Metamodel [7] defines the common vocabulary for building assurance arguments, including the concepts of claim, argument strategy, causal support of claims, assumptions, and evidence references. The OMG approach to justifying assurance claims is semi-formal, based on the so-called “substantiative reasoning” [11].

In order to support the argument, concrete and compelling evidence must be collected. The sources of evidence are records of the performance of the certain system life cycle processes, certain artifacts generated by the system life cycle processes (such as ER1), records of validation

and verification activities, and the results of system analysis, performed in the context of assurance project. Assurance Evidence is usually either derived from the system artifacts (product-based evidence) or is produced during the life cycle (process-based evidence) or involves the facts based on expert judgment, and is presented to demonstrate that the claim to which it relates is valid (i.e., the claim statement is true) [4]. Often evidence fact is a record of some sort, demonstrating that a certain event took place or that a certain relationship exists.

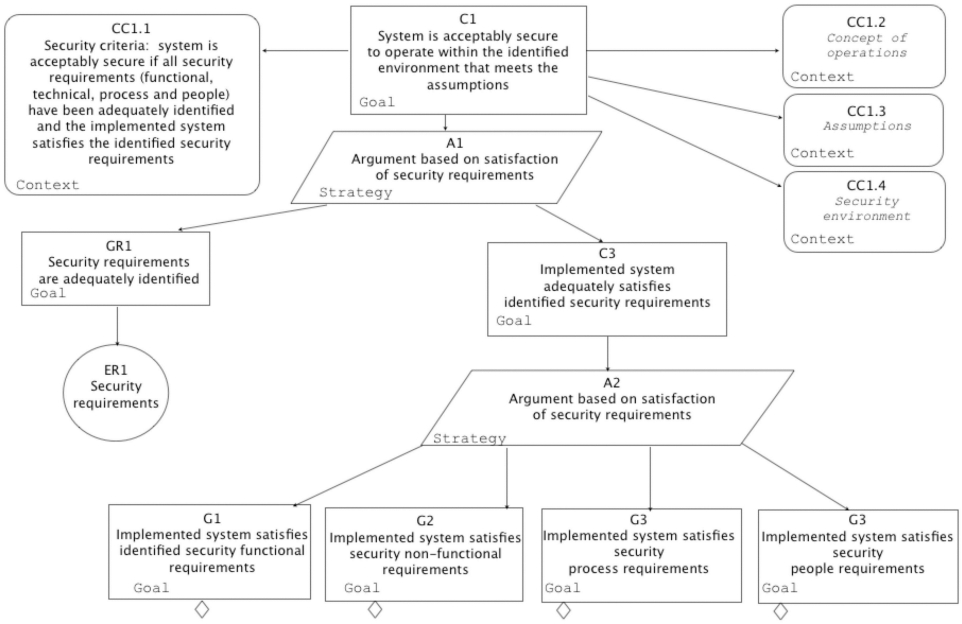

Figure 1

While evidence is the set of available, agreed-upon and undisputed facts about the system of interest, assurance argument addresses the conceptual distance between the available facts and the claim statements, for example, when the claim involves analysis, or when the available facts needs to be accumulated because none of them provides direct support to the claim.

Decomposition is repeated until there is no need for further simplification since the claim can now be objectively proven. End results of this divide-and-conquer strategy are sub-claims that do not need further refinement and could be viewed as a contractual agreement on how evidence will be collected and how much of evidence is required. These sub-claims form the set of measurable goals for collecting evidence. Figure 2 illustrates the decomposition of the "Unobservability" claim into several subclaims. This will be discussed further in the Case Study section.

The OMG Evidence Metamodel [10] defines the common vocabulary for managing and evaluating the information items that substantiate claims. Evidence can be diverse as various

things may be produced as evidence, such as documents, expert testimony, test results, measurement results, records related to process, product, and people. The OMG Evidence Metamodel defines the common vocabulary for assurance evidence and the corresponding meta-data (who-when-where-when; the chain of custody; confidence; and results of the evaluation).

Evaluation of evidence is required for complex claims, supported by multiple pieces of direct and indirect evidence. Evaluation of evidence is a systematic procedure by which each evidence item is weighted according to the strength of its support to the corresponding claim. Evaluation of evidence concerns with counter evidence to the claim.

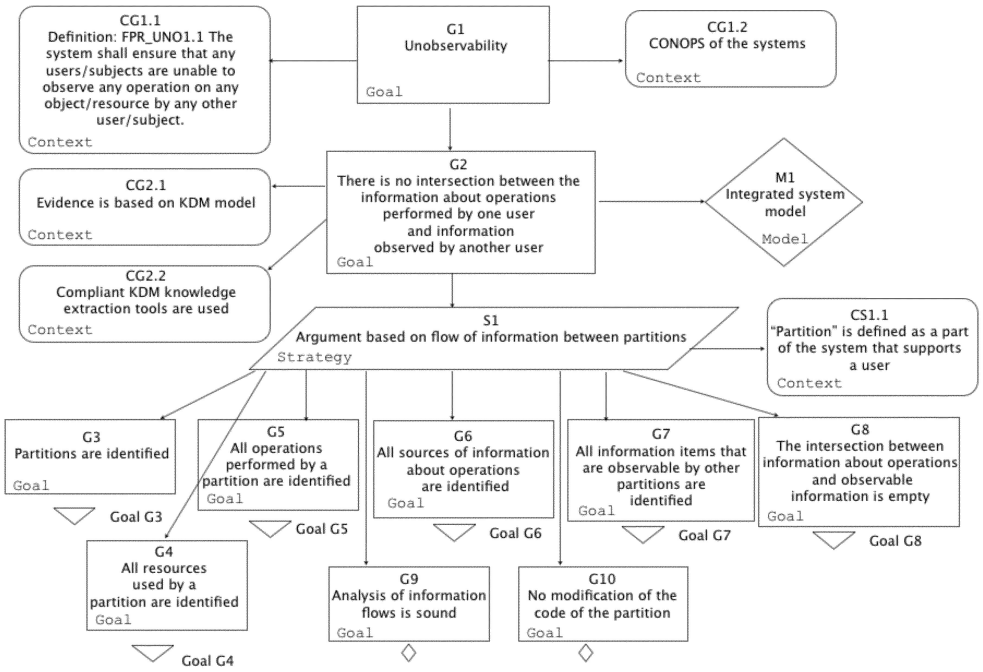

Figure 2

Fact-oriented system assurance is characterized by using an integrated system model as the primary source of direct evidence and supporting traceability between multiple external facts that are added into the integrated system model as evidence and the core facts related to the system of interest. The foundation of the OMG Assurance Ecosystem is KDM [8] which provides a standard protocol for exchanging facts about systems. KDM defines multiple views, including code, data, user interface, platform, structure and behavior views of the system of interest, and facilitates integration of other facts with these basic views, such as vulnerabilities, threats, risks and countermeasure views. KDM protocol allows importing facts from existing compilers and reverse engineering tools into a normalized fact-oriented repository, and linking them with the assurance facts (see Figure 3). Thus the assurance argument can be used to plan and manage the

system analysis and evidence gathering activities, while the resulting integrates system model provides the evidence elements to justify the assurance claims.

So, we need to identify risks, link risks to precise operational events, link the operational events to precise locations in the system of interest (as well as in the enabling systems), identify countermeasures, link countermeasures and then analyze the effectiveness of the countermeasures by analyzing the system of interest and the available evidence gathered according to agreed-upon evidence gathering procedures. The traceability from risks to events also works in the reverse direction: by studying the operational events, we can understand risks of the system of interest.

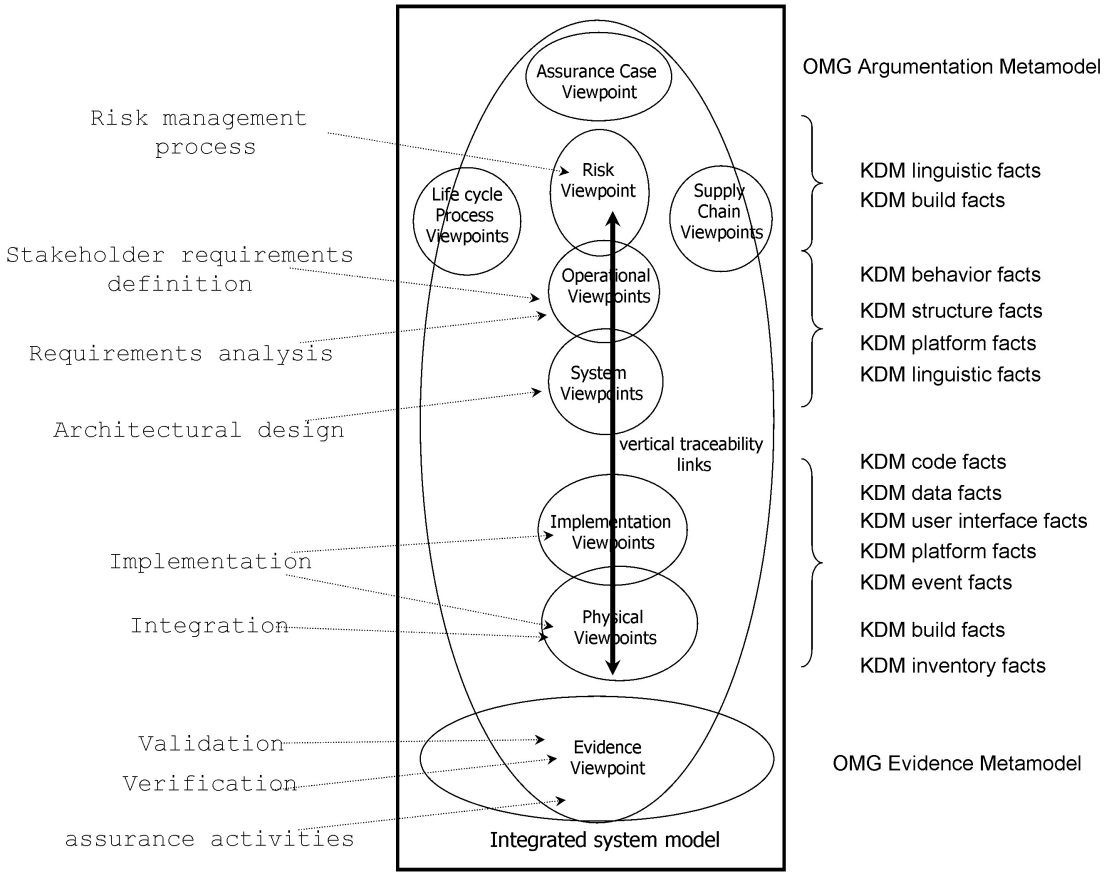

Figure 3

Collecting evidence for assurance, in particular within the defense-in-depth approach to uncertainty, is more demanding on the depth and accuracy of the system analysis and the accuracy of the corresponding data.

4. Fact-oriented approach

The OMG Assurance Ecosystem involves a rigorous approach to knowledge discovery and knowledge sharing where individual knowledge units are treated as *facts*. These facts can be described using statements, in a controlled vocabulary of noun and verb phrases using structured English and represented in efficient fact-based repositories, or represented in a variety of machine-readable formats, including XML.

More formally, a fact model is a combination of a conceptual schema, and for one specific system, a set of assertions, defined using only the concepts of the conceptual schema. Conceptual schema is a combination of concepts and statements of what is possible, necessary, permissible and obligatory in each possible world. Fact models focus on simple factual statements that assert the existence of individual objects/things and individual relations between these objects/things. Fact models are introduced in the OMG standard called Semantics of Business Vocabularies and Business Rules (SBVR) [9].

Assurance content is a collection of facts about the system of interest and its operating environment, and these facts need to be gathered, managed, accumulated, and shared. Affordable cost-effective assurance requires efficiency in discovering system knowledge, exchanging knowledge, integrating knowledge from multiple sources, collating pieces of knowledge, and distributing the accumulated knowledge. Evidence of the effectiveness of the safeguards have to be specific to the system of interest. Therefore, there is a need to discover accurate knowledge about the system of and use it together with the general units of knowledge such as the off-the-shelf vulnerability knowledge, vulnerability patterns and threat knowledge.

The OMG Software Assurance Ecosystem provides the infrastructure for assurance in the form of standard protocols that define information exchange contracts between the participants of the ecosystem. The first step toward achieving this goal is to agree on the set of discernable concepts in order to build the conceptual commitment. The second step is to represent all segments of information in a common format in which information can be distributed, man- aged, integrated, transformed, refined, and analyzed.

Fact models are related to the so-called linguistic models that deal with statements as they are used to express meanings. While fact models focus on simple operational meanings within a given conceptual commitment (existence of objects, discernability of objects as representatives of certain concepts, characteristics of individual objects and relationships between objects), linguistic models focus on meanings corresponding to the conceptual commitment itself, regardless of the actual objects, such as what are the characteristics of concepts, how concepts are defined in terms of other concepts, and what relationships between concepts are necessary, permissible or

obligatory. Thus statements in fact models describe individual *views* while statements in linguistic models describe *viewpoints*. Linguistic models focus at the definitions of new meanings, therefore they facilitate unconstrained communications where new meanings are defined on-the-fly. The flexibility of unrestricted linguistic communication comes with a price: the intended meaning needs to be unraveled from the expression, and the more complex the meaning, the more complex is the unraveling process. The key challenge of unrestricted linguistic communication is not so much in parsing a natural language sentence (which can be quite complex) but rather in dealing with complex meanings that are defined on-the-fly by some sentences and are used in the follow-up sentences.

Fact models, on the other hand, facilitate constrained communications in which the vocabulary is preselected; the forms of expressing this meaning are preselected, and consequently the range of meanings is bounded. Fact models do not define new meanings on-the-fly. Instead, they simply express meanings from the preselected set defined by the vocabulary. These boundaries (the vocabulary, the form of expressing meanings, and the intended range of meanings) become the contract between the participants of the information exchange. This contract is essential for repeatable information exchanges.

The fact-oriented approach allows discovery of accurate facts by tools, integration of facts from multiple sources, analysis, and reasoning, including derivation of new facts from the initial facts, collation of facts, and verbalization of facts in the form of English statements. Generic units of knowledge can also be represented as facts and integrated with the concrete facts about the system of interest.

5. Case Study

Let's consider a single security requirement, called the "Unobservability" which is stated as follows: "the system shall ensure that all users/subjects are unable to observe any operation on any object/resource by any other user/subject" [6]. The linguistic analysis of this property to identify the noun and verb concepts involved to provide guidance to the development of the assurance case for this property. The first tier simply lists the original noun and verb concepts used in the formulation of the Unobservability property.

The noun concepts are System, User/subject, Object/resource, and Operation. The verb concepts are : "System *involves* object/resource", "System *involves* operation", "User/subject *performs* operation *on* object/resource" and "User/subject *observes* operation"

These noun and verb concepts (the "unobservability" vocabulary) provide the conceptual schema for "unobservability" fact models. Particular facts can be as follows (using Prolog to represent 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').
```

These facts can be used as counter evidence to the claim that "The system clicks2bricks has the unobservability property". Facts from a different fact model using the same conceptual schema could be used as evidence that some other system does have the "unobservability" property. However, the conceptual schema is disconnected from the system artifacts (for example, the source code of the system), leaving the gap in the systematic and repeatable procedure for assurance.

The second tier maps these concepts onto the concepts available in the integrated system model. The additional noun concepts are: Partition, Activity, Information item.

We can use the concept Information item to split the verb concept "User/subject *observes* operation" into: "User/subject *observers* information item" and "Information item *records the result of* operation".

Further, we can introduce more concrete verb concepts: "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 observable by* partition", "Activity *discloses* information item *to* partition".

The new noun and verb concepts can be more easily identified in the system artifacts, making the assurance procedure more systematic and repeatable. Standards are required to make this approach practicable. Once a standard protocol for exchanging system facts is defined, an auto-

mated “knowledge discovery tool” can be independently implemented, that analyzes the system artifacts and produces the set of facts in the standard vocabulary, regardless of the particular assurance claim. On the other hand, the mapping of the operational vocabulary (such as the initial “unobservability” vocabulary) into the standard vocabulary of system facts, can be done independently on the implementation of the knowledge discovery tools. System facts must be independent on a particular implementation language, so multiple knowledge discovery tools can be developed by different vendors (for example as export adaptors from existing compiler and reverse engineering tools).

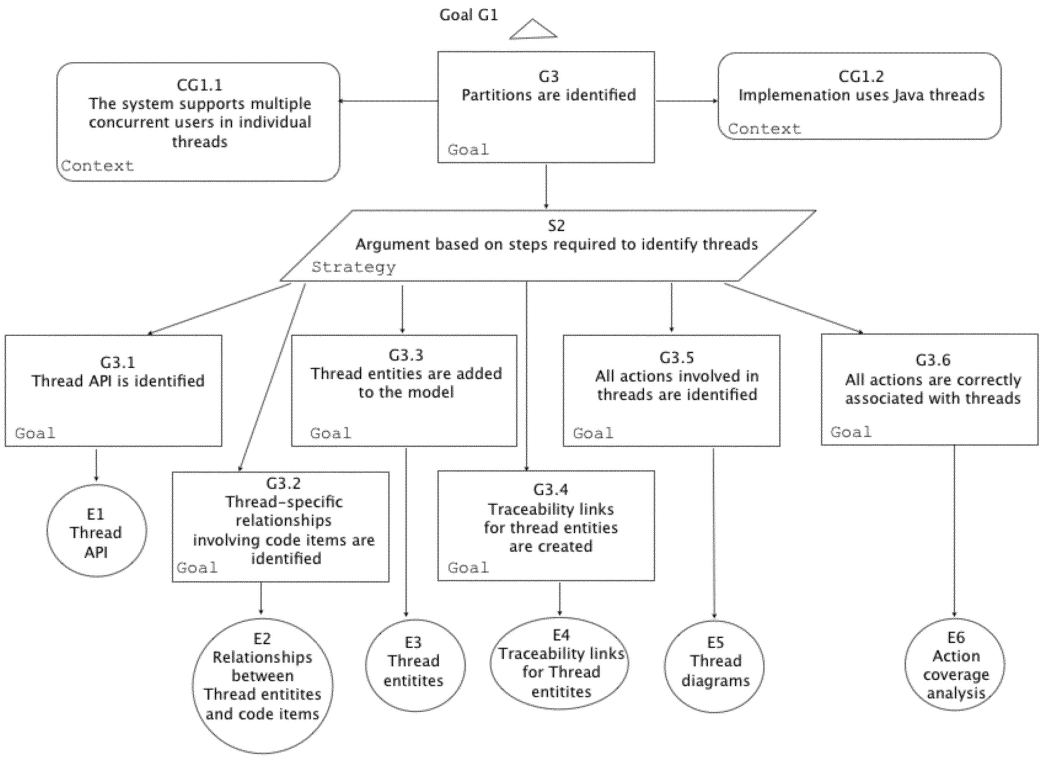

Figure 4

Facts can be managed in a repository. Assurance claims eventually turn into *requests for evidence collection* from the system artifacts and the development team and then - into the physical *queries* into the fact repository. Figure 2 illustrated the decomposition of the “Unobservability” claim into subclaims driven by the mapping of the operational vocabulary in the standard vocabulary of system facts. Figures 4-8 further illustrate each subclaim. Goal G3 describes the steps for identifying system-specific partitions in the baseline facts, starting with the identification of the particular platform API to create threads. Goals G4 and G5 identify system-specific resource and operations, and establish traceability links to the baseline facts.

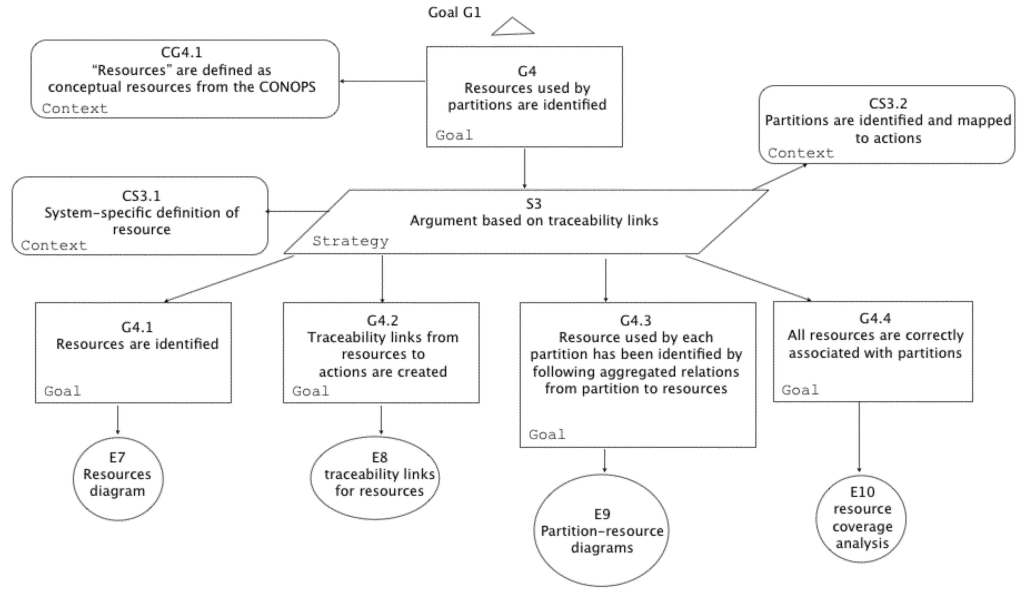

Figure 5

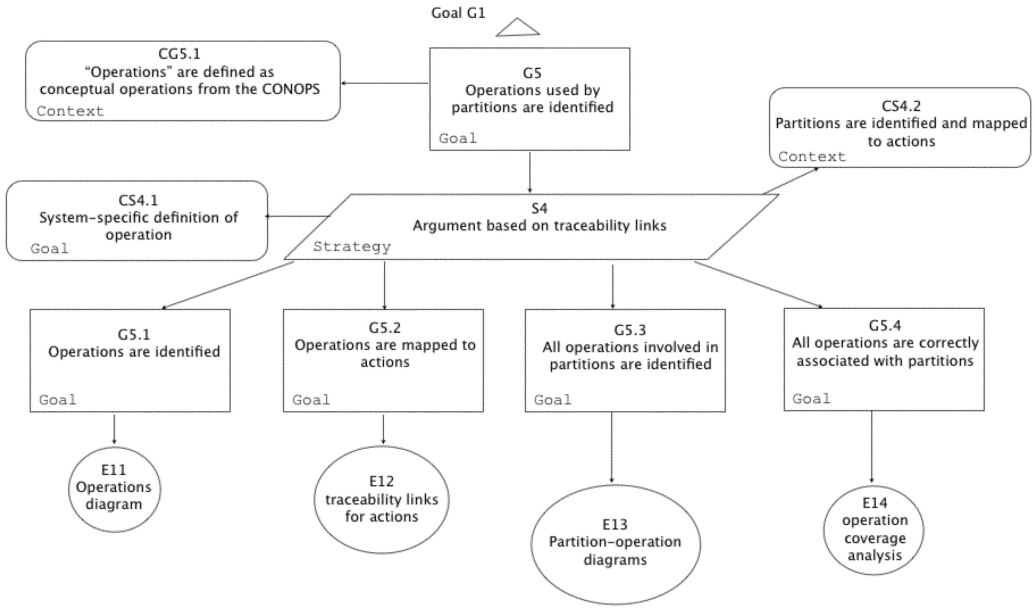

Figure 6

Goal G6 (see Figure 7) identifies sources of information about operations. Goal G7 identifies information items that can be observed by other partitions. Goals G6 and G7 only consider the system-specific operations, identified at Goal G5. When evidence gathering is concerned, Goals G4-G7 are records that certain system analysis steps have been performed. The gathered information is used in Goal G8 (see Figure 9) to evaluate the "Unobservability" property which directly contributes to the satisfaction of Goal G2 (see Figure 2). Goal G2 has two additional

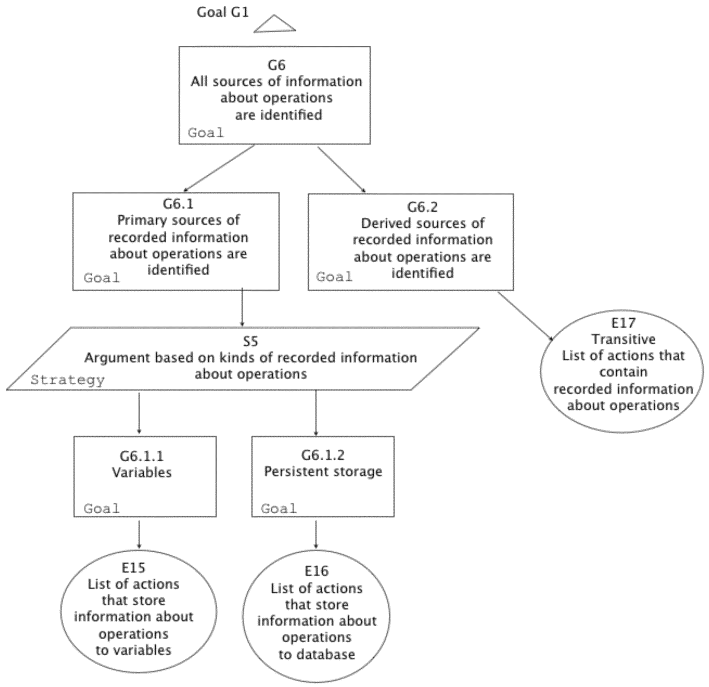

Figure 7

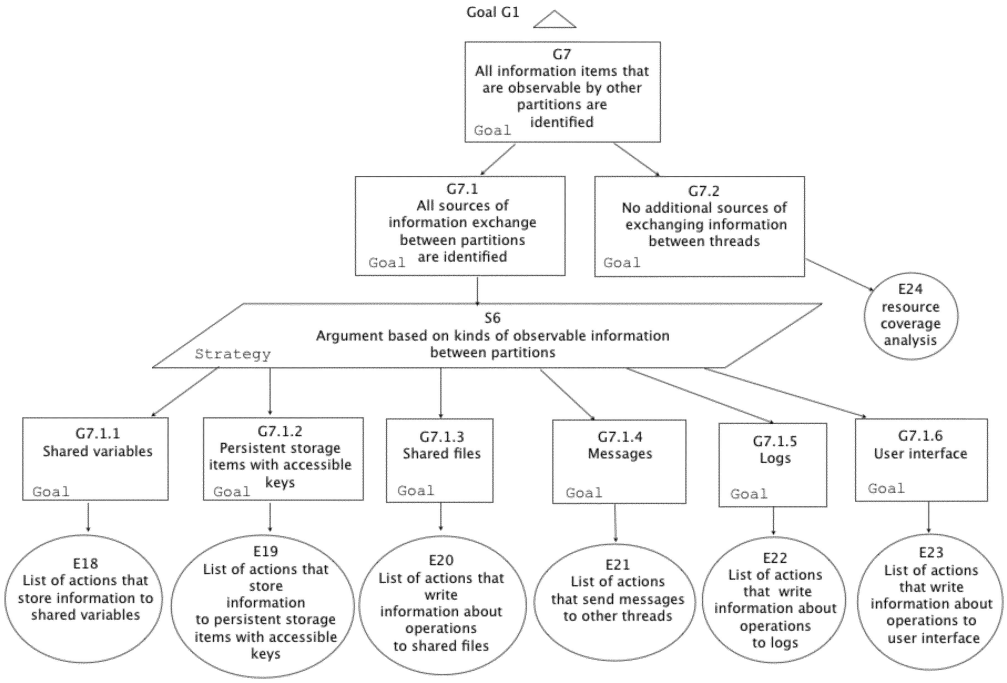

Figure 8

subclaims: Goal G8 requires evidence to the soundness of the analysis involved in evidence collections for Goals G3-G8. Goal G9 requires evidence that no code modification is happening within the scope of the system analysis, which justifies the collection of the lists of the sources of information about operation and the list of observable information items.

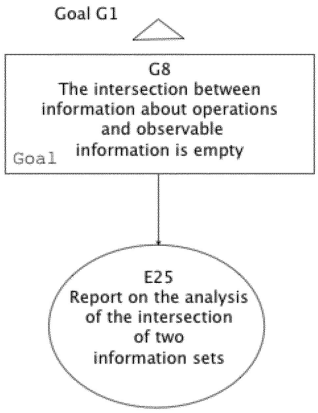

Figure 9

6. Conclusions

This uniform fact-oriented environment industrializes the use of knowledge in system assurance - allows description of the patterns of facts, sharing patterns as content and using automated tools to search for occurrences of patterns in the fact-based repository. Standard protocol for exchanging system facts, defined by the OMG Knowledge Discovery Metamodel, decouples the producers and consumers of concrete facts for the system of interest: the facts can be collected independently of the particular assurance claims by automated tools. On the other hand, the standard allows systematic decomposition of the assurance claims into subclaims based on the refinement of the vocabulary in such a way that eventually, the primitive claims can turn into physical queries into the fact repository. The key to the fact-oriented approach is the conceptual commitment to the common vocabulary which is the key contract that starts an ecosystem.

The OMG Asssurance Ecosystem defines an end-to-end fact-oriented environment, where assurance arguments are represented as facts, and can be reused. The system facts are integrated with the facts about the system's operational environment, using the vocabulary of assets, threats, risks, and countermeasures. The OMG Evidence Metamodel provides additional management capabilities for dealing with large collections of claims, subclaims and their traceability links to the individual facts. Understanding the detailed flow of knowledge within an assurance project and standardization of the assurance content contributes to effective and efficient modular construction and certification of assured systems from assured components, often at the fine levels of granularity.

The OMG Software Assurance Ecosystem defines a stack of standard protocols for the knowledge-based tools, including knowledge discovery tools, knowledge integration tools,

knowledge transformation tools, knowledge provisioning and management tools, and knowledge delivery tools.

An ecosystem opens up a market where participants exchange tools, services, and content in order to solve significant problems. The essential characteristic of an ecosystem is establishment of knowledge content as an explicit product, separated from producer and consumer tools, and delivered through well-defined protocols, in order to enable economies of scale and rapid accumulation of actionable knowledge.

7. References

1. Eurocontrol, Organization For The Safety of Air Navigation, European Air Traffic Management, *Safety Case Development Manual*, DAP/SSH/091. (2006).

2. ISO/IEC 19506 *Architecture Driven Modernization—Knowledge Discovery Metamodel*. (2009).

3. Kelly, T. P. (1998). *Arguing Safety – A Systematic Approach to Managing Safety Cases*. University of York, PhD Thesis.

4. Landoll, D. J. (2006). *The Security Risk Assessment Handbook*. New York, NY: Auerbach Publications.

5. Mansourov, N., Campara, D., *System Assurance: Beyond Detecting Vulnerabilities*, Morgan-Kaufmann, 2010

6. Merkow, M. S., & Breithaupt, J. (2005). *Computer Security Assurance Using the Common Criteria*. Clifton Park, NY: Thompson Delmar Learning. NDIA, Engineering for System Assurance Guidebook. (2008).

7. Object Management Group, *Argumentation Metamodel (ARM)*. (2010).

8. Object Management Group. (2006). *Knowledge Discovery Metamodel (KDM)* 1.2.

9. Object Management Group. (2009). *Semantics of Business Vocabularies and Rules (SBVR)* 1.1

10. Object Management Group. (2010). *Software Assurance Evidence Metamodel (SAEM)* 1.0.

11. Toulmin, S. E. (1984). *An Introduction to Reasoning.* New York, NY: Macmillan.