To appear in Proceedings of the 13th IEEE International Requirements Engineering Conference (RE '05)
Workshop on Requirements Engineering for High-Availability Systems (RHAS), 8/29 – 9/2, Paris, France

Engineering Dependability Requirements for Software-intensive Systems through the Definition of a Common Language

Seok-Won Lee and Robin Gandhi
Dept. of Software and Information Systems, The University of North Carolina at Charlotte
Charlotte, NC 28223-0001, USA. {seoklee, rgandhi}@uncc.edu

Abstract

Engineering dependability requirements for software-intensive systems is inherently difficult. Dependability of these systems relies heavily on the emergent properties that result from the complex inter-dependencies that exist among the involved systems and their environments. Furthermore, the choice of a modeling technique significantly affects the semantics and the level of abstraction at which these systems are modeled and analyzed. Therefore, to effectively predict the emergent properties of the system as a whole, it is necessary to gather information based on multiple philosophies from complementary modeling techniques and analyze them in the context of each other. To realize such a unified approach during the early stages of the RE lifecycle, we advocate the need for the definition of a common language. The common language provides a framework within which several modeling techniques can be used in harmony to elicit and create a common understanding through the problem domain concepts, properties and their relationships. We provide examples from our case study on automating the standard Department of Defense Information Technology Security Certification and Accreditation Process (DITSCAP) to motivate the applicability and appropriateness of our approach.

1. Introduction

We are witnessing ever more reliance on "systems of systems" that have emerged as complex amalgamations of underlying software, systems, practice, and environment in government as well as industry. The wide-spread use of such systems in critical applications that range from embedded process controls with mostly solid-state static interfaces to large-scale information infrastructures operational in dynamic socio-technical environments requires them to exhibit compelling evidence that their services satisfy certain dependability properties [29]. Such dependability is mostly expressed in terms of safety, security, availability, and reliability dimensions [36]. As we start to consider these dimensions for *systems of systems*, we realize that they rely heavily on the characteristics and capabilities of the system working as a whole to satisfy their real world goals and objectives. In essence, dependability along with other non-functional and functional features arises from the emergent properties exhibited by the collective influences of individual systems on each other as well as their propagative effects throughout the system. Such influences build a cluster of closely interdependent systems of systems. In the class of such systems, the ones in which software offers a significant portion of the system functionality are commonly referred to as software-intensive systems [38].

Software-intensive systems that are operational in socio-technical environments typically involve interactions between software, hardware, people, data, physical spaces, organizational policies, standards, procedures, laws, and regulations. Naturally in such multi-faceted environments, emergent properties which affect dependability are inherently difficult to understand and predict. Emergent properties can provide unanticipated benefits or can deviate from required capabilities of the system. In all cases, emergent properties make predictions about dependability, which is potentially the greatest risk to software-intensive systems [3].

The key challenges in engineering dependability requirements for software-intensive systems can be summarized as: 1) Difficulty in understanding the complexity of the relationships between diverse system components in a socio-technical environment; 2) Difficulty in establishing the extent to which dependability requirements satisfy their real-world goals and objectives; 3) Emergent properties of the system as a whole influence dependability; 4) Multitude of dependability requirements and their interdependencies; 5) Varying semantics and levels of abstractions in specifying dependability requirements based on different methods and techniques; and 6) A wide-range of system/service stakeholders that address and evaluate dependability attributes from different perspectives. Also, these issues further cause breakdowns in communication between stakeholders as well as porting of information from one RE stage to another [7].

We contend that to address these challenges and their diversity, software-intensive systems require RE methods based on multiple philosophies from complementary modeling techniques to elicit and capture several dimensions of the problem domain. Individual modeling techniques are effective for capturing certain aspects of the system but themselves cannot guarantee the accuracy and comprehensiveness of the predicted emergent behavior. Therefore, to effectively predict the emergent properties of the integrated system it is necessary to gather information based on multiple philosophies from complementary modeling techniques and analyze them in the context of each other. We believe that the lack of a comprehensive and well-defined framework that can accommodate different philosophies, methods and techniques for modeling and analysis of dependability requirements is at the root of the problem. To address this issue, we contend that the definition of a *common language* is required during the early stages of the RE lifecycle. Such a common language provides a framework within which complementary philosophies and associated modeling techniques can be used in harmony to elicit and create a common understanding through the problem domain concepts, properties and relationships in the Universe of Discourse (UofD). As one would expect, the definition of such a common language requires a rich set of representations, modeling techniques and tools as well as systematic ways to collect and organize the necessary information, i.e. the *Science of Design* of engineering quality dependability requirements for software-intensive systems.

In this paper, we present an integrated and comprehensive framework which combines novel techniques from RE and knowledge engineering to define and utilize a common language during the early stages of RE. Within the framework we also introduce the concept of Multi-Dimensional Link Analysis (MDLA) that supports early identification/prediction of emergent properties that affect dependability, by means of its traceable rationales and tool support. We understand that different dimensions/types of dependability attributes are interrelated [16], but we currently focus on the security dimension through examples derived from our case study [20] on automating the DITSCAP [12].

2. The Definition of a Common Language

We believe that in order to engineer quality (co-operative, comprehensive and cost-effective) dependability requirements, it is necessary to capture a holistic view of the system. Such a view includes relationships between system attributes, the environment and the nexus of causal chains [18], spanning several dimensions and levels of abstractions that exist to satisfy the real world goals of the associated users, business and organization. Understanding user needs within the context of the proposed system and as part of the wider organizational setting can greatly increase the accuracy and completeness of real-world requirements [7]. Furthermore, for critical operations, early reflections on the emergent system properties are necessary to predict whether the eventual system behavior will be dependable or not. In the following subsections we outline the characteristics and components of a common language that embodies a comprehensive view of the system under consideration.

2.1. Building a Common Language

Ideally, the definition of a common language should encompass all dimensions of a problem domain, captured at various levels of abstractions from diverse viewpoints. Although such expressiveness in representation is practically infeasible, we use hierarchical organization of ontological concepts to capture several key dimensions of the problem domain with related properties and non-taxonomic dependencies among them. The ontological concepts, their properties and relationships are elicited from various sources using RE modeling methods and techniques that is most suitable. During the early phases of RE, such ontological concepts are elicited from users, natural-language documents, requirements enforced through standards, various taxonomies in the domain, transcripts, organizational policies, domain knowledge, environmental constraints, laws and regulations, etc. Finally as the higher level ontological concepts become available, they are instantiated at the leaf-node level with specific criteria that help in gathering the related user/system information. These leaf-nodes provide a way to establish the extent to which the higher-level abstractions are satisfied through specific policies, procedures or technical rationales in the actual environment. Figure 1 provides an overview of the various levels of abstractions in the definition of a common language. Such a structured organization of the problem domain knowledge has several benefits: 1) It provides the definition of a common language for interoperability and communication at various levels of abstractions; 2) Allows the projection of a system-wide view while analyzing requirements; 3) The level of completeness of the common language gives a clear indication of areas of the problem domain that require further exploration; 4) It provides traceability among concepts in the problem domain to understand their inter-dependencies.

As shown in Figure 1, the integral part of our framework is a Problem Domain Ontology (PDO) that provides the definition of a common language. The PDO is a machine understandable, hierarchical representation, engineered using object-oriented ontological modeling techniques. The PDO construction leverages knowledge engineering techniques to support meta-knowledge creation, representation and processing. It also provides the ability to query and browse structured information sources based on inference mechanisms. The inherent benefits of such a PDO lie in the uniformity of its representation allowing for its use, reuse and evolution through all stages of the RE lifecycle.

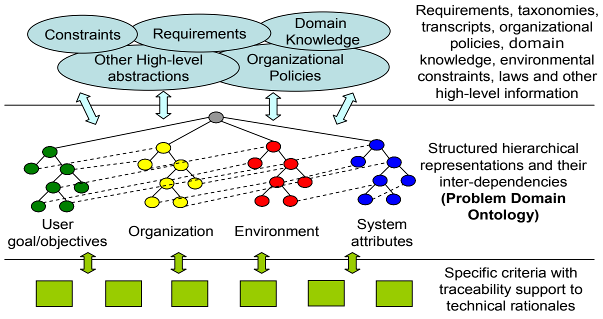

Constraints
Requirements
Domain Knowledge
Other High-level abstractions
Organizational Policies
Requirements, taxonomies, transcripts, organizational policies, domain knowledge, environmental constraints, laws and other high-level information
Structured hierarchical representations and their inter-dependencies
(Problem Domain Ontology)
User goal/objectives
Organization
Environment
System attributes
Specific criteria with traceability support to technical rationales

Figure 1: Information at various levels of abstractions in a Common Language

The PDO is modeled based on the GENeric Object Model (GenOM) [23]. GenOM is an integrated development environment for ontological engineering processes with functionalities to create, browse, access, query and visualize associated knowledge-bases. GenOM inherits the theoretical foundation of the frame representation and is compatible with the OKBC specification [6] as well as the OWL representation [28] format. GenOM's rich modeling constructs coupled with easily understandable semantics make it an optimal choice for the creation of a common language with participation from diverse stakeholders and experts in the UofD. The conceptual architecture of GenOM is shown in Figure 2.

The GenOM meta-language consists of *objects*, *properties,* and *features* with semantics that effectively support knowledge acquisition and representation. GenOM *objects* can be used to describe the concepts in a domain. It also supports modeling of hierarchical structures by single or multiple inheritance mechanisms. GenOM *properties* are used to describe the characteristics or attributes of *objects* and *features*. Finally, GenOM *features* are used to describe the relationship or dependencies that exist between objects. Once the objects, properties and features are defined, they are instantiated to represent specific instances that exist in a problem domain. GenOM is also associated with an inference engine [4] which supports reasoning based on the *objects*, *properties*, *features* and *instances* defined in its knowledge-bases. In summary, GenOM supports object modeling in its representation, usage of objects in its application model, and ability to aggregate evidence that supports the analysis of objects' behaviors (through the associated properties and relationships between objects).

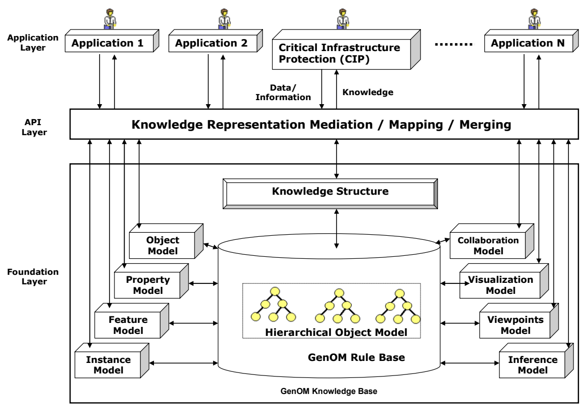


Figure 2: GenOM Conceptual Architecture

2.2 The Components of a Common Language

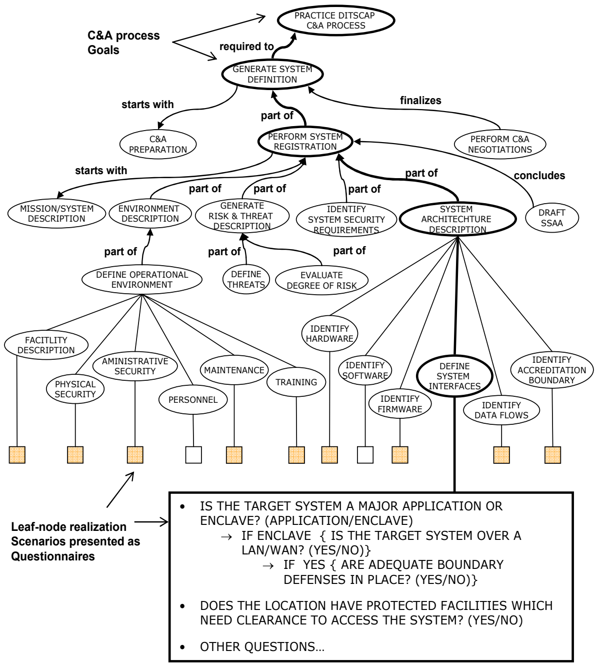
To systematically capture various aspects of the problem domain through the definition of a common language, we employ several popular and well-studied RE philosophies, methods and techniques. Specifically, we focus on RE modeling techniques based on the notions of goals [40], viewpoints [15], scenarios [5], and their combinations. Before outlining the fundamental models and techniques for the creation of a common language, we provide a brief introduction to DITSCAP [12], which will help in presenting these models using examples from the DITSCAP domain. DITSCAP examples focus on the security dimension of dependability requirements.

The role of DITSCAP is to maintain information assurance and the security posture of the Defense Information Infrastructure (DII) throughout the life-cycle of the information systems contained in it. DITSCAP focuses on the system mission, environment, architecture, and life cycle while assessing the impact of operation of the information system on the overall security posture of the DII [19]. In a nutshell, the motivation of DITSCAP is to identify and evaluate information security requirements applicable to the target system; outline a set of activities for performing Certification and Accreditation (C&A); generate required documentation; provide security solutions; and manage information system security activities. The DITSCAP problem domain is an ideal venue to study critical software-intensive systems, as it consists of systems which connect Department of Defense (DoD) mission support, command and control, and intelligence computers and users through voice, data, imagery, video, and multimedia services. These information processing and value-added services demand high quality of trust in the information furnished through them to the DoD and national-level decision makers.

We now introduce several models and techniques which systematically guide the creation of a common language in the DITSCAP domain.

2.2.1 Goal-driven Scenario Composition. RE techniques that combine the benefits of goal and scenario based approaches have been well researched [33] [26] [39]. To incorporate aspects of the system captured through these approaches in the definition of a common language, we present the goal-driven scenario composition technique. Following this technique, high level goals are successively decomposed into a set of specific goals whose realization criteria is captured using leaf-node scenarios. The goals in the hierarchy may express real-world goals of the system and its users; or from a process perspective, they can represent tasks and activities in a process. The leaf node scenarios capture specific user/system criteria related to the satisfaction of parent goals in the hierarchy. Such a goal-driven composition of scenarios helps in establishing their coverage over the problem domain concepts in addition to restricting their scope.

In the DITSCAP problem domain, goals of the C&A process are extracted from the homogenous groupings of well-defined tasks and activities outlined in the DITSCAP Application Manual [11]. The resulting hierarchical representation of the overall C&A process systematically guides the user through the DITSCAP. Furthermore, the leaf node scenarios employ carefully designed questionnaires that elicit user/system information required to identify the applicable DITSCAP-oriented security requirements as well as generate DITSCAP related documentation. To elaborate in further detail, we present an example where the security requirement of "Network Boundary Defenses" i.e. firewalls installed at appropriate places in a network, is being analyzed for a target system. Figure 3 outlines a path through the C&A goal hierarchy that will lead to a systematic exploration of the concepts related to this security requirement. The specific criteria gathered through the leaf-node scenarios also helps to prune the search space identified by the goals over DITSCAP-oriented security requirements.



- IS THE TARGET SYSTEM A MAJOR APPLICATION OR ENCLAVE? (APPLICATION/ENCLAVE)
  → IF ENCLAVE { IS THE TARGET SYSTEM OVER A LAN/WAN? (YES/NO)}
    → IF YES { ARE ADEQUATE BOUNDARY DEFENSES IN PLACE? (YES/NO)}
- DOES THE LOCATION HAVE PROTECTED FACILITIES WHICH NEED CLEARANCE TO ACCESS THE SYSTEM? (YES/NO)
- OTHER QUESTIONS…

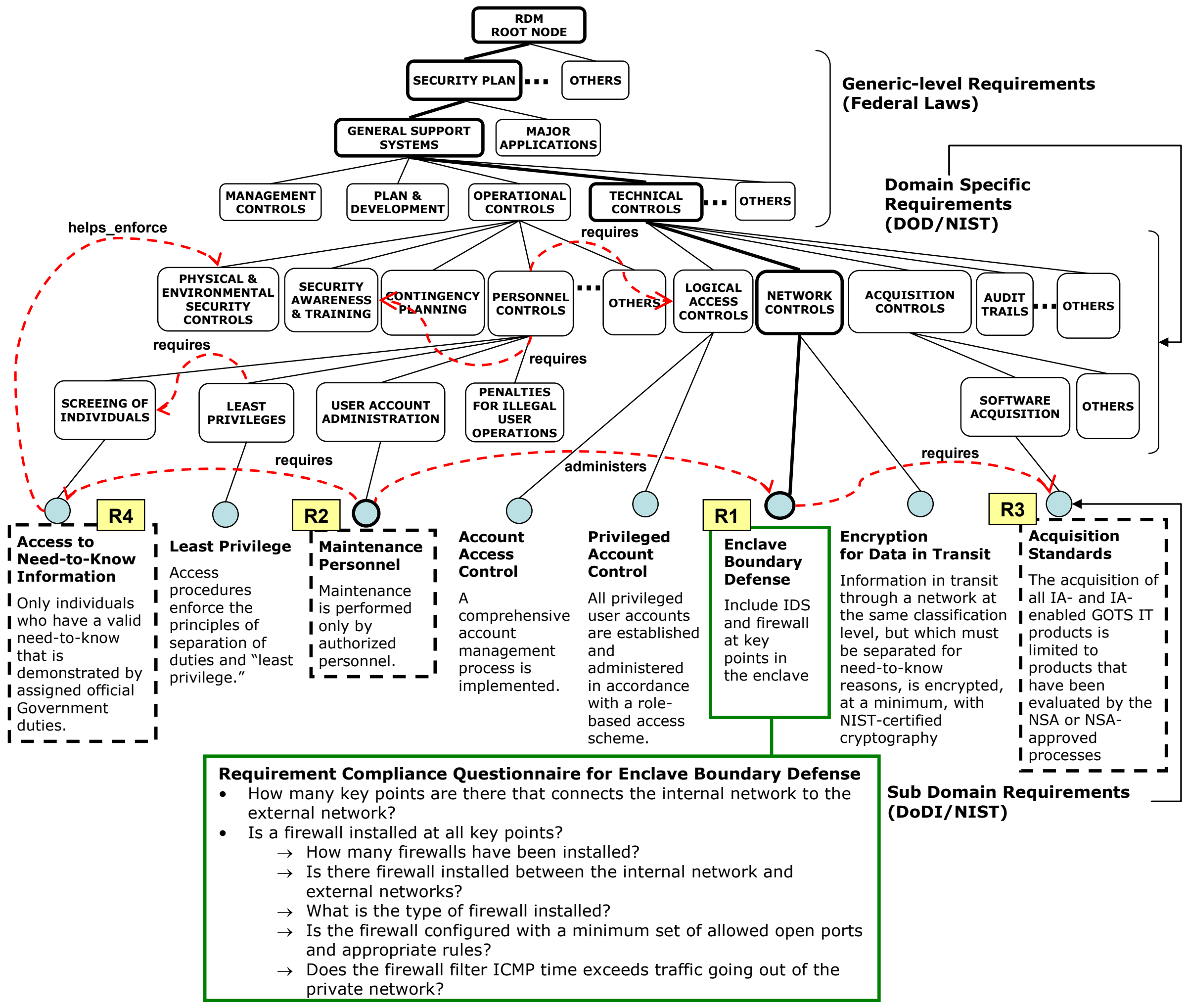
Figure 3: Partial DITSCAP C&A Goal Hierarchy

2.2.2 Requirements Domain Model (RDM). The RDM provides a systematic way to capture the problem domain requirements, their properties and inter-dependencies in the UofD. The RDM follows a hierarchical representation format that includes top-level generic requirements, mid-level domain spanning requirements and leaf-node sub-domain requirements. Such an organization of requirements allow for their exploration to be conservative in nature i.e. to be more inclusive rather than exclusive of aspects that need to be considered in early stages of the RE lifecycle. All requirements in the hierarchy also have several attributes that capture information about their descriptions, source, stakeholders, keywords, and dependent requirements. The sources of requirements in the RDM can be natural-language requirements, requirements from C&A standards, taxonomies, transcripts, manuals, organizational policies, domain knowledge, environmental constraints, laws and regulations and other domain-specific requirements sources. Apart from hierarchical relationships, there also exists several non-taxonomic links that represent relationships within the RDM as well as with other models in the definition of a common language. For example, based on the level of abstractions of the goals in the goal hierarchy, they map to requirements at the corresponding levels of abstractions in the RDM.

In the DITSCAP PDO, a RDM is constructed using security requirements extracted from DITSCAP-oriented security requirements such as high-level Federal laws [32], mid-level DoD policies [9] [37] [12], and leaf-node site-specific security requirements [10]. Continuing the example from the previous sub-section, Figure 4 shows a partial RDM, which is brought into focus based on the C&A goals and the leaf-node questionnaires shown in Figure 3. In Figure 4 the security requirement under consideration is labeled as R1 which is elaborated further using questionnaires that address specific criteria for its dependable operationalization. Furthermore, the non-taxonomic relationships are used to identify related requirements in other dimensions within the RDM, which are labeled as R2, R3 and R4 in Figure 4. Such relationships help to better understand and enforce the requirements for DITSCAP target systems.

The questionnaires for requirement compliance are usually derived from the elaborations of the security requirements in DITSCAP-oriented documents or best practices recommended in the DITSCAP domain. The questionnaire shown in Figure 4 is generated from the elaborations of that security requirement in [10], as well as the best practices suggested in [13]. Similarly the non-taxonomic relationships between requirements are also discovered from the identification of keywords in requirements descriptions and related best practices.

2.2.3 Other Supporting Hierarchical Models. In addition to the C&A goal hierarchy, other models in the DITSCAP PDO also relate to the RDM in various ways. One such model is a *viewpoints hierarchy* which is used to organize the concerns from a wide-range of



Only individuals who have a valid need-to-know that is demonstrated by assigned official Government duties.

Access procedures enforce the principles of separation of duties and "least privilege."

Maintenance is performed only by authorized personnel.

A comprehensive account management process is implemented.

All privileged user accounts are established and administered in accordance with a role-based access scheme.

Include IDS and firewall at key points in the enclave

Information in transit through a network at the same classification level, but which must be separated for need-to-know reasons, is encrypted, at a minimum, with NIST-certified cryptography

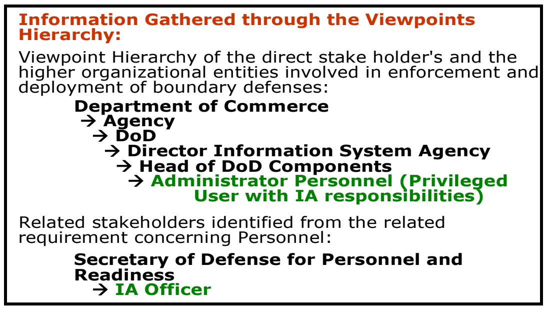
The acquisition of all IA- and IA-enabled GOTS IT products is limited to products that have been evaluated by the NSA or NSA-approved processes

Requirement Compliance Questionnaire for Enclave Boundary Defense

- How many key points are there that connects the internal network to the external network?
- Is a firewall installed at all key points?
  - → How many firewalls have been installed?
  - → Is there firewall installed between the internal network and external networks?
  - → What is the type of firewall installed?
  - → Is the firewall configured with a minimum set of allowed open ports and appropriate rules?
  - → Does the firewall filter ICMP time exceeds traffic going out of the private network?

Figure 4: Partial Requirements Domain Model in the DITSCAP PDO

stakeholders associated with a software-intensive system. Viewpoints [15] provide an effective way to organize the diversity of factors associated with the requirements in the RDM. In a viewpoints hierarchy, the higher level non-leaf nodes in the hierarchy consists of viewpoints, such as organizational viewpoints which map to generic and mid-level requirements in the RDM, and the leaf nodes usually represent viewpoints such as those of specific system stakeholders, services or concerns that are related to specific requirements in the leaf nodes of the RDM. To further elaborate on the requirement for "boundary defenses" in our example, Figure 5 outlines the various stakeholders identified from a viewpoints hierarchy in the DITSCAP PDO. The viewpoints in the DITSCAP domain are identified from the roles and responsibilities of various stakeholders related to requirements in DITSCAP-oriented documents.


Information Gathered through the Viewpoints Hierarchy:

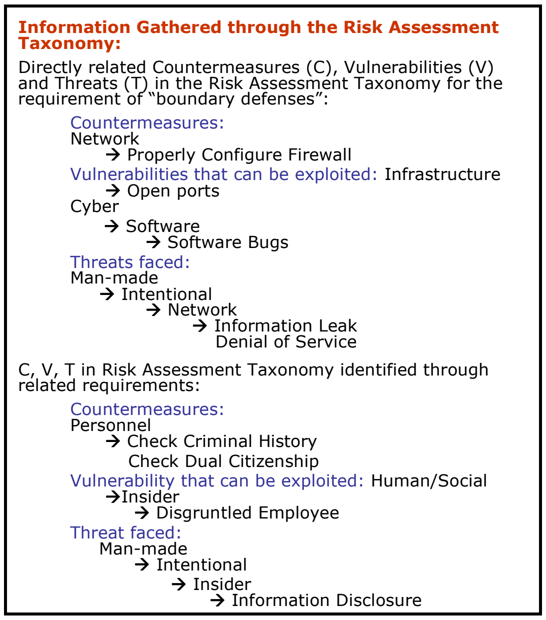
Viewpoint Hierarchy of the direct stake holder's and the higher organizational entities involved in enforcement and deployment of boundary defenses:

Department of Commerce
→ **Agency**
→ **DoD**
→ **Director Information System Agency**
→ **Head of DoD Components**
→ **Administrator Personnel (Privileged User with IA responsibilities)**

Related stakeholders identified from the related requirement concerning Personnel:

Secretary of Defense for Personnel and Readiness
→ **IA Officer**

Figure 5: Information from a Viewpoints Hierarchy


Information Gathered through the Risk Assessment Taxonomy:

Directly related Countermeasures (C), Vulnerabilities (V) and Threats (T) in the Risk Assessment Taxonomy for the requirement of "boundary defenses":

Countermeasures:
Network
→ Properly Configure Firewall
Vulnerabilities that can be exploited: Infrastructure
→ Open ports
Cyber
→ Software
→ Software Bugs
Threats faced:
Man-made
→ Intentional
→ Network
→ Information Leak
Denial of Service

C, V, T in Risk Assessment Taxonomy identified through related requirements:

Countermeasures:
Personnel
→ Check Criminal History
Check Dual Citizenship
Vulnerability that can be exploited: Human/Social
→Insider
→ Disgruntled Employee
Threat faced:
Man-made
→ Intentional
→ Insider
→ Information Disclosure

Figure 6: Information from a Risk Assessment Taxonomy in the DITSCAP domain

In the DITSCAP PDO we also include a *risk assessment taxonomy* which aggregates a broad spectrum of possible categories and classification of risk related information from the DITSCAP domain. The upper level nodes of the risk assessment taxonomy consist of threat, vulnerabilities, countermeasures, asset properties, and mission criticality concepts related to risk assessment. Each node is then further decomposed into more specific categories. In addition, several non-taxonomic links identify relationships within risk categories as well as with other entities in the DITSCAP PDO.

A risk assessment taxonomy is necessary to better understand security requirements as they naturally relate to various concepts of risk [21]. Such relationships can be discovered from various sources such as security requirements descriptions, research literature, taxonomies or from domain experts. Figure 6 depicts the risk information gathered for the requirement of "boundary defenses" used in our example, based on such relationships.

3. Multi-Dimensional Link Analysis

Examples in the previous section demonstrate how several core pieces of information that affect the emergent properties of the system can be systematically identified from the definition of a common language and its underlying models. In the DITSCAP domain, identification of such properties contributes to strengthen and effectively estimate the security dimension of system dependability.

Once the required pieces of information have been gathered from the PDO, they finally become valuable knowledge when they establish 'links' with each other from various aspects/dimensions based on a certain set of goals [22]. Following this paradigm, we introduce the concept of MDLA which can be triggered from different dimensions such as user criteria, viewpoints, system goals, scenarios, business/mission requirements, regulatory requirements, and risk categories. Such knowledge helps to understand the complex interdependencies and causal chains that exist between the real world goals, objectives and the components of a software intensive system. The PDO naturally fosters such analysis through the hierarchical organization of ontological concepts captured using several complementing RE modeling techniques. It increases the cohesiveness of information from various dimensions and provides inherent properties of an active approach to link requirements with other entities in the environment. Such an integrated framework for analytical analysis promotes a comprehensive coverage of the system's dependability attributes and systematically drives their elicitation, modeling and analysis by actively assisting in the process of discovering missing, conflicting, and interdependent pieces of information. A conceptual overview of MDLA in the DITSCAP domain is presented in Figure 7. It clearly demonstrates how different aspects of a requirement can be identified and analyzed within the framework of a common language based on information gathered from diverse system models with complementary semantics.

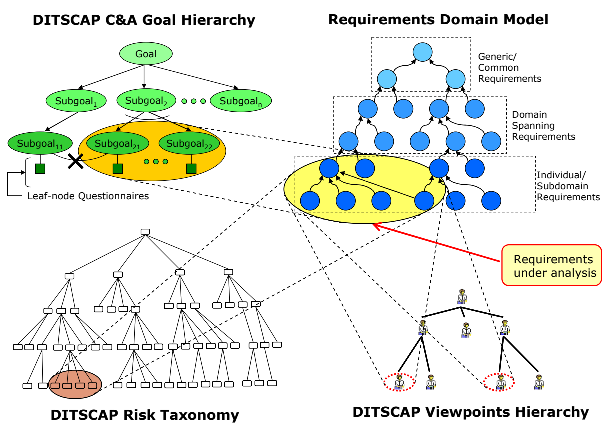

DITSCAP C&A Goal Hierarchy
Requirements Domain Model
Goal
Subgoal$_1$
Subgoal$_2$
Subgoal$_n$
Subgoal$_{11}$
Subgoal$_{21}$
Subgoal$_{22}$
Leaf-node Questionnaires
Generic/ Common Requirements
Domain Spanning Requirements
Individual/ Subdomain Requirements
Requirements under analysis
DITSCAP Risk Taxonomy
DITSCAP Viewpoints Hierarchy

Figure 7: Conceptual overview of MDLA in the DITSCAP PDO

4. Related Work

For critical software-intensive systems, formal methods have often been used to provide a priori evidence that the overall system behavior will be dependable [8]. Apart from being costly, formal approaches are not very effective to gain a common understanding during the early stages of RE. Their complexity also limits participation of stakeholders with diverse skills and expertise. Through the definition of a common language we seek to gather contributions from several information sources and provide an early opportunity for analyzing the emergent system behavior.

Throughout the RE lifecycle several, several approaches exist to elicit and model dependability requirements which are primarily driven by illicit usage or threat scenarios captured using misuse/abuse cases [35] [1], abuse frames [25], intruder anti-goals [41], or attacker modeling and analysis [27], but they uncover only a limited set of dependability attributes based on the current context of analysis. The use of general taxonomies [16] [17], attack patterns [30] and threats [34] to derive security and survivability requirements can contribute to the refinement and elaboration of ontological concepts in the PDO. However, we envision the classification and categorization of ontological concepts in the PDO to be primarily based on information gathered from the problem domain.

The LEL approach [24] was one of the first initiatives to support the elicitation and representation of a lexicon based on natural language processing, but the lexicon itself does not carry any information unless it is instantiated using a conceptual model. Currently, the use of a LEL to construct machine understandable ontologies resulting from the RE process, has been pointed out in [2]. Recent efforts for ontology based object-oriented domain modeling have also been expressed in [14].

We believe that the development of a common language in the early stages of RE, should not be restricted to specific modeling notions, constructs, or techniques as it may become too narrow-focused or stove-piped. Through our framework, we also strive to offer the flexibility in choosing the method or technique for RE that is most suited for the need of the situation, skills of the people involved or the uniqueness of the domain.

5. Conclusion and Future Work

Our contributions in this paper can be summarized as follows. Firstly, we identify specific issues in engineering quality dependability requirements for critical software-intensive systems, which motivate the need for the definition of a common language. Secondly, we present a comprehensive framework that integrates novel methods and techniques for the creation and utilization of a common language during the early stages of RE lifecycle. Examples presented throughout the paper demonstrate how several system models produced within the framework can be used in harmony to elicit and create a common understanding through the problem domain concepts, properties and their relationships. Finally, we introduce the concept of MDLA, which provides the ability to analyze different aspects of a requirement based on the information gathered from various system models available in the framework.

We believe that the definition of a common language offers several benefits throughout the RE lifecycle. It provides an opportunity for the early identification and prediction of emergent system attributes required for engineering quality dependability requirements. It also provides an integrated environment for the development of systematic processes of designing dependable systems and the related metrics and measures, i.e. the Science of Design of dependable systems. Furthermore, the definition of a common language can be reused across multiple systems with support for the incorporation/traceability of changes in policies, regulations, functional and non-functional requirements, and the real-world goals of the system.

As part of our future work we would like to address the following on-going research objectives: 1) Systematically identify the emergent properties of the system based on the evidences collected from various system models to establish the value of "objects", and their influences and interdependencies in the definition of a common language; 2) Establish metrics and measures of dependability based on the understanding and reflected language from various dimensions of a common language; and 3) Systematically analyze various system models within our framework to identify the "objects of interest" that affect dependability, accuracy, performance, usability, efficiency, criticality etc.

Acknowledgements: This work is partially supported by the grant (Contract: N65236-05-P-0597) from the Critical Infrastructure Protection Center, Space and Naval Warfare Systems Center, Charleston, SC. USA. The authors acknowledge the support from Scott West, John Linden, Bill Bolick, Gail-Joon Ahn and Bill Chu. Finally, the authors thank Deepak Yavagal, Divya Muthurajan, Raghuram Gururajan, and Swapnil Brahmankar for their contributions.

6. References

[1] Allenby, K., and Tim K., "Deriving Safety Requirements Using Scenarios", In Proceedings of the 5th International Symposium on Requirements Engineering, 2001, pp: 228-235

[2] Breitman, K.K., and Leite, J., "Ontology as a Requirements Engineering Product", In Proceedings of the IEEE Int'l Requirements Engineering Conference, Mini-tutorial on Ontology Development, 2003

[3] Brownsword, L. L., Carney, D. J., Fisher, D., Lewis, G., Meyers, C., Morris, E. J., Place, P. R. H., Smith, J., and Wrage, L., "Current Perspectives on Interoperability," Technical Report CMU/SEI-2004-TR-009, Carnegie Mellon Software Engineering Institute, 2004

[4] Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., and Wilkinson, K. 2004., "Jena: implementing the semantic web recommendations," In Proceedings of the 13th International World Wide Web Conference ACM Press, New York, NY, 74-83.

[5] Carroll, J. M., "Five Reasons for Scenario-Based Design", In Proceedings of the Hawaii Int'l Conf. on Systems Sciences, IEEE Computer Society Press, Los Alamitos, California, pp:123, 1999

[6] Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., and Rice, J. P., "OKBC: a programmatic foundation for knowledge base interoperability," In Proceedings of the Fifteenth National/Tenth Conference on Artificial intelligence/innovative Applications of Artificial intelligence, American Association for Artificial Intelligence, Menlo Park, CA, 600-607

[7] Coughlan, J., Macredie, D. R., "Effective Communication in Requirements Elicitation: A Comparison of Methodologies," Requirements Engineering, Volume 7, Issue 2, Jun 2002, Pages 47 – 60

[8] de Groot, A., Hooman, J., Lemoine, M., Gaudiere, G., Winter, V.L., and Kapur, D., "A survey: applying formal methods to a software intensive system", Sixth IEEE Int'l Symposium on High Assurance Systems Engineering, 2001, pp:55 – 64

[9] DoD 8500.1. Information Assurance. Oct. 2002

[10] DoD 8500.2. Information Assurance Implementation. Feb. 2003

[11] DoD 8510.1-M, "DITSCAP Application Manual", 2000

[12] DoD Instruction 5200.40, "DoD Information Technology Security Certification and Accreditation Process" (DITSCAP), 1997

[13] Ed Skoudis, *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, Prentice Hall PTR, 2001

[14] Evermann, J., and Wand, Y., "Ontology based object-oriented domain modeling: fundamental concepts", Requirements Engineering Journal, Springer-Verlag London Ltd., 2005

[15] Finkelstein, A., and Sommerville, I., "The Viewpoints FAQ", BCS/IEE Software Engg. Journal, Vol. 11(1), 1996

[16] Firesmith, D. G., "Common Concepts Underlying Safety, Security, and Survivability Engineering", Technical Report CMU/SEI-2003-TN-033, Carnegie Mellon Software Engineering Institute, 2003

[17] Hughes, K.J., Rankin, R.M., and Sennett, C.T., "Taxonomy for requirements analysis" Proceedings of the 1st Int'l Conference on Requirements Engineering, 18-22 April 1994, pp:176 – 179

[18] Jackson, M., "The Meaning of Requirements." Annals of Software Engineering, Vol 3, pp: 5-21, Baltzer Science Publishers, 1997

[19] Kimbell, J. and Walrath, M., "Life Cycle Security and DITSCAP", IANewsletter, Vol. 4, No. 2. Spring 2001

[20] Lee, S. W., Gandhi, R. A., and Ahn, G., "Establishing Trustworthiness in Services of the Critical Infrastructure: Automating the DITSCAP", Workshop on Software

Engineering for Secure Systems (SESS05), 27th International Conference on Software Engineering, May 2005, pp: 43-49

[21] Lee, S. W., Gandhi, R. A., and Ahn, G., "Security Requirements Driven Risk Assessment for Critical Infrastructure Information Systems", To appear in Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS 05), RE '05, Paris, France, August 2005.

[22] Lee, S.W. and Rine, D.C. Missing Requirements and Relationship Discovery through Proxy Viewpoints Model. Studia Informatica Universalis: Int'l. Journal on Informatics, Vol. 3(3), pp. 315-342, December 2004.

[23] Lee, S.W. and Yavagal, D., "GenOM User's Guide". Technical Report TR-SIS-NISE-04-01, Knowledge Intensive Software Engineering Research Group, Dept. of Software and Information Systems, UNC Charlotte, Spring 2004

[24] Leite JCSP, Franco APM. "A strategy for conceptual model acquisition," In proceedings of the IEEE international symposium on Requirements Engineering. IEEE Computer Society Press, Los Alamitos, CA, pp 243–246, 1993

[25] Lin, L., Nuseibeh, B., Ince, D., Jackson, M, "Using abuse frames to bound the scope of security problems", In Proceeding of the 12th IEEE Int'l Requirements Engineering Conference, 2004, pp: 354- 355

[26] Liu, L., Yu, E., "Designing Information Systems in Social Context: A Goal and Scenario Modeling Approach," Information Systems. Vol. 29(2), Elsevier, 2003

[27] Liu, L., Yu, E., Mylopoulos, J., " Security and privacy requirements analysis within a social setting", In proceedings of the 11th IEEE International Requirements Engineering Conference, 2003, pp: 151-161

[28] McGuinness, D., and van Harmelen, F. (editors), "OWL Web Ontology Language Overview", W3C Recommendation, 10th February, 2004, http://www.w3.org/TR/owl-features/

[29] McLean, J., and Heitmeyer, C., "High Assurance Computer Systems: A Research Agenda", America in the Age of Information, National Science and Technology Council Committee on Information and Communications Forum, Bethesda, 1995

[30] Moore, A. P., Ellison, R. J., and Linger, R. C., "Attack Modeling for Information Security and Survivability", Technical Note CMU/SEI-2001-TN-001, Carnegie Mellon Software Engineering Institute, 2001

[31] National Institute of Standards and Technology (NIST) Special Publication 800-12, "An Introduction to Computer Security: The NIST Handbook," October 1995

[32] Office of Management and Budget Circular No. A-130, "Management of Federal Information Resources," Feb 8, 1996

[33] Rolland, C., and Souveyet, C., and Achour, C. B., "Guiding Goal Modeling Using Scenarios", In IEEE Transactions on Software Engineering, Vol. 24(12), 1998, pp:1055-1071

[34] Schneier, B., "Attack Trees: Modeling Security Threats", Dr. Dobb's Journal, December 1999

[35] Sindre, G., Opdahl, A.L., "Eliciting security requirements by misuse cases", In Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems, 2000, pp:120-131

[36] Sommerville, I., Software engineering, Addison Wesley; 7 edition, May 10, 2004

[37] Swanson, M. "Guide for Developing Security Plans for Information Technology Systems," NIST Special Publication 800-18, 1998

[38] The Integration of Software-Intensive Systems (ISIS) initiative. Carnegie Mellon Software Engineering Institute, URL: http://www.sei.cmu.edu/isis/isis-main.html

[39] van Lamsweerde, A. Willemet, L. "Inferring declarative requirements specifications from operational scenarios" IEEE Transactions on Software Engineering, Vol. 24(12), 1998, pp: 1089 – 1114

[40] van Lamsweerde, A., "Goal-oriented requirements engineering: a guided tour", In Proceedings of the fifth IEEE International Symposium on Requirements Engineering, Aug. 2001 pp:249-262

[41] van Lamsweerde, A., Brohez, S., De Landtsheer, R., and Janssens, D., "From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering" In Proceeding of Requirements for High Assurance Systems Workshop (RHAS'03), 11th International Requirements Engineering Conference (RE'03), 2003