Essays’ Topics

1. Modeling multiple view
2. Modeling classifications
3. Multiple models: Traceability, Consistency, and accurate representation of one thing that has been modeled with multiple abstractions
4. Seven Analysis Guidelines
5. MRPF and Reuse
6. Roles vs. Actors
7. My CRC Cards
8. Type-Oriented Paradigm (TOP)
9. Types vs. Classes
10. Design Patterns vs. Hooks vs. Hotspots
11. Meta Modeling
12. Object vs. Agent
13. My Use Case Template

Require reading my “Software Stability” – 4 columns, Thinking Objectively, Communications of the ACM (will be posted on the web)

14. Complexity vs. Simplicity
15. Map the software stability concepts into the entire lifecycle
16. Use software stability concepts as a way to activate/ deactivate capabilities
17. Use software stability concepts as a way to add/ delete/ modify capabilities
18. Use software stability concepts as meta-language to describe and/or develop system software and other software domain
19. Use the stability model (SM) as an accessibility model using roles
20. Definitions (Simplification): EBTs, Bos, Ios
21. Design/ analysis/ process patterns vs SM
22. Aspects vs SM
23. KRs in AI vs SM
24. More Stability patterns
25. The modeling of SM with different modeling language such as UML, formal methods, OMT, fusion, RA/SA
26. Measurement of SM within the software
27. The impact of SM on Scalability
28. The impact of SM on Layering/ categorizing
29. SM framework
30. In SM, where are the hooks and at what level should be?
31. The impact of SM on Prevention and security
32. The impact of SM on Multiple view
33. The impact of SM on the cost and expending of your system or your operation.
34. Criteria for SW stability in Analysis
35. Criteria for SW stability in Design
36. Criteria for SW stability in Coding and Debugging
37. Criteria for SW stability in Testing
38. SM and representation adequacies, such as:

**Descriptive Adequacy**

Descriptive adequacy refers to the ability to visualize objects in the models. Every defined object should be browse-able, allowing the user to view the structure of an object and its state at a particular point in time. This requires understanding and extracting meta-data about objects that will be used to build a visual model of objects and their configurations. This visual model is domain dependent -- that is, based on domain data and objects’ meta-data. Descriptive adequacy requires that all of the knowledge representation is visual, as follows:

- Visual models are structured to reflect natural structure of objects and their configurations
- All the visual knowledge (data & operations) in the visual model is localized
- Relationships among objects in the visual model are well-defined
- Interactions among objects in the visual model are limited and concise
- The visual model must transcend objects, and instead highlight crosscutting aspects.

**Logical Adequacy**

Logical adequacy refers to the representation tools that describe the framework components’ behavior, roles and responsibilities.

**Synthesis Adequacy**

Synthesis adequacy refers to an integrated problem resolution methodology, or built in trouble-shooting tools. Built-in trouble-shooting tools are very important in managing complex distributed systems, because there are typically many potential points of failure.

**Analysis adequacy**

Analysis adequacy refers to integrated validation and verification tools. With built in validation and verification, the process of maintenance and regression testing can be streamlined and the cost of validation and verification is minimized.

**Blueprint Adequacy**

Blueprint adequacy refers to the modeling features that provide for integrated system specifications. Integrated system specifications are important because they facilitate the extensibility of the system. An integrated blueprint for an enterprise framework should clearly identify the hot-spots and frozen-spots in the framework.

**Epistemological Adequacy**

Epistemological adequacy refers to tools for representing objects in the real world. There are two ways to view the world based on simplicity: (1) Perfect but simple view – the world is represented in this view as an ideal environment and (2) As-is, but complex and detailed view – the world is represented as an ultimate reality. There are also two ways to view an organization: 1) Flat and single view and 2) layered and multiple views. It is very obvious that most of modeling techniques, such as unified modeling language (UML) and object-modeling technique (OMT) model the world as an ideal environment and flat or single view of itself. Nevertheless,
successful enterprise frameworks have made great leaps in representing objects in the real world and in providing the necessary tools to alter these objects as required by the business.

**Notational Adequacy**

Notational adequacy refers to the presentation constructs the impact the presentation tools have on the operation of the system as well as the ease of modification.

**Procedural Adequacy**

Procedural adequacy refers to recognition, and search capabilities.

**Contractual Adequacy**

Contractual adequacy refers to the client tools for representing the system behavior.

**Scalable Adequacy**

Scalable adequacy relates to the constructs and tools supporting partitioning, composition, security, and access control.

**Administrative adequacy**

Administrative adequacy refers to the tools for modeling the deployed system’s performance, reliability and administrative characteristics and to the actual tools for administering the system. Administrative adequacy also considers the availability of install set builders, start and stop procedures or scripts, integrated database management capabilities, archiving, fail-over mechanisms, etc.

**Other topics (required my approval):**

1. Best and economical ways to document your software design
2. Adaptability by exemplars
3. Aspect-oriented design with examples
4. Multi-agent design with examples
5. Model-based software reuse
6. Models defining the assets to be reused.
7. Models describing requirements, models of the architecture of composed systems
8. Determining and modeling features on the requirements level
9. Composition and interaction problems in research and industry.
10. Theoretical foundation of modeling and asset inconsistency detection and specification.
11. Methods to assure certain qualities of service after composition
12. Modeling of assets and their desired relationships
13. Detection of undesired negative asset interaction
14. Conformance between model and implementation / reality and corresponding consequences for reasoning about the implementation / reality.