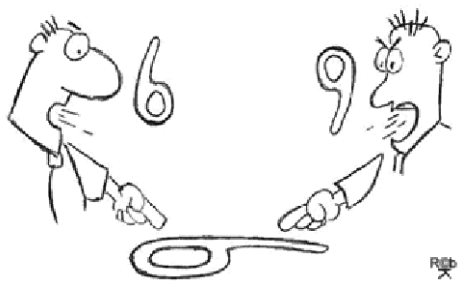
Software Intensive Systems Modeling
A Methodological Approach for MBD

Dr. Amir Tomer, PMP, CSEP, CSQE
Head, Software Engineering Department
Kinneret Academic College on the Sea of Galilee, Jordan Valley, Israel
amir@amirtomer.com

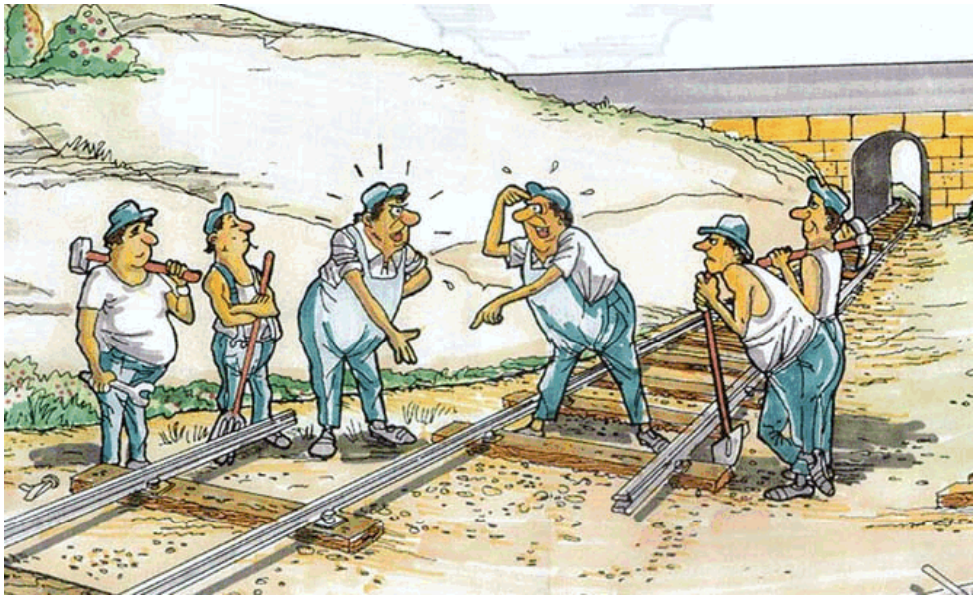
Motivation

- The quality of a product depends much on the quality of its requirement and design specifications
  - Missing, vague or sloppy specifications lead to wrong implementation, inaccurate testing and poor product usability
- Modeling techniques and languages are increasingly used for software engineering, replacing textual specifications with visual ones
- However…
- In many software-intensive systems, modeling is not applied properly, according to
  - Lack of modeling methodology
  - Complexity of languages and tools
  - Inconsistencies between development levels
    - Particularly between the system and the software level

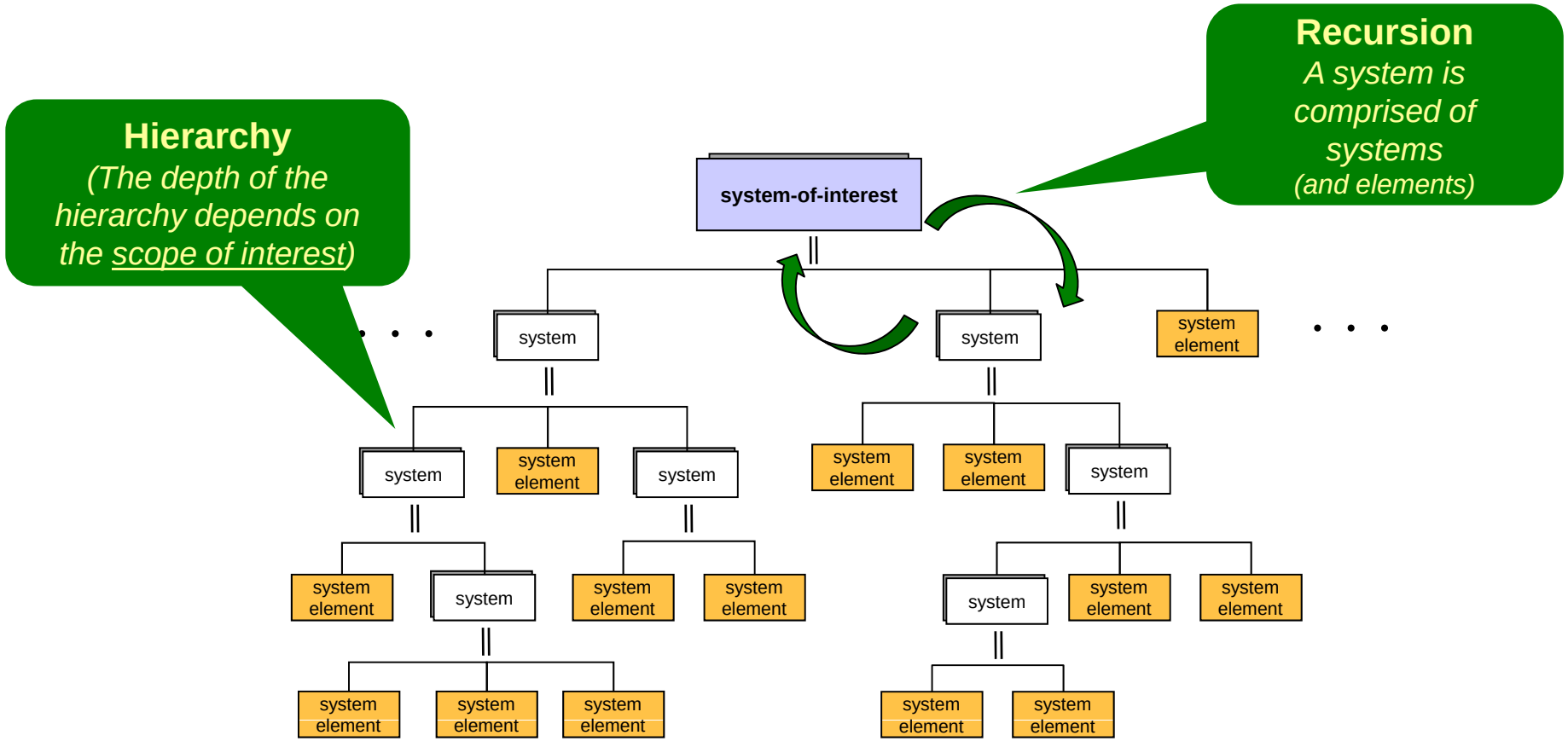


Our Approach

- We propose a methodological approach to software-intensive system modeling, which provides
  - Systematic modeling for all system breakdown levels
  - Consistency within and between system and software level models
  - Simple and “just enough” application of modeling elements (UML/SysML)



“System” – a recursive-hierarchical Structure*



*ISO/IEC/IEEE 15288

Properties of a System

- System – Definition*
  - combination of interacting elements organized to achieve one or more stated purposes
    - Thus, each system has the following properties
      - Purpose(s)
      - Elements
      - Interaction (among its elements)
      - Organization (over its elements)
- System Element – Definition*
  - member of a set of elements that constitutes a system
    - Thus, according to the recursive-hierarchical structure, a system element may be either
      - A system by itself – possessing all system properties
      - An elementary (atomic) entity – possessing just purpose(s)

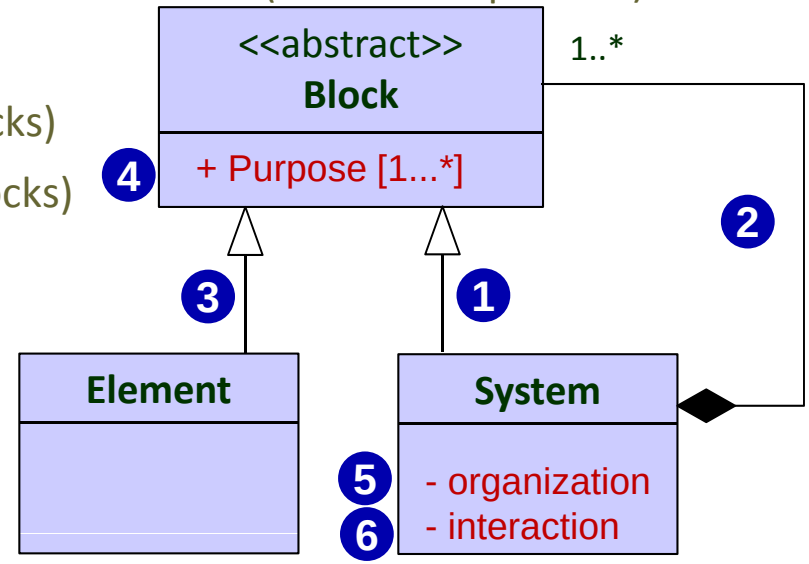
*ISO/IEC/IEEE 15288

“Block” – a unified notion

- In order to obtain unified modeling concepts for systems and elements at all levels
  - i.e. System-of-system, system, subsystem, assembly, component, unit, etc.

we define a unified entity, noted as “Block”, as follows:

1. A system is a block
2. A system is composed of one or more blocks
3. An element (system element) is a block, which is atomic (non-decomposable)
4. Every block has one or more purposes
5. A system has an organization (over its blocks)
6. A system has an interaction (among its blocks)



Based on the “composition” design pattern: Vlissingen et al, *Design Patterns*, 1994

The Four Views of a Block (System/Element)

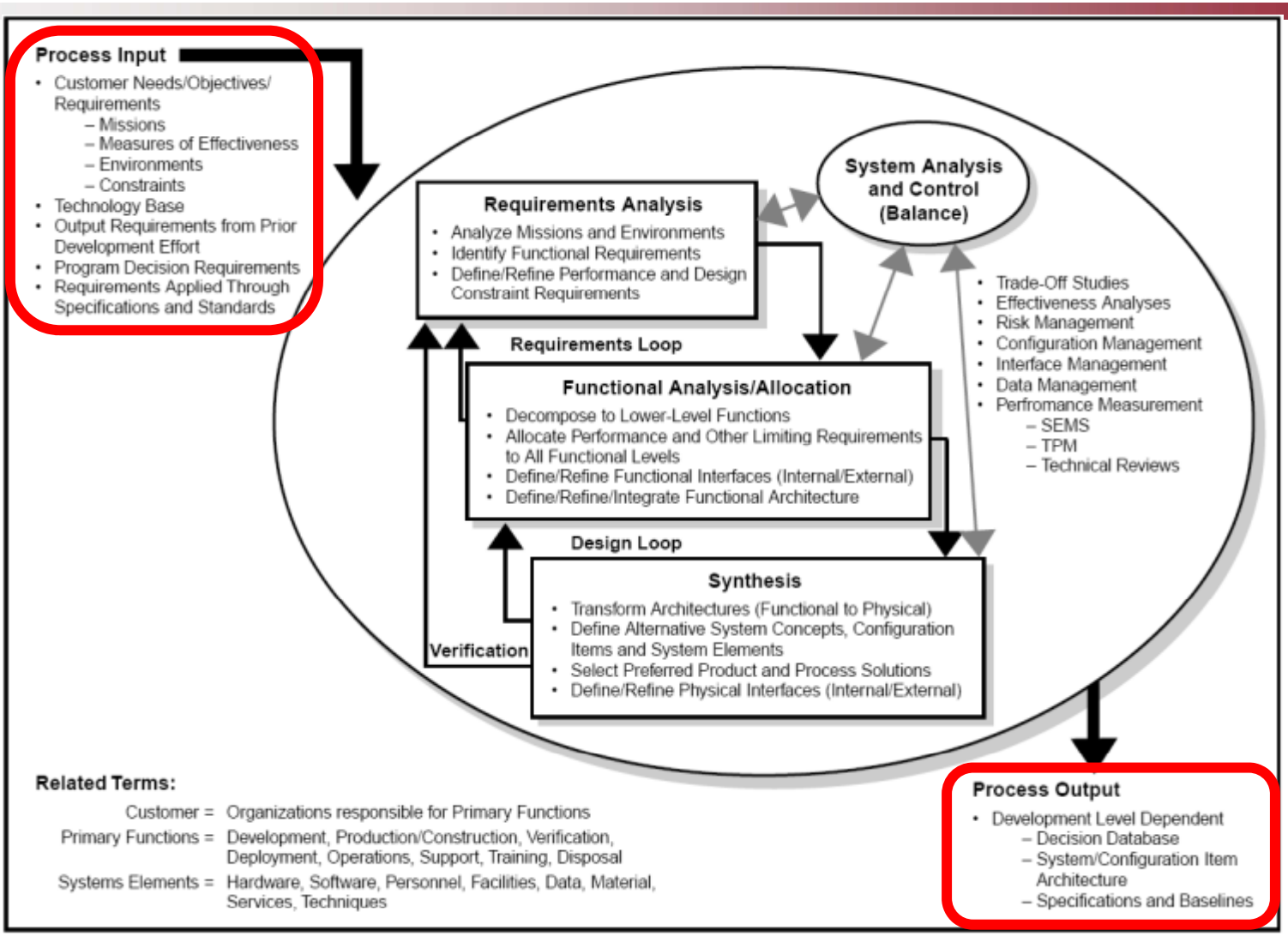
	Static Viewpoint	Dynamic Viewpoint
External Viewpoint (Black Box)	**Environment** The connection points (interfaces) between the block and its external entities	**Services (Functions)** The provided/acquired services(functions) to/from external entities and the way these services are obtained
Internal Viewpoint (White Box)	**Structure** (non-atomic blocks only) The elements comprising the block and their interrelations	**Behavior** The activities the block needs to perform in order to provide its services

Organization

purpose

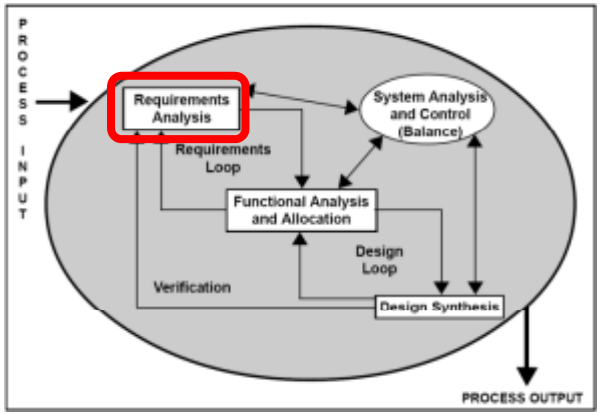
Interaction

A Systematic Systems Engineering Process (SEP)*



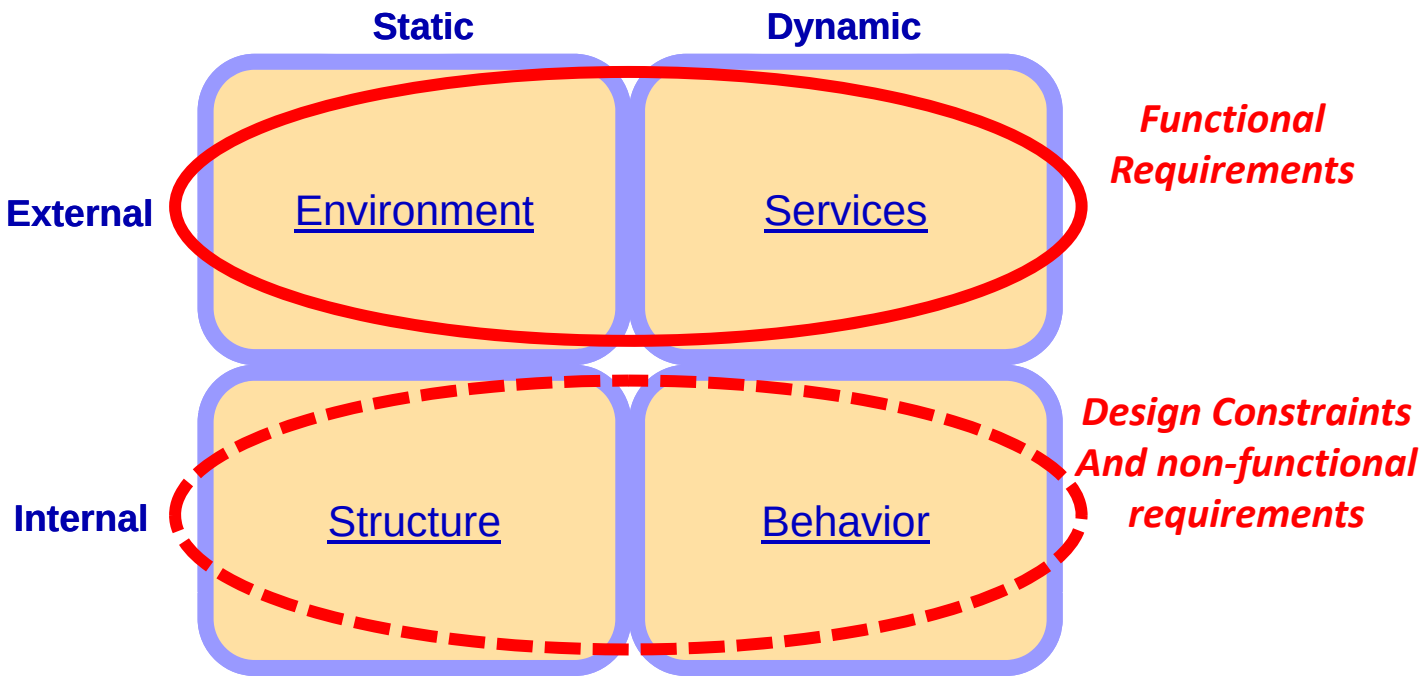
** Systems Engineering Fundamentals, DOD, 2001 and IEEE-Std-1220*

Modeling in the Requirements Analysis Phase

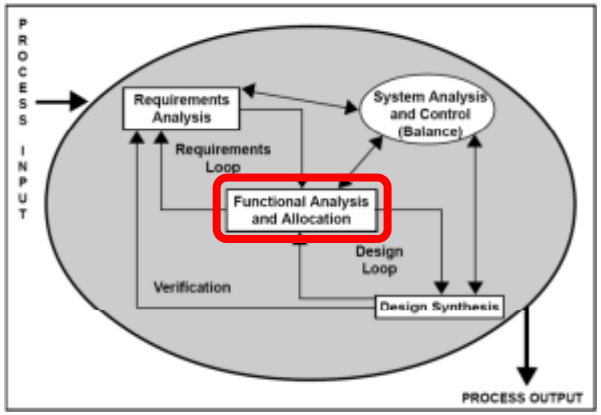


Requirements Analysis

- Analyze **Missions** and **Environments**
- Identify **Functional Requirements**
- Define/Refine **Performance** and **Design Constraint Requirements**

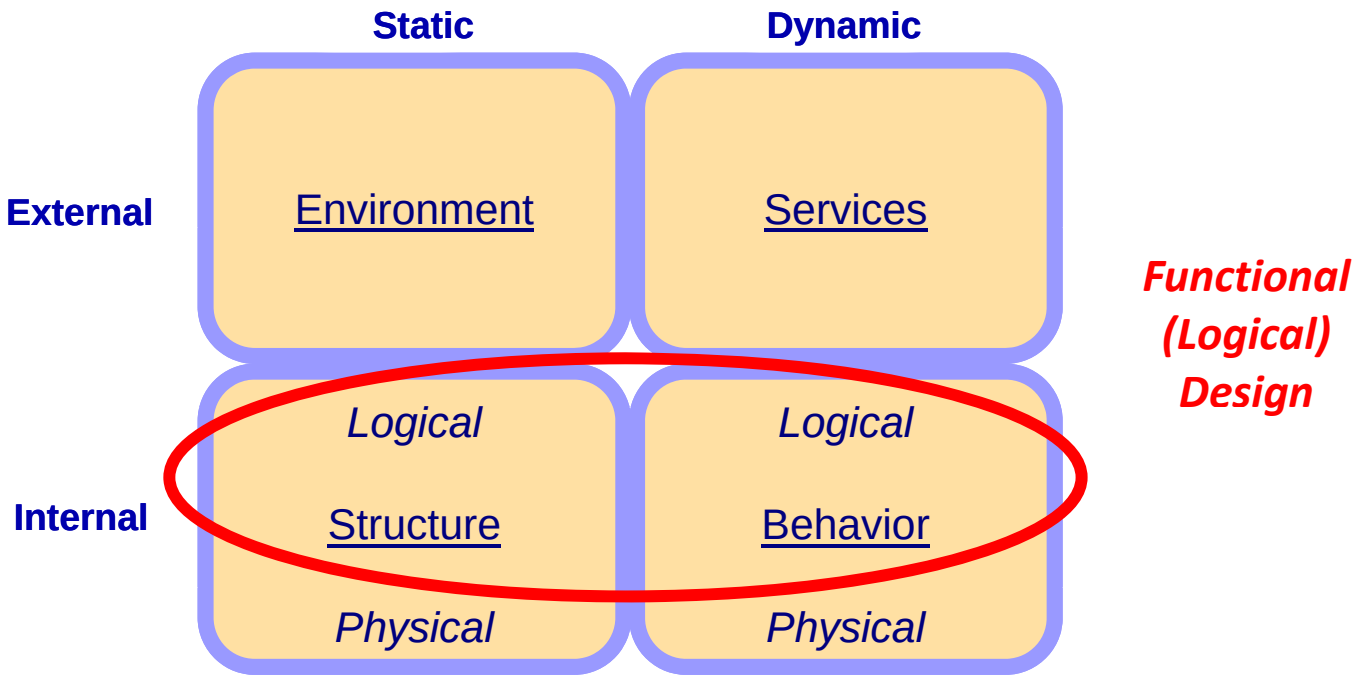


Modeling in the Functional Analysis Phase

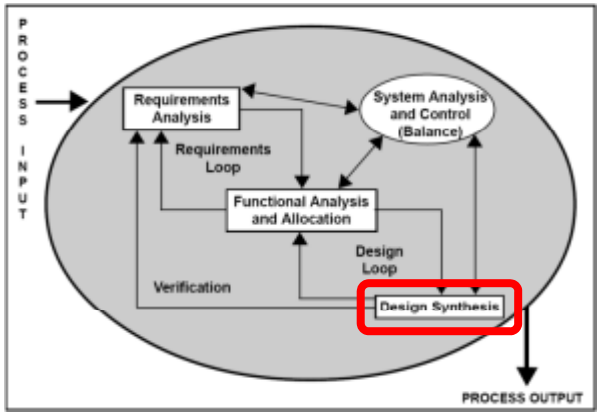


Functional Analysis

- Decompose to **Lower-Level Functions**
- Allocate Performance and Other Limiting Requirements to All Functional Levels
- Define/Refine **Functional Interfaces** (Internal/External)
- Define/Refine/Integrate **Functional Architecture**



Modeling in the Design Synthesis Phase



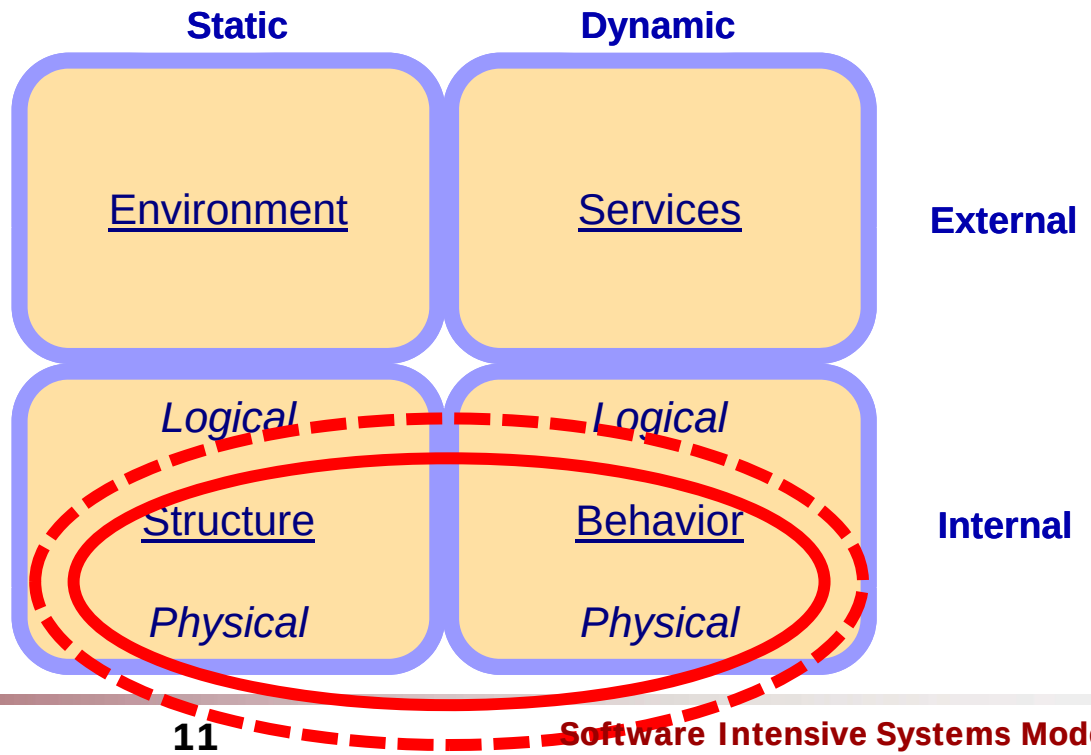
Design Synthesis

- Transform **Architectures** (Functional to Physical)
- Define Alternative System Concepts, Configuration Items and System Elements
- Select Preferred Product and Process Solutions
- Define/Refine **Physical Interfaces** (Internal/External)

Physical Design (Architecture)

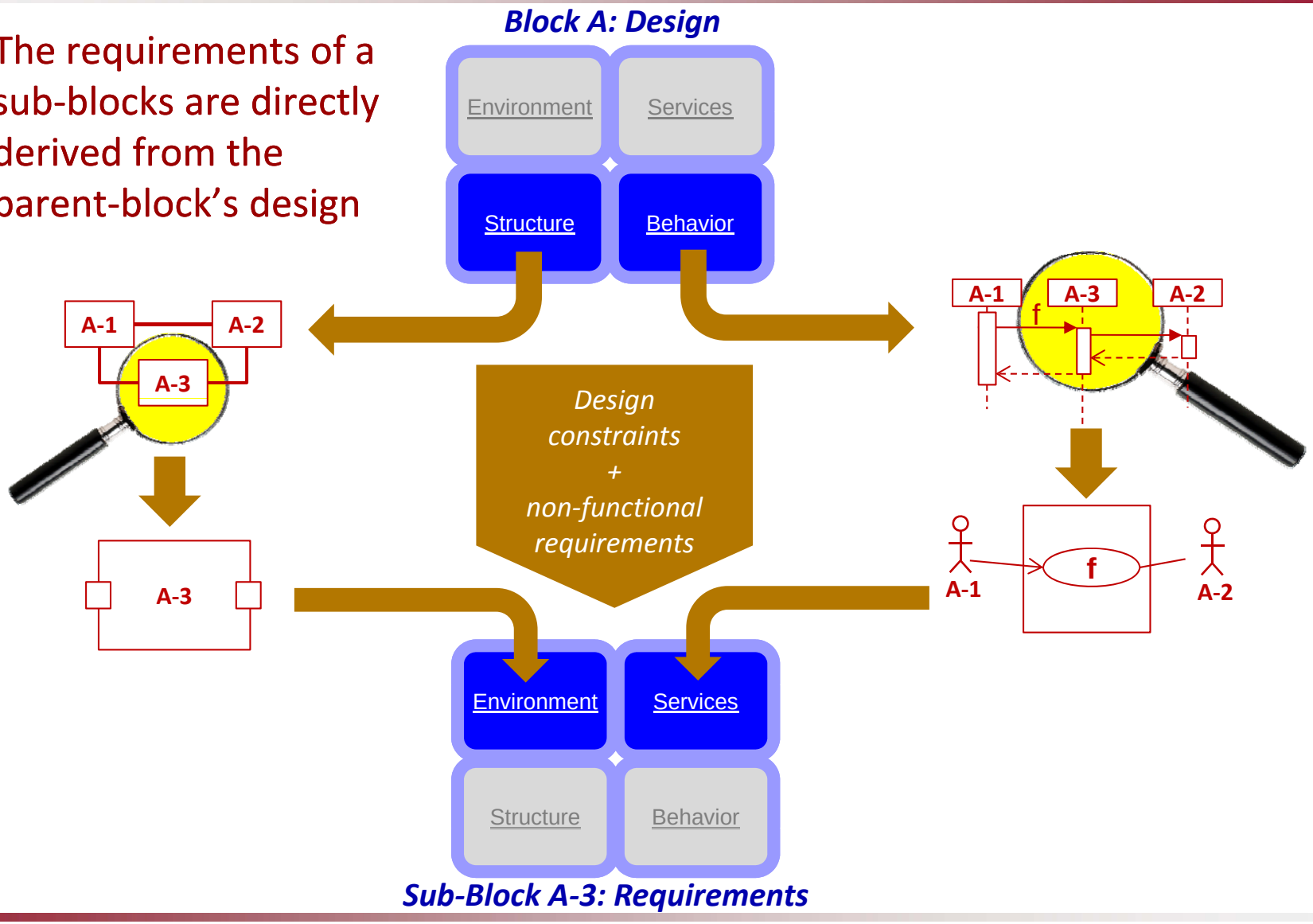
Subject to design constraints

And other non-functional requirements



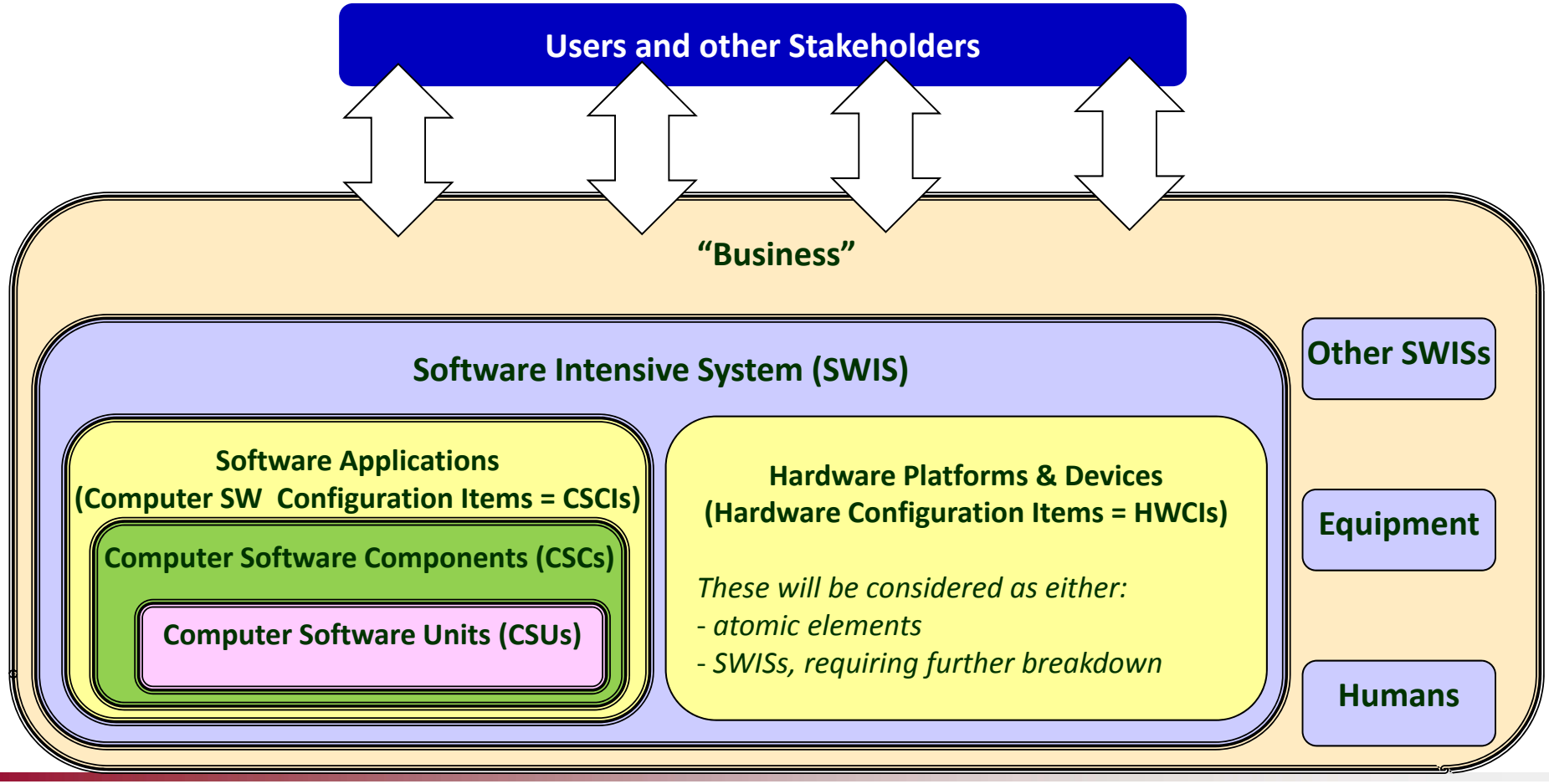
From Block to Sub-Block Modeling

- The requirements of a sub-blocks are directly derived from the parent-block's design



The “Five Levels of Interest” of a Software-Intensive System (SWIS)

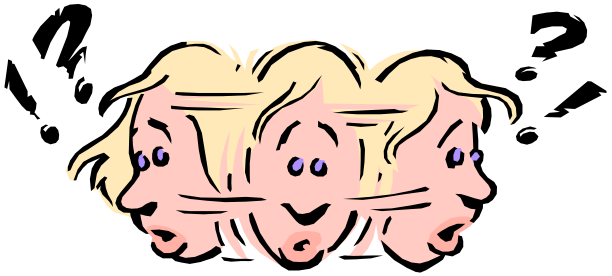
- The general definition of a “system” allows unlimited depth of hierarchical breakdown
  - Although this is applicable also for SWISs, there are 5 types of levels, for which certain model types are preferred for the sake of modeling effectiveness



Related Topics and Issues

- Choosing the right model for each view
  - The use-case/state-machine/activity-diagram dillema
- Model consistency
  - Between models on the same breakdown level
  - Between breakdown levels
- Model verification
  - Correctness, quality, applicability etc.
- Life-cycle architecture modeling
  - Test architecture, production architecture, maintenance architecture, …
- Balancing visual and textual specifications
  - One picture vs. 1000 words
- Non-top-down engineering modeling (re-engineering, reverse engineering, …)
  - Open-source style, unsynchronized system arrays

Thank you!



Any questions?

amir@amirtomer.com
052-8890202