Complexity-Awareness in the Design of Safety Critical and High Integrity Systems

Dr. Carl Elks, Aidan Collins
DeCyPS Lab
Department of ECE
Virginia Commonwealth University

VCU College of Engineering

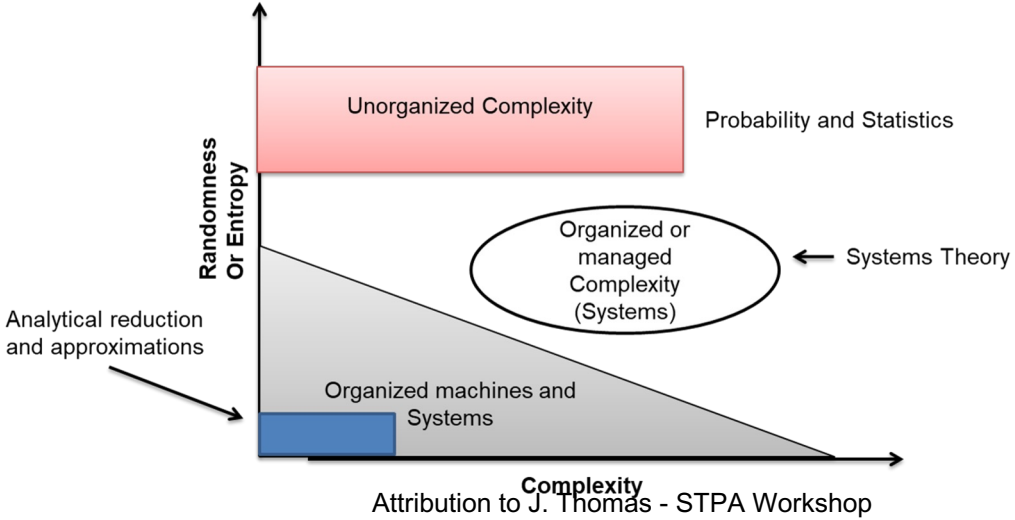
Part 1: Why is it Important

Complex Systems? A Systems Perspective

Modern CPSs involve complex interactions between many components:

- *Software, hardware, human operators, environment, management, maintenance etc.*
- *Interactions can be overlooked*
- *Need to understand the whole system of interactions*
- *Complexity of interactions leads to unexpected emergent system behavior*

- *Systems Theory, Formal Reasoning, System Based Hazards Assessment (e.g. STPA, HARA), Cognitive Science help "manage" safety and risk of Complex Systems - to a degree.*



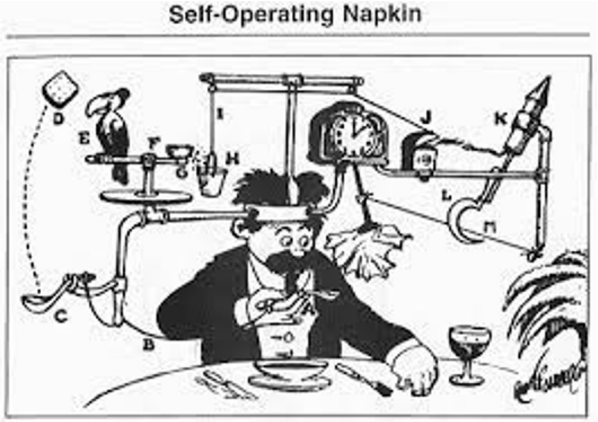
Attribution to J. Thomas - STPA Workshop

Perhaps the greatest impact of complexity is that it erodes the trust we place upon these systems

Complexity and Simplicity:Do we need Complex Systems

- One of the most quoted heuristic guidelines in system design is: “KISS” – “Keep it Simple and Sweet”.
- This idiom expresses an engineer’s intuition that a less complex design is a better design.
- The simpler the system is, the easier it is to design, verify, implement, and maintain.
- Especially important for safety and security critical systems.
- BUT, recently we seem to be trending away from the KISS idiom. Some good reasons:
  - Consumer driven - Highly connected and mobile world
  - Public need - Smart Grid, Electric Transportation, Smart Cities, etc..
  - Age of algorithms - Miraculous black box data driven systems (AI and ML).

Rube Goldberg machine - definitely NOT KISS



“Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.” Edsger Dijkstra

Encounters with Complexity

2013 Ford Fusion

"*We had a sequence of events that caused the cooling system software to restrict coolant flow," he says. Most of the time, that would not be a problem and is the intended behavior. But in rare cases the coolant pressure coupled with other conditions may cause the coolant to boil."*

Overlooked System interactions and contexts.

BOEING 737 Max MCAS

Because the MCAS system was initially only expected to be needed in cruise flight, its limit was set at 0.6 degrees and its DO-178C criticality was set at DAL-C (Major), rather than DAL-B (Hazardous) or DAL-A (Catastrophic).When it was realised that the MCAS would also be needed in slow-speed flight and its limit was increased to 2.5 degrees, no change was made to the DAL.

Communications of the ACM, January 2021, Vol. 64 No. 1, Pages 22-25

Philip Koopman on Toyota Camry Acceleration Incidents

There are a large number of functions that are overly complex. By the standard industry metrics some of them are untestable, meaning that it is so complicated a recipe that there is no way to develop a reliable test suite or test methodology to test all the possible things that can happen in it. Some of them are even so complex that they are what is called unmaintainable... [3]

VCU College of Engineering

Complexity is not a goal of engineering. It emerges as a consequence of the increasing sophistication of technology.

Benefits of “Taming” Complexity

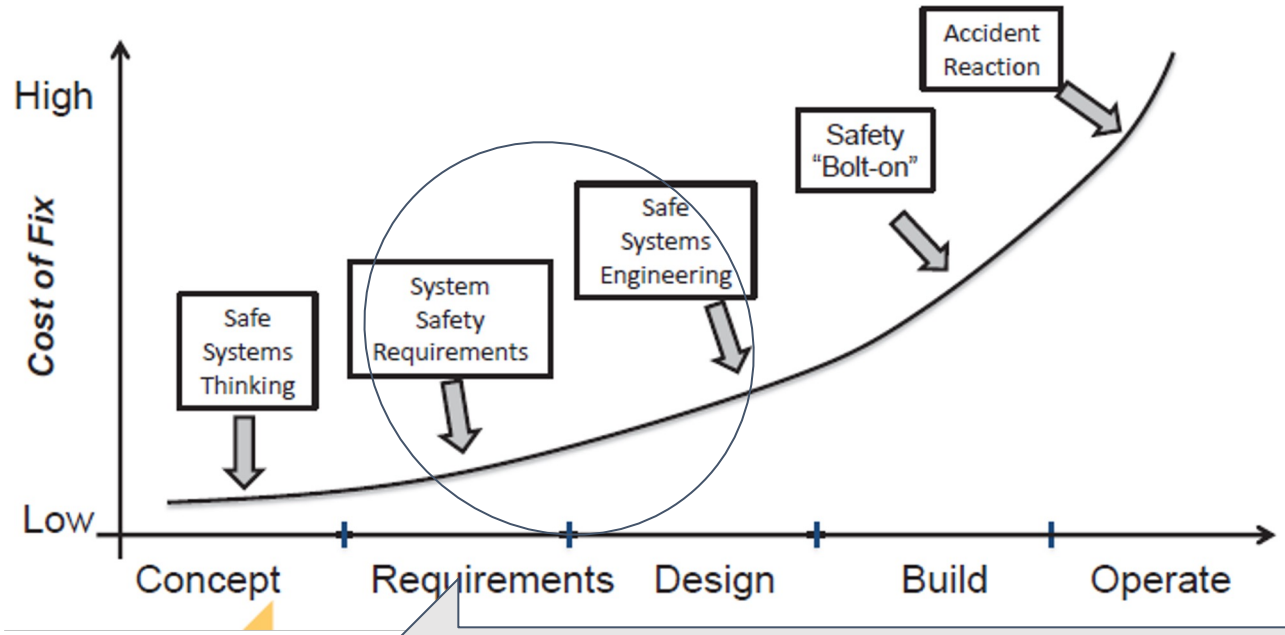


Figure attributed to J. Thomas - Risk Management of Complex Systems..

Sounds Good, but what is lacking ?

Our interest for exploring “complexity awareness” was motivated by the SymPle Project.

- SymPLe project - Design and Implementation of a highly verifiable FPGA-Based platform for critical Nuclear Power safety protection applications[1].
- We wanted to understand architectural and interaction complexity with respect to requirements/design.
- However, methods like STPA, design patterns, SW complexity metrics, etc seem to gloss over complexity in its many forms.
  - STPA is not design oriented.
- Systems theory is helpful, but tended to be too abstract for design oriented activities.
- So, we went exploring.

1-Hite, R., Rajagopala, A., Gautham, S., Deloglos, C., Jayakumar, A., Collins, A., Elks, C. and Gibson, M., 2021, June. SymPLe: Complexity-Aware Design for Safety Critical I&C Systems. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks- (DSN-S)* (pp. 53-56). IEEE.

Begin to Think of “Complexity Awareness”as a Dimension of Design…..

- Design is mainly concerned with achieving functionality as constrained by “other” requirements -safety, security, reliability, etc..
- Goal: make complexity awareness a dimension of design
- Transition from something that must be dealt with to something that can be understood and managed
- If complexity is an impediment to V&V and certification of I&C systems, we need to characterize it and understand it.
- The “What is it”.

Our working Definition - Complexity-Awareness is not a prescriptive process, rather it is a way of thinking about design with a technical basis, workflow, rules, and metrics – that eases verification and produces strong evidence of assuredness. It has an objective.

Part 2: What is it?

What is Complexity?

INCOSE Definition

*A system is said to be **complex** when its structures cannot be described at a single level or with a single view; multi-scale descriptions are needed to understand the systems. Complex systems have emergent behavior, derived from the relationships among their elements and with the environment, via internal and external feedback loops, giving rise to observed patterns that may not be understood or predicted.*

- The concept of complexity has been studied broadly in a variety of disciplines such as computer science, systems engineering, and information theory.
- **Treating complexity as one monolithic property is not very useful in creating methods and tools to manage it.**

Kopetz:Types of Complexity

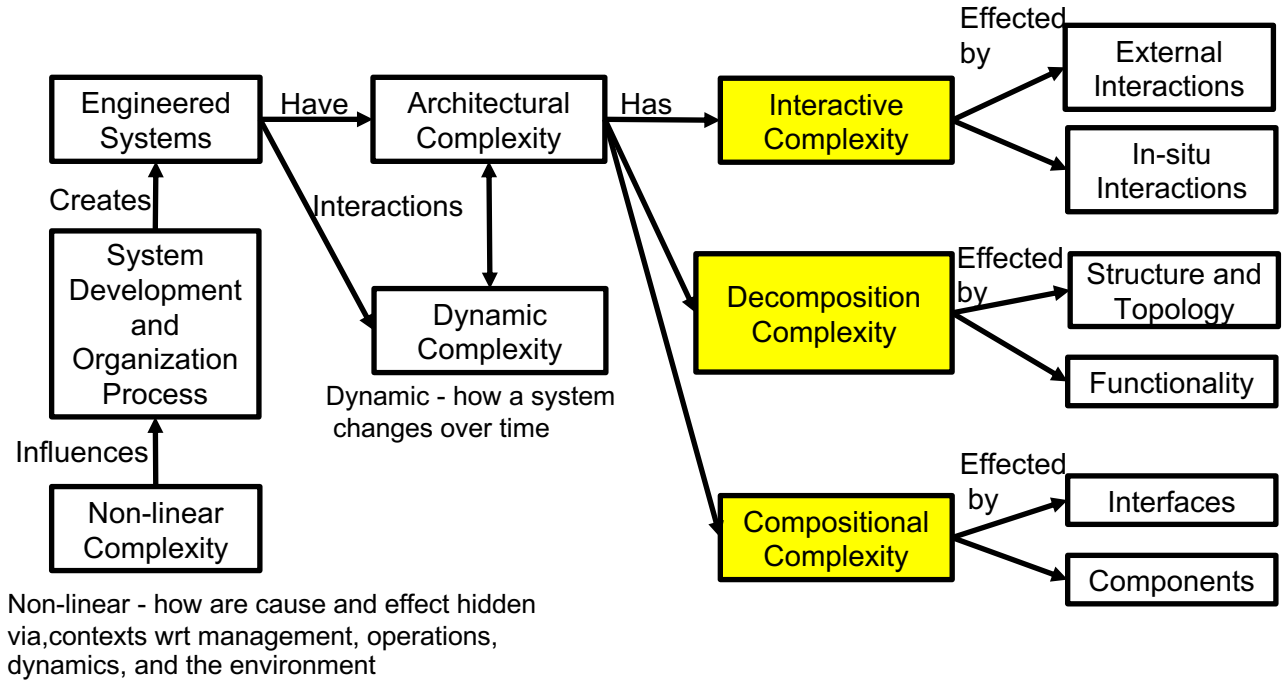
Kopetz & Efatmaneshnik

- Cognitive/Subjective Complexity
  - how difficult a system is to understand by an observer
  - Distance between observer's reference model and the system
- **Object/Objective Complexity**
  - a quality of the system's components and interactions in a given context
  - Quantity and variety of components, interfaces, communication, and other links comprising the system,
  - The **system architecture (HW/SW)** contributes to the overall complexity of the system.
  - Design phases and implementation phases are opportunities to constrain it for a benefit.

Kopetz, H., and Simplicity Is Complex. "Foundations of Cyber-physical System Design." (2019).

Leveson:Classification of Complexity Types

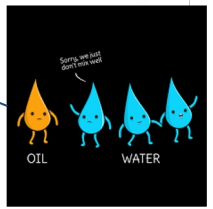
- Architectural - how a system is constructed and organized
- Important that we select the complexity types relevant to architectural complexity
- This diagram depicts the relations between Leveson's complexity types
- Complexity types in Yellow were the types selected because they collectively influence architectural complexity



Initial Complexity Awareness Principles

Initial selection of principles was inspired by Leveson, Kopetz, and Maier [1-3].

- **Principle of Orthogonality** - A set of basic building blocks can be combined in a relatively small number of ways to build functionality.
- **Principle of Hierarchy** - Manage and limit interactions between system layers
- **Principle of Separation of Concerns** - Functions that are separable are grouped in self-contained modules
- **Principle of Consistent Time and Determinism** - Consistent time allows agreement and consensus of events
- **Principle of Modularity** - A module encapsulates a function or set of functions related to a system requirement.
- **Principle of Observability** - Non-visible communication channels or states among architectural units pose a severe impediment to the understanding of system behavior.
- **Principle of Independence** - Interdependence between modules or architectural units at one level of the hierarchy should be reduced to the necessary minimum
- **Principle of Parsimony (Ockham's razor) -** states that whenever different models can explain a set of observations the model that relies upon, the smallest set of assumptions is preferred.



[1] *N.* G. Leveson, "Complexity and safety," in Complex Systems Design & Management. Springer, 2012, pp. 27–39.

[2] H. Kopetz,Simplicity is complex. Springer, 2019.

[3] M. Maier, "Dimensions of complexity other than 'complexity'," in Symposium on Complex Systems Engineering, 2007, pp. 11–12.

VCU

Measuring Facets of Complexity

Goal: Identify/Develop objective metrics related to principles which provide tangible benefit to stakeholders

Challenge: How to capture presence of complex attributes?

- Size, number of interactions, interface complexity
- Coupling and Cohesion
- Random/ad-hoc vs hierarchical/modular structure

Points of Interest

- What do existing metrics measure?
- How can these metrics be applied to design?
- How are principles accounted for in metrics?
- Given a principle, how do you design a metric for it?

General Attributes of Existing Metrics

- Number of:
  - LOC
  - Components
  - Interactions
  - Operators
  - Operands
  - Data types
  - etc.
- Fan-in/Fan-out - number of ingoing/outgoing edges to a block/module/subsystem
- Shannon Information/Entropy

Common Metrics

- Halstead
  - Measures size, volume, and difficulty of code
  - Can be translated to FBD
- McCabe (Cyclomatic Complexity)
  - Counts decision points in code
  - $C = E - N + 2$
- Information Flow
  - $IF(i) = Size(i) * (fanin(i) * fanout(i))^2$
- Dincel
  - Required Service Utilization
    - $RSU_x = R_{actual} / R_{total}$
  - Provided Service Utilization
    - $PSU_x = P_{actual} / P_{total}$
- Shannon Entropy
  - Information - Bits represented by an event
    - $I(x_i) = -\log_2(p(x))$
  - Entropy - Expected number of bits for a set of events
    - $H(X) = -\Sigma_{x \in X} p(x) \log_2(p(x))$

Component-Level Metric: Card and Glass

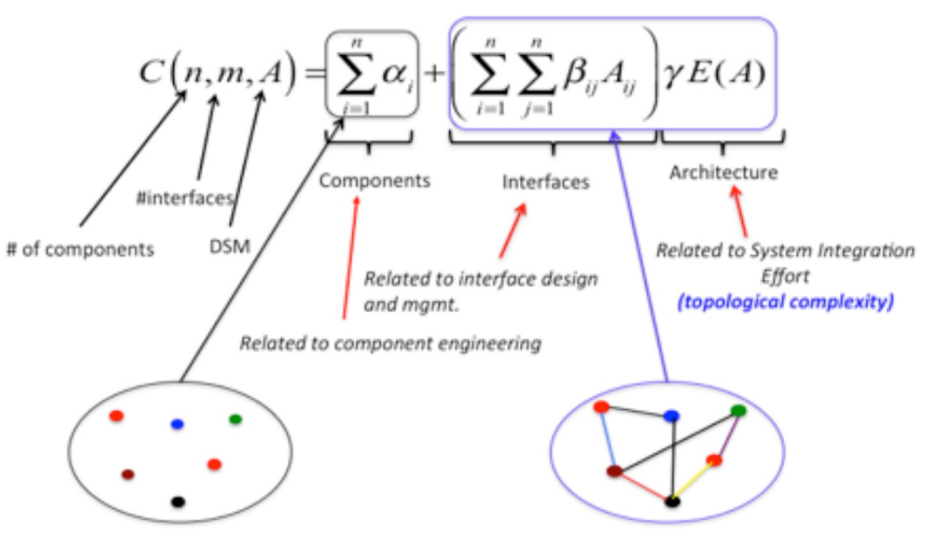
Structural & Data Complexity (Card and Glass)

- Structural Complexity - intends to measure coupling
  - $Cs(i) = fanout(i)^2$
  - Focal point is how many subsystems are invoked by the subsystem
- Data Complexity - intends to measure cohesion
  - $Cd(i) = IOCnt(i) / (fanout(i) + 1)$
  - Suggests that more IO/fewer subsystem invocations means that it has more responsibilities/complexity
- Total Complexity
  - $Ct(i) = Cs(i) + Cd(i)$

System-Level Metric: Sinha's Structural Complexity

[Sinha 2014] Structural Complexity (EPRI's DEG methodology has this flavor)

- Derived from energy calculation in molecular physics
- Key points are
  - (1) atoms/nodes
  - (2) bonds/edges
  - (3) structure
- Notes
  - E(A) is the sum of the singular values of graph A
  - γ = 1/n scales the global connectivity
  - Up to user to assign component and interface complexities



Comparative Analyses

[Basili 1996] Applying OO metrics to Software Defect Detection

- Tested a variety of metrics related to object oriented design
  - Coupling, Cohesion, Depth of Inheritance, LOC, etc.
- Applied metrics to several implementations of a similar entertainment system
- Some metrics were significantly correlated with presence of code defects
  - Weighted Methods Per Class
  - Depth of Inheritance Tree
  - Response for Class
  - Coupling between Objects
- Drawback is this becomes applicable after implementation

Comparative Analyses

[Hennig 2021] Comparison of metrics to design principles

- Sought to determine whether common design principles were represented in metrics
  - Belief 1: Complexity rises with size
  - Belief 2: Complexity rises with interactivity
  - Belief 3: Complexity decreases with modularity
- Analyzed 6 metrics against randomly generated graphs differing in (a) number of nodes, (b) number of edges, (c) degree of modularity
- Metrics agreed on Belief 2, but not the others
- **Points to needed effort designing metrics based on design principles**

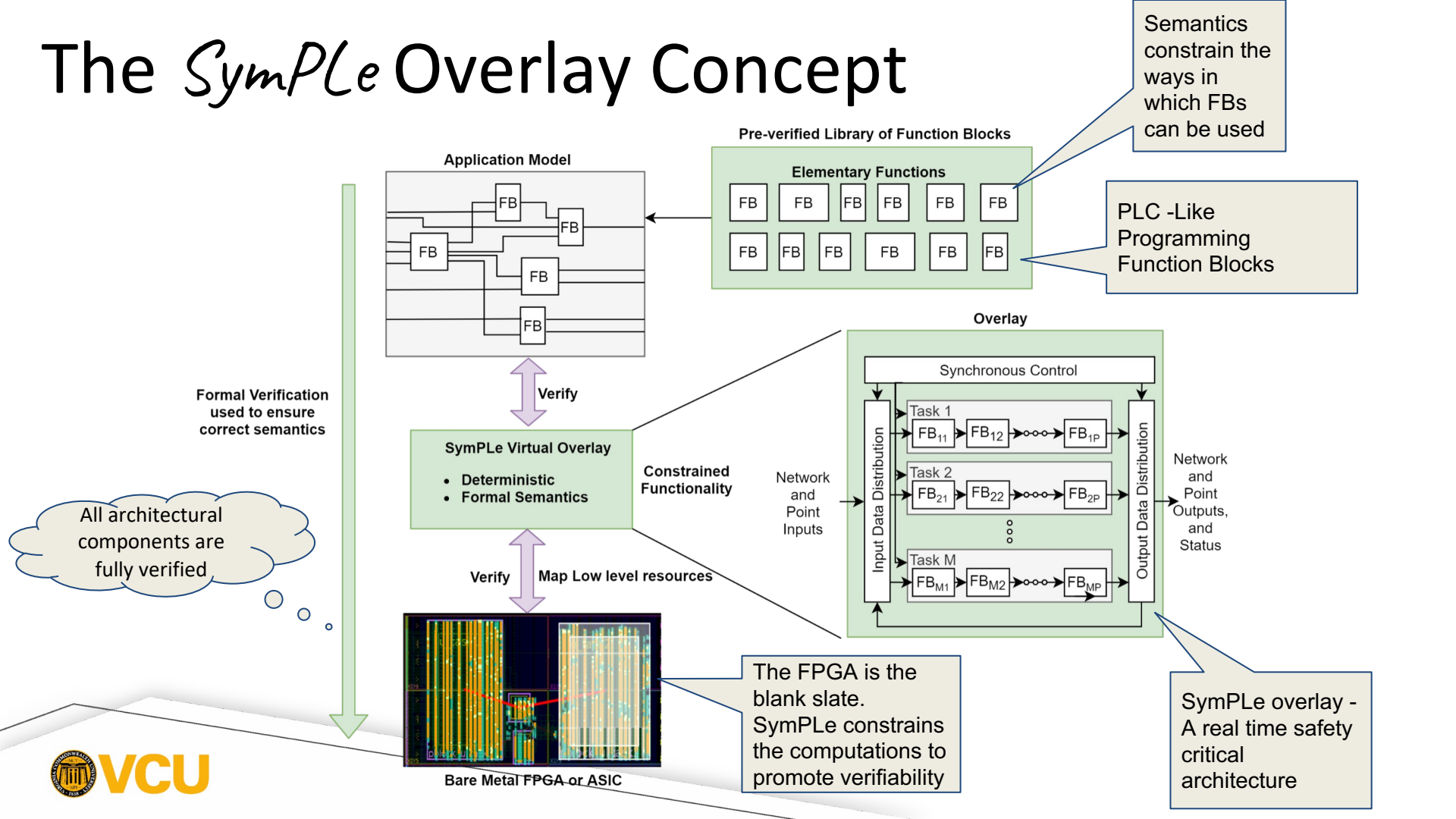
Focused Literature Review Wrap up

- Types of complexity well established
  - Subjective/Objective, Structural/Dynamic, etc.
- Principles and methods to limit it
  - Hierarchy, Modularity, separation of concerns, etc..
  - Design Patterns
- Not all principles have the desired effect
  - misuse problem
- Rules or guidelines or patterns to apply during design are lacking.
- Metrics make conformity to a principle actionable in design

- Metrics generally measure things like:
  - Number of components
  - Number/difficulty of interfaces
  - Entropy
- Metrics can be component- or system-level
  - Structural & Data Complexity
  - Sinha's Graph Energy
- Metrics can be useful in locating issues, defects or problem areas in a design/code
- More work needs to be done to develop metrics for design oriented principles

Part 3: Our Application of Complexity Awareness Concepts

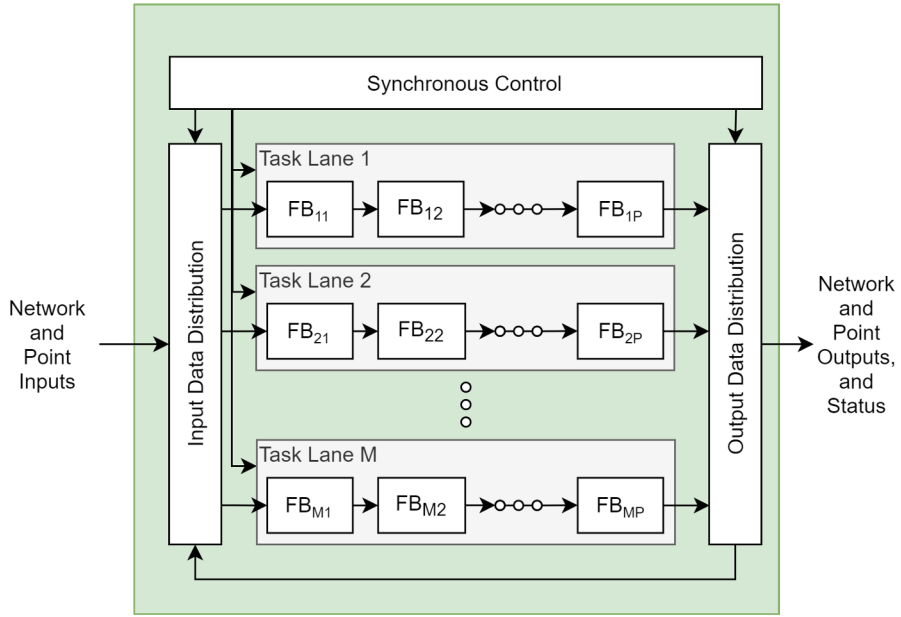
The SymPLe Overlay Concept



VCU

SymPLe Abstract Machine

- **To develop well-formed overlay**,we need a abstract computing machine.. An abstract machine is a specification - a formal executable specification.
- Defines what something is and what it is supposed to do, but not how it is realized.
- SymPLe Abstract Machine = Single computing platform performing multiple, independent computations interacting with a physical world
- **Task lane** = model of real-time computation
- **FB** = Function Block = Elemental unit of computation (not shared across task lanes).
- I/O Model - PLC scan cycle
- Used exactly the same as a Rate Monotonic Schedule except we have no interactions.



The *SymPLe* Architecture Manifesto

SymPLe is designed according to a philosophy, the SymPLe manifesto[1].

The basic tenets of the *SymPLe* manifesto are:

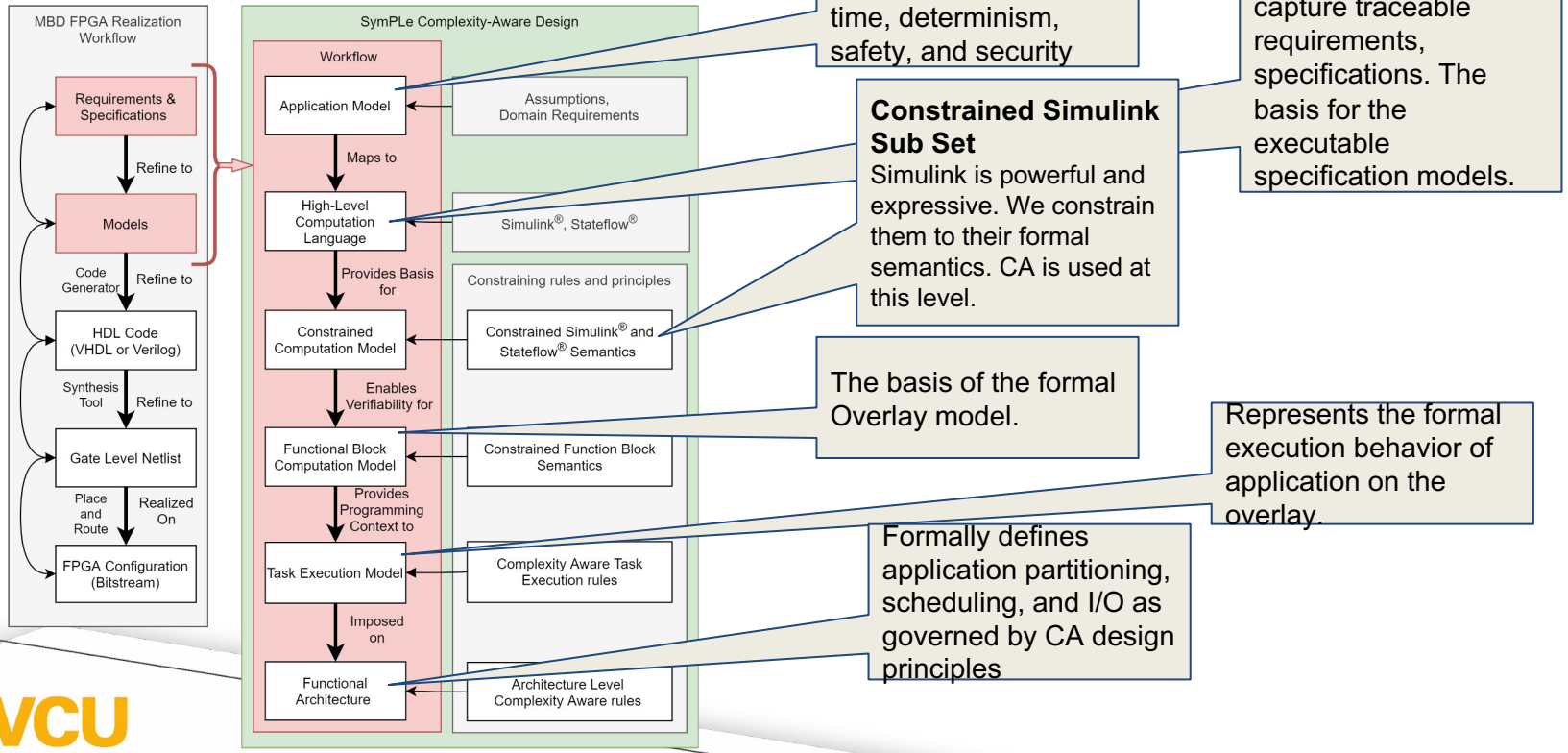
1. ***Complexity-Awareness***: During the design process, tradeoffs are made in favor of; (1) designs that minimize sophisticated structure and functionality, and (2) designs that are deterministic.
2. ***Verifiability***: Favor designs that reduce testing/measurements to reduce V&V efforts.
3. ***Foundation of Trust***: We favor designs where a few well-understood principles and established methods form: (1) a foundation for a well-composed design, (2) roots of trust for security and safety case arguments, and (3) limits of that trust.
4. ***Formal Models and Methods***: We favor designs that allow strong design assurance evidence to be collected. We employ "accessible" formal models and method, Model-Based Design and Engineering methods to provide chains of evidence from requirements to implementation.

These basic tenets serve as "guard rails" to keep complexity in check, maximize verifiability and comprehension.

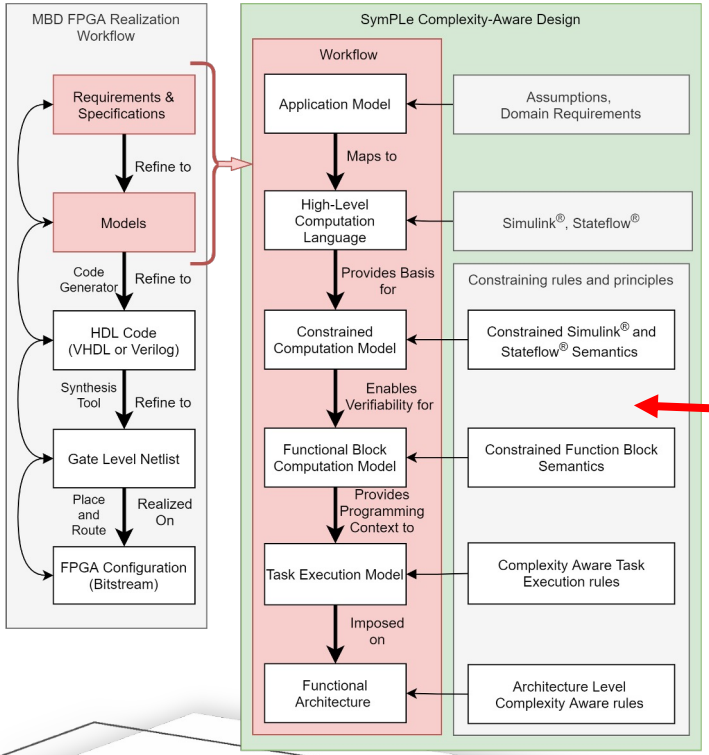
VCU

1 All good practicing luddites have a manifesto.

How Do We Integrate Principles in the Design Workflow?



From Complexity Principles Come Rules



Rules are used at various levels of Complexity-Aware Design

Just stick to the rules

CA Level	Rule	Purpose	Description	CA Principle
Language restrictions	State machine transitions 5	Restrict non-deterministic state machine behavior	Only one transition from current state to next state is allowed except as required. There shall be no more than two transitions from a state.	Testability
Language restrictions	State machine transitions 6	Restrict transition guard conditions	Transitions should be single conditions. This eliminates MC/DC coverage analysis requirements.	Testability
Computational model	Function block control 2	Partitioned control and data operations	Each function block is physically partitioned into components for the read, execute, and write phases.	Partitioning and independence
Architectural model	Scheduling 1	Scheduling is separated from all other functions	The scheduler function shall produce execution schedules for the task lanes and nothing else.	Decomposition and partitioning

VCU

Preliminary Results of CA Concept Efficacy

IEC 61508 Verification Summary

IEC 61508-3 V&V Requirements	V&V Metrics Achieved
Bidirectional Traceability Between Requirements and Model	
• Forward traceability between requirements and software: 100% • Backward traceability between requirements and software: 100%	100% requirement traceability (81% were low-level requirements,19% were high-level requirements)
Bidirectional Traceability Between Requirements, Test Cases (TC) and Formal Proofs (FP)	
• Forward traceability between requirements and TC: 100% • Backward traceability between requirements and TC: 100% • Forward traceability between safety requirements and FP: 100% • Backward traceability between safety requirements and FP: 100%	• 100% requirement traceability • 81% low level requirements were verified using TC and FP • 19% high level requirements were verified through reviews
Bidirectional Traceability Between Requirements, Model and Code	
• Forward traceability between requirements, model and code: 100% • Backward traceability between requirements, model and code: 100%	• 100% code to requirements traceability • 100% code to model traceability
Model Based Testing and Coverage Analysis	
• Structural test coverage (entry points, branches, statements: 100% • Structural test coverage (conditions, MC/DC): 100%	• 99% Decision Coverage • 100% Condition Coverage • 97% Execution Coverage • 99% MC/DC Coverage
Static Analysis	
0 failures required	60% passed, 40% warnings, 0 failures
Complexity Metrics Recommended	**Complexity Metrics Achieved**
Cyclomatic Complexity < 30 recommended	Achieved Cyclomatic Complexity of < 20 at unit level
Test Vector Reusability	• 84% of architecture supported reusability of unit test vectors. • 31 of 37 testable allowed reusability of unit test vectors.

Verifiability Metrics

SymPLe components	Cyclomatic Complexity		Test Objectives (MC/DC)		Test Cases	
	V2	V3	V2	V3	V2	V3
Global Sequencer	78	13	222	80	2	5
Local Sequencer	64	51	104	224	10	9
FBController	66	X	168	X	22	X
IO Manager	16	12	19	336	11	33
Bus Switch	21	X	58	X	8	X
Shared Memory	4	X	24	X	1	X
System Clock	X	17	X	48	X	2
Scheduler	X	11	X	31	X	2
Functional Blocks	1060	919	7230	4788	742	58
TOTAL	**1309**	**1023**	**7825**	**5507**	**796**	**109**

V2 = SymPLe version 2 = Partial Complexity Aware design
V3 = SymPLe version 3 = Full Complexity Aware design

86% reduction

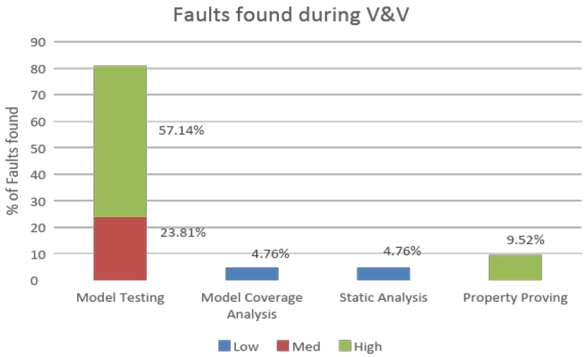
Model-Based Design and V&V: Finding bugs early. Formal Verification (model checking) tended to find very difficult bugs

84 % of the architecture supported Test case

Other findings that encourage more work.

Effectiveness of Complexity Awareness and Model-Based Design Assurance

- **81%** of the total count of design issues were found during **Model testing, 9.5%** were found during **Formal verification**.
- 66.7% of total faults were high severity faults during Model Testing and Formal Verification.
- Application of CA principles and Rules - Design Properties on a whole were easier to formulate and discharge (Model checking)
- **<u>Almost all design issues were found early at the model level before code generation.</u>**



Synergy between model testing and formal verification

- Testing helped **to identify vulnerable/weak areas in design** that could be a potential source for Hazardous design flaws. .
- These **critical areas** in the design could then be targeted for more **rigorous formal verification.**

VCU

Closing, Future Work and Open Questions

We believe Complexity Awareness as a design dimension has significant value wrt the certification of safety and security critical systems.

- That said, much work is needed by the community.

- Are our beliefs about complexity on solid footing?
- How do we define and adopt a “Characterization Model” of Complexity?

- Complexity theory seems sound and formal but too abstract for design.

- Current metrics seem to lack connection to complexity oriented design principles

- Comparative analysis of existing and new metrics - How to assess metrics.

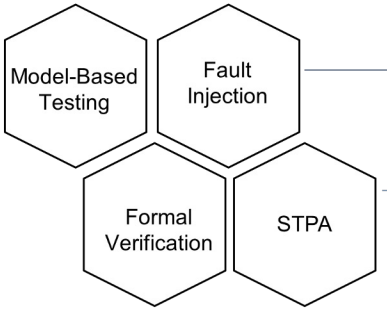
Questions?

Extras

Design Assurance Strategy

Elements of our Design Assurance Approach

- We followed **IEC 61508 standard at SIL 4 for evidence artifacts** and methods. Similar to DO-254.
- Advanced **model-based engineering and testing** - **first** for the nuclear industry
- **End-to-end verification** and **traceability** of requirements.
- Targeted Formal Verification with testing
- In Progress: **Vulnerability and hazard analysis** by STPA
- Design and V&V **metrics**, Complexity metrics to track progress and benefits.

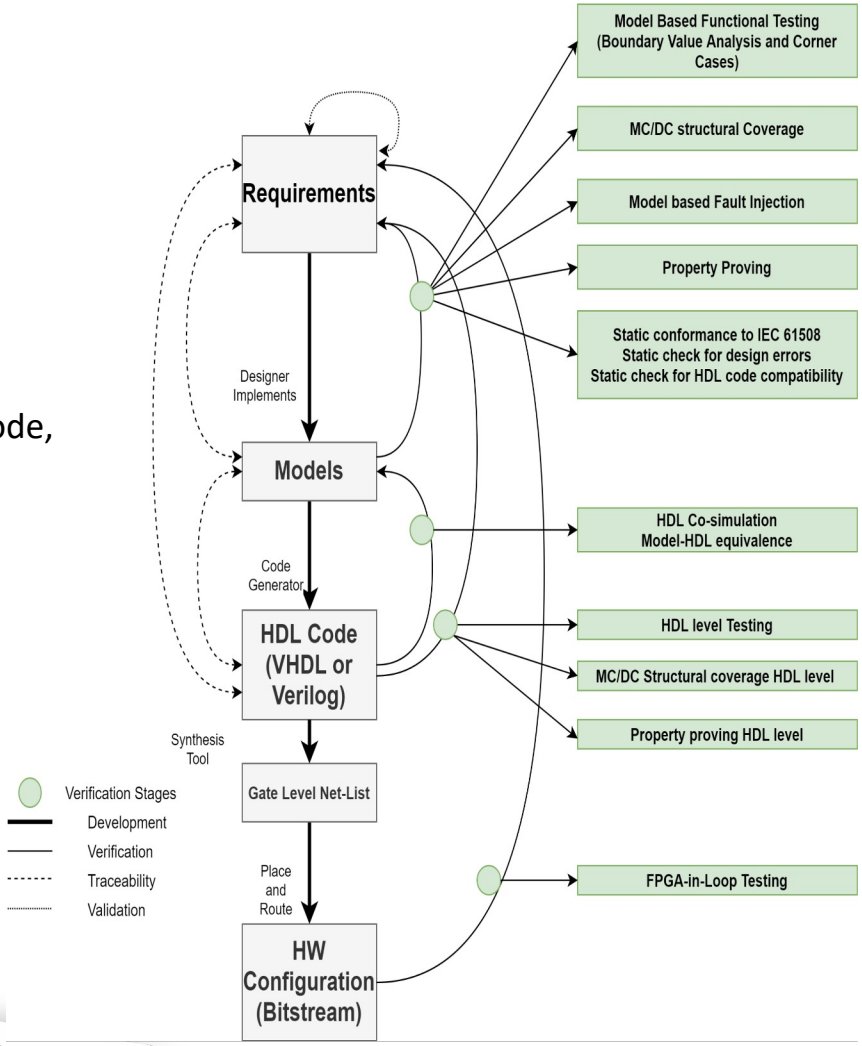


Model-Based Fault Injection Toolbox developed to perform: Strategic and Property Based Fault Injection

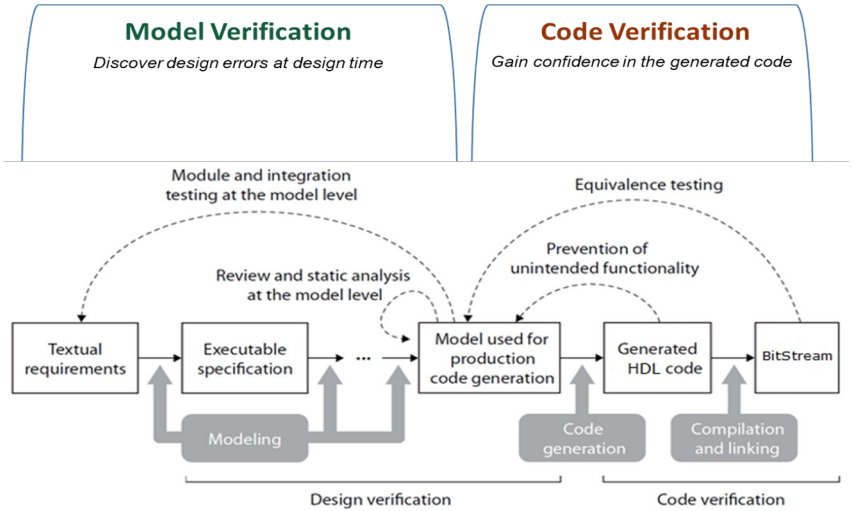
- STPA coupled with Model based Design and Engineering => captures overall system's perspective and interactions .
- **Failure analysis of the SymPLe platform is critical part of design assurance.**
  (In progress)

IEC 61508 and DO-254 V&V Workflow

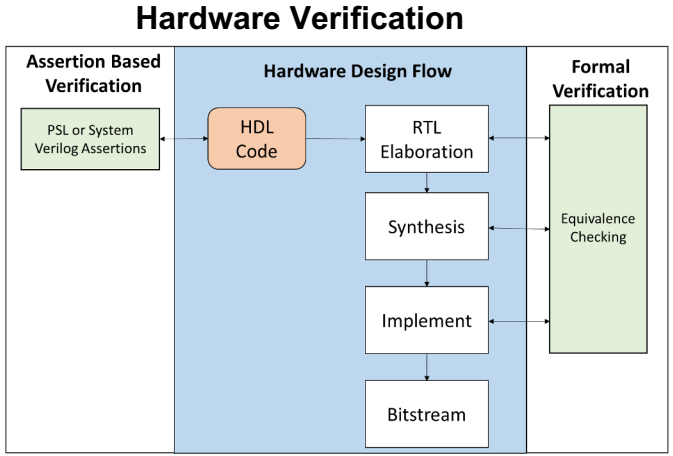
- IEC 61508 – Need for a **Systematic Verification Process** with Evidence Reports
- **Verified incrementally at all levels** - model, generated HDL code, and hardware implementation
- Test Vectors derived from Mid and Low Level Requirements.
- Ensure **100% Requirements Coverage**
- **Formal** verification
- Software **co-simulation** and **code coverage** analysis
- **FPGA-in-loop** testing



Model-Based Designs to FPGAs: Tool Support



Model-Based Design Engineering Tools were IEC 61508 SIL-3 certified



- Model-Based Engineering – MathWorks Simulink
- Trace to Requirements - Simulink Requirements
- Model Advisor Checks - Simulink Check
- Design Error Detection - Simulink Design Verifier
- Testing by Simulation with Model Coverage - Simulink Test, Simulink Coverage
- Property Proving - Simulink Design Verifier

- Assertion Based Verification performed using Mentor Graphics tools.
- Formal verification of the code from RTL level to bitstream generation.
- RTL simulation – Questa Sim
- Gate Level Verification – Formal Pro

References

1. Basili, V. R., Briand, L. C., & Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22(10), 751–761. https://doi.org/10.1109/32.544352
2. Efatmaneshnik, M., & Ryan, M. J. (2016). A general framework for measuring system complexity. Complexity, 21(S1), 533–546. https://doi.org/10.1002/cplx.21767
3. Hennig, A., Topcu, T. G., & Szajnfarber, Z. (2021). So you think your system is complex?: Why and how existing complexity measures rarely agree. Journal of Mechanical Design, 144(4). https://doi.org/10.1115/1.4052701
4. Hite, R., Rajagopala, A., Gautham, S., Deloglos, C., Jayakumar, A., Collins, A., Elks, C., & Gibson, M. (2021). Symple: Complexity-aware design for safety critical I&C Systems. 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S). https://doi.org/10.1109/dsn-s52858.2021.00031
5. KOPETZ, H. (2020). Simplicity is complex: Foundations of Cyber -Physical System design. SPRINGER NATURE.
6. Leveson, N. G.: Complexity and safety. Complex Systems Design & Management. Springer, 2012, pp. 27–39.
7. Olszewska, M., Dajsuren, Y., Altinger, H., Serebrenik, A., Waldén, M., & van den Brand, M. G. (2016). Tailoring complexity metrics for Simulink models. Proccedings of the 10th European Conference on Software Architecture Workshops. https://doi.org/10.1145/2993412.3004853
8. Sinha, K. (n.d.). Structural complexity and its implications for design of cyber-physical systems (dissertation).
9. van der Hoek, A., Dincel, E., & Medvidovic, N. (n.d.). Using service utilization metrics to assess the structure of product line architectures. Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717). https://doi.org/10.1109/metric.2003.1232476