18 October 2019

SAND2019-12574C

# Dishwashers of Armageddon: Verifying high consequence systems for Nuclear Weapons

**Robert Armstrong**, Noah Evans, Geoffrey Hulette, Karla Morris, Jason Michnovicz, Jon Aytac, Philip Johnson-Freyd, Jackson Mayo, Ratish Punnoose

Sandia National Laboratories, Livermore, CA 94551, USA



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Outline


1 Introduction

2 Statecharts

3 Q Tool

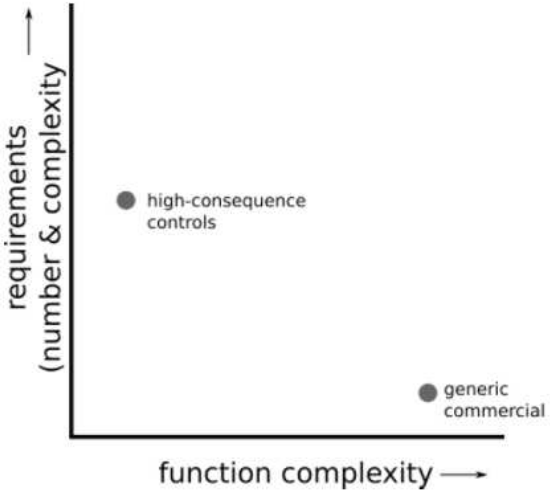
4 Correct by Construction

5 Accidents and Out-of-Nominal Analysis

6 Complexity and Robustness

# High-consequence controls: simple function, complex "always/never" requirements

- Our control systems are mostly *low complexity*, relatively *easy to analyze*,like a dishwasher.
- But, they often have a large number of *complex*, *high-consequence* safety, security, and reliability requirements.
- Low complexity + high consequence + complex requirements = ideal for a formal approach to design and/or verification.

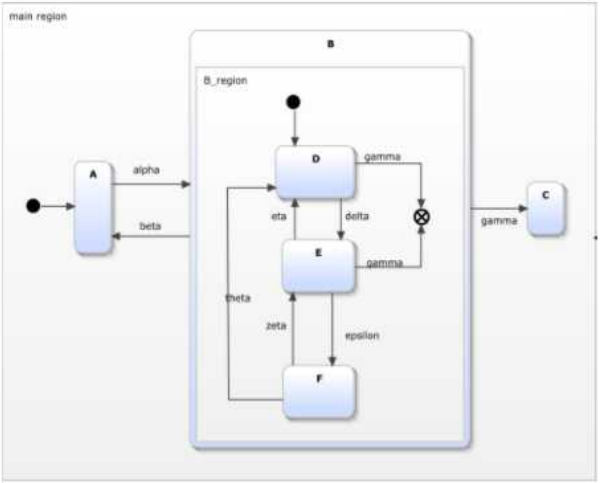



# Research interests

We are broadly interested in research areas related to refinement, action systems, and/or Statecharts:

- "Components", composition and connections to rely/guarantee reasoning
  - exploit abstraction/refinement for scalability of analysis
- Liveness properties, hyper-properties
- Mathematical foundations (coalgebraic models, connections to logic, category theory)
- Practical issues, e.g., tractably managing large networks of components and deep hierarchies.
- Develop tools for verifying embedded hardware/software
- Like DeepSpec, our goal is **One QED**

# Statecharts is an intuitive design language for simple controllers




- Well-suited to our simple controller domain.
- State machine-based semantics, based, most broadly, on TLA – simple, effective, and approachable for end-users.
- Semantics is easy to express within action systems: Event-B, TLA+, etc.
- Control systems engineers have adopted tools with a similar metaphor (e.g. Simulink/Stateflow™, Scade™)

Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Statecharts can support an intuitive, refinement-based design pattern

Chart-based constructions in our Statecharts are refinements in the action system sense:

- Parallel and hierarchical composition
- Signal-based synchronization

Extended with a "math language," our system also supports GCL-style refinement (strengthening guards, weakening actions, etc.)

# Statecharts provide "natural" mechanisms for refinement

Hierarchical composition: An abstract parent state is refined by a set of concrete child states and their transitions.

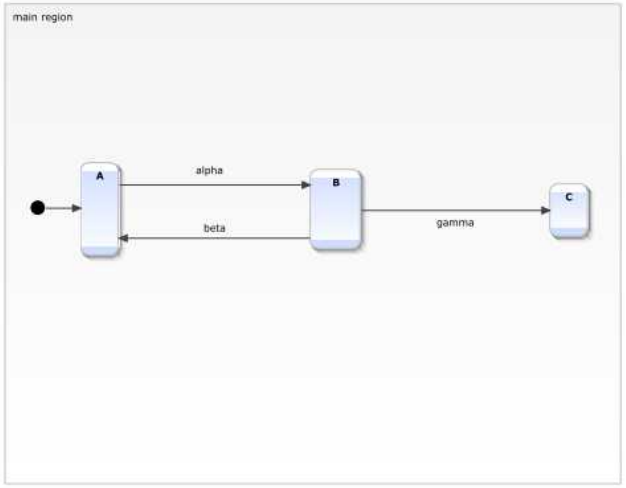



Figure: Abstract model

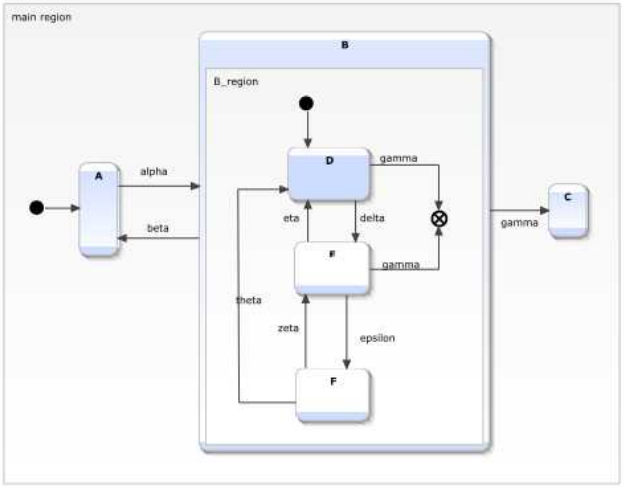



Figure: Refined model

# Statecharts provide "natural" mechanisms for refinement

Guard Strengthening: Add guards to previoulsy defined abstract transitions. New guards are based on new concrete variables.

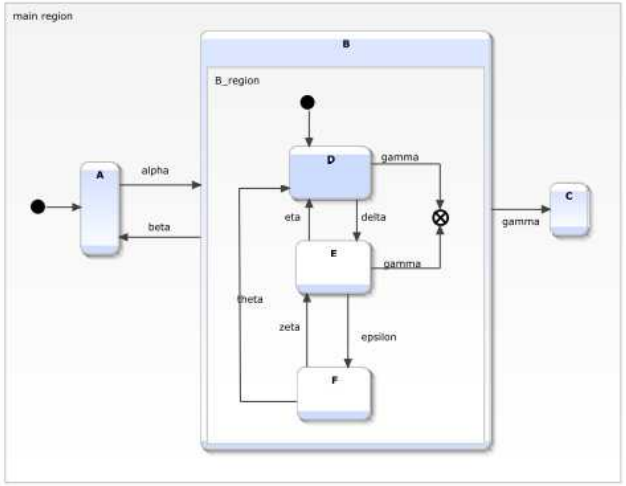



Figure: Abstract model

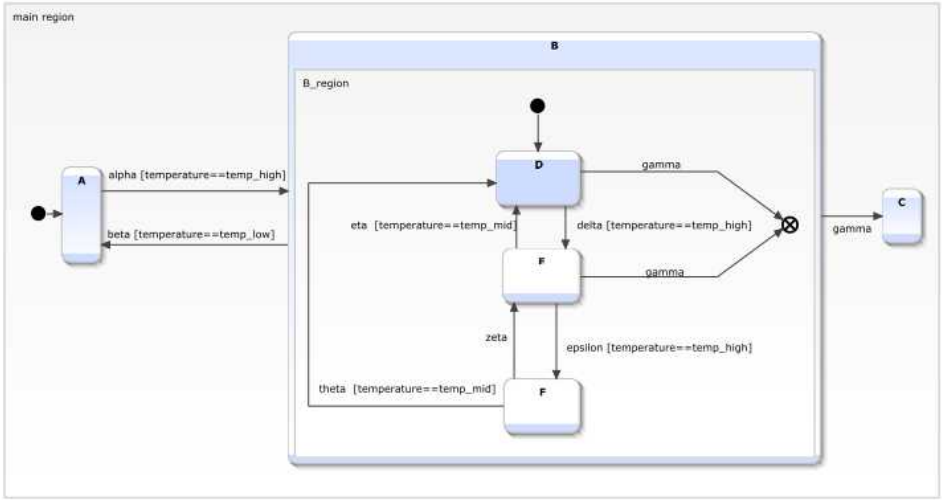
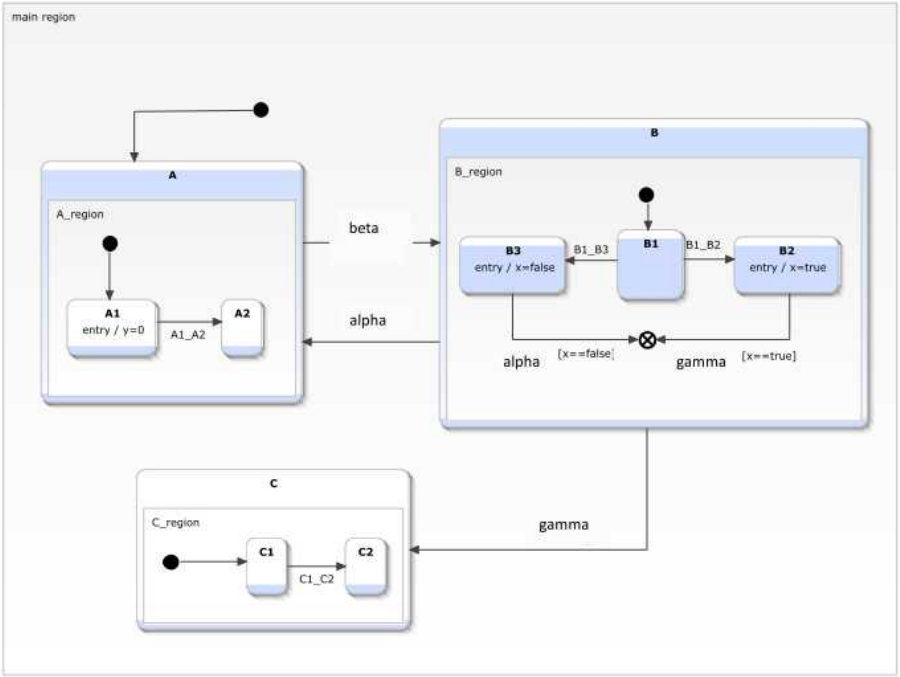



Figure: Refined model with temperature conditions

# Restrictions on the Statechart Semantics

Our version of Statecharts is restricted vis-á-vis Harel's original paper, simplifying the formal semantics:

- Arrows can only go up or down one encapsulation level at a time
- Signals are scoped to the box in which they are created
- We call this variant: "Q Charts"



# Collaborators interested in refinement-friendly Statecharts:

- Jet Propulsion Laboratory (US)


- University of Southampton (UK)


- French Alternative Energies and Atomic Energy Commission (France)

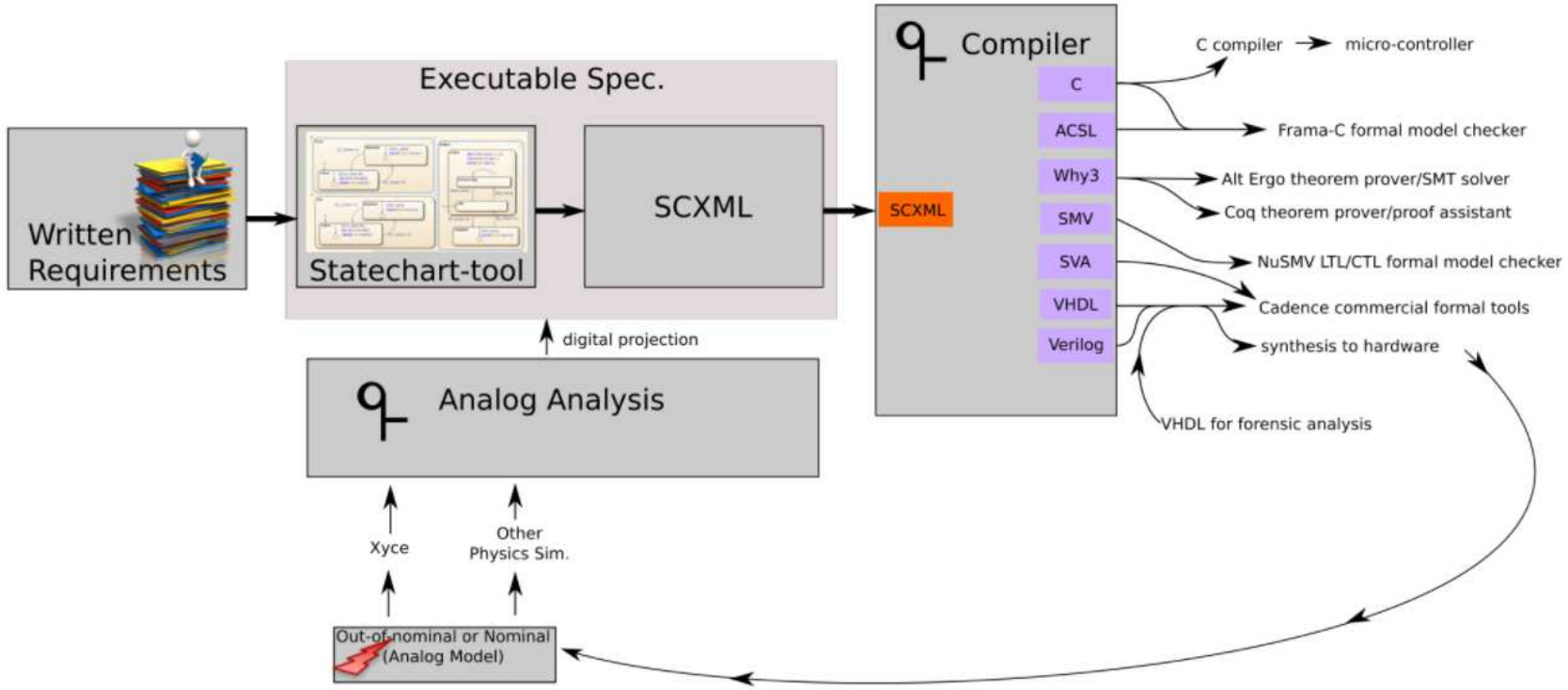

- Atomic Weapons Establishment (UK)

# W3C SCXML Statechart Representation

W3C text representation called SCXML has been modified to accomodate refinement

- XML tools allow new meta-model namespaces to be introduced.
  - Existing SCXML tools will ignore them
- Needed in order to support:
  - Refinement levels (new attribute <iumlb:refinement >)
  - Invariants (new element <iumlb:invariant >)
  - Guards (new element <iumlb:guard >)

# Q tool maps Statecharts to many languages for further analysis by various formal tools





Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Understand out-of-nominal electrical behavior: Failure modes can be understood via abstractions

- Examples of failures that result in an overapproximation:
  - A logic gate becomes unreliable and nondeterministic
  - A sensor fails, providing random input to a digital control
  - Generally: any malfunction that generates additional behaviors that were not part of the design intent

- Errors induced by environmental physics are common:
  - Radiation (cosmic rays, etc.)
  - Heating (fire, etc.)
  - Physical insult (destruction of sensor, etc.)
- Abstraction techniques can reveal failure modes for which a particular design will be robust
- Abstraction techniques can support designed-for failure modes anticipating likely accidents and faults

Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Square diagram shows refinement relationships that preserve requirements

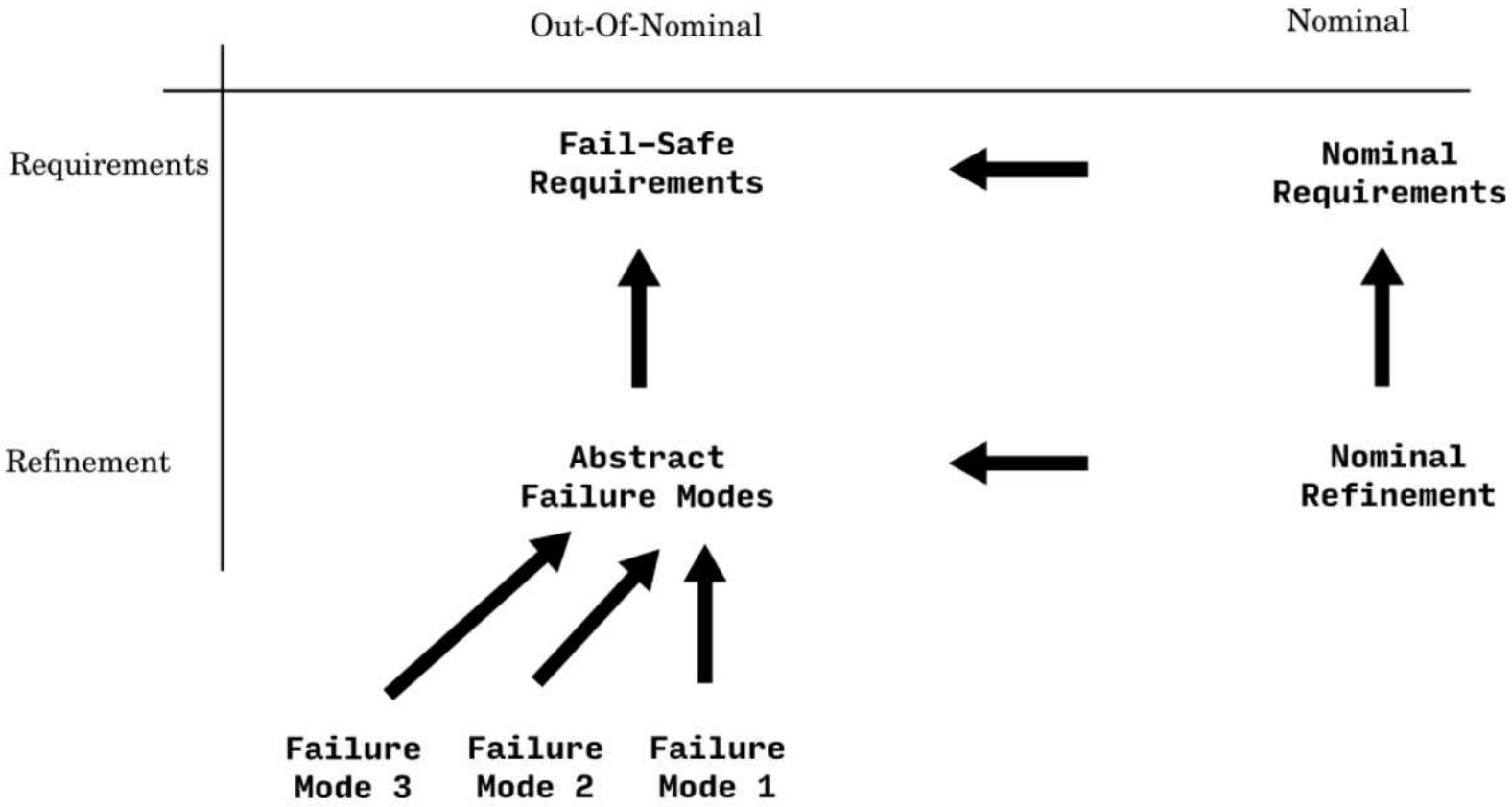



Figure from J. R. Mayo et al., Proc. 4th FTSCS Workshop, CCIS 596, doi:10.1007/978-3-319-29510-7_10. © 2016 Springer.

- Refinement/abstraction conceptual diagram for treating out-of-nominal and nominal models in a unified way
- Arrows point in the direction of abstraction

# Systems designed using formally-informed design appear more robust, even beyond what is proven

- The SMACCMPilot project (Hickey et al. 2014) developed control software for a drone in the Ivory domain-specific programming language (DSL)
  - Ivory constrains against some unexpected behavior by enforcing basic memory safety properties
  - The resulting drone software was dubbed "unhackable" after extensive red-teaming
- The Compcert C compiler (Leroy 2009) was developed in the Coq theorem prover, tantamount to a restricted programming language
  - Extensive randomly generated tests ("fuzzing") uncovered hundreds of errors in mainstream C compilers but none in Compcert's core (Yang et al. 2011)

Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Outsize benefits of up-front formal modeling have been noted in practice

- Key observation: design for analysis yields increased robustness, regardless of *when* or even *whether* the analysis is performed
  - Faults and vulnerabilities are reduced if the developer starts with a high-level formal model – even if no further verification is done and even if the implementation is not explicitly constrained (Woodcock et al. 2009)
  - This supports our hypothesis that robustness is conferred because of design characteristics promoted by the formal modeling process
- By contrast, formal verification *after the fact* does not increase robustness more broadly, if the design was not formally informed
  - Example: the LLVM compiler infrastructure has undergone some formal analysis, but fuzzing suggests it is no more robust than other compilers

# Conclusion: Broader principles support robustness in digital systems informed by formal constraints

- Biological and social complex systems typically are *not* formally verified, but show impressive robustness to unforeseen failures
- Why? They have inherent stability constraints from their origins in adaptation and selection
- Our hypothesis: Digital designs constrained by formal requirements also exhibit enhanced robustness to unforeseen failures by a similar mechanism

Verifying simple, yet high-consequence controls

Introduction

Statecharts

Q Tool

Correct by Construction

Accidents and Out-of-Nominal Analysis

Complexity and Robustness

# Digital Assurance "Q" Team

Rob Armstrong
Digital Foundations
& Maths

Jon Aytac
Digital Foundations
& Maths

Ratish Punnoose
Weapons Subsystems 2

Geoff Hulette
Digital Foundations
& Maths

Ben Nilsen
Digital Foundations
& Maths

Philip Johnson-Freyd
Digital Foundations
& Maths

Tristan Duckworth
Cyber Systems Research

framework
digital assurance

Shyamali Mukherjee
Scalable Modeling
Analysis

Jackson Mayo
Scalable Modeling
Analysis

Raheel Mahmood
B83 System
Engineering

Laura W. Edwards
Digital Foundations
& Maths

Karla Morris
Digital Foundations
& Maths

Srinivas Nedunuri
Data Science &
Cyber Analytics

Samuel Pollard
Digital Foundations
& Maths

John Matthew Bender
Digital Foundations
& Maths

Andrew Smith
Digital Foundations
& Maths

Jason Michnovicz
Digital Foundations
& Maths

Misha Shashkov
Quantitative Modeling
& Analysis

Blake Rawlings
Digital Foundations
& Maths