NAVAL POSTGRADUATE SCHOOL

Behavioral Modeling of System Architectures with Monterey Phoenix

Kristin Giammarco, Ph.D.
Associate Professor, Department of Systems Engineering

Monica Farah-Stapleton
Ph.D. Candidate, Department of Computer Science

Mikhail Auguston, Ph.D.
Associate Professor, Department of Computer Science

Monterey, California

WWW.NPS.EDU

Leonard Kleinrock on Emergent Behavior

Emergent behavior is:
you create a system,
you think you know how it's going to behave,
you think you know how it's going to control,
you think you've written down the equations or analyzed it,
and it configures itself in a way that you did not anticipate,
and exposes a behavior, a phenomenon, a result, a performance,
that you did not see coming at you.
And it's because the control is all over the place.
The interaction of a large number of simple things is very hard to predict.

*The complexity is not in the individual things, **it's in the way in which they're interconnected**.*

- Leonard Kleinrock, Distinguished Professor of Computer Science, UCLA

http://dsc.discovery.com/tv-shows/curiosity/topics/l-kleinrock-what-is-emergent-behavior.htm

NAVAL POSTGRADUATE SCHOOL

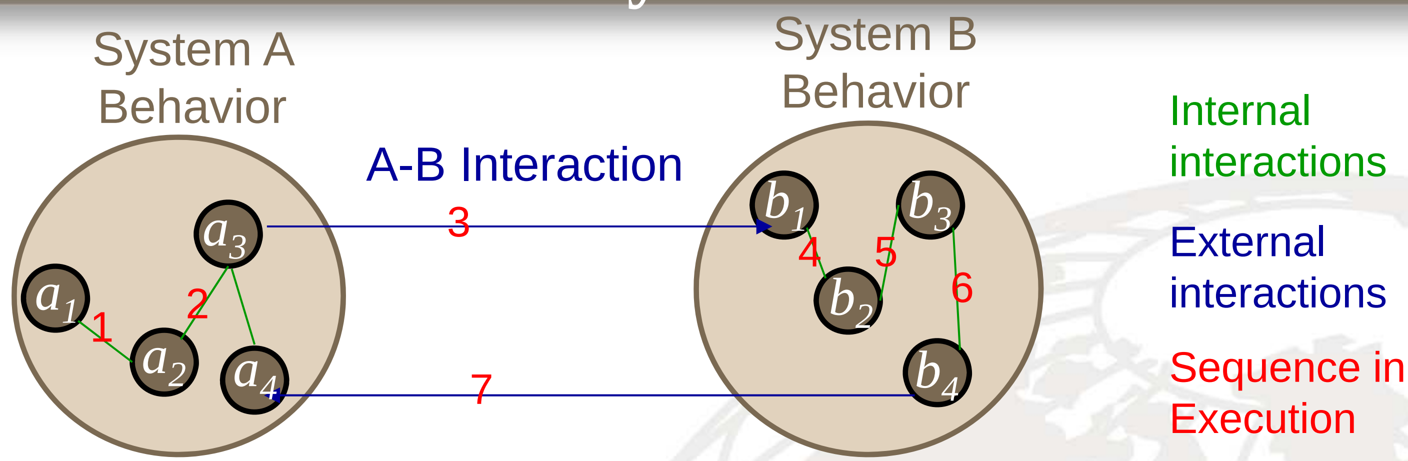
Agenda

- Monterey Phoenix, a new strategy that separates the specification of system interaction from system behavior
- Limitations in current strategies lacking separation of concerns
- Using MP to generate views, use cases, and to conduct architecture verification and validation
- Using MP to expose undesired emergent behavior in an architecture model in advance of system testing or operation
- Summary, Way Ahead, Discussion

The key advancement for SoS modeling:
a decoupling of *system interaction* and *system behavior.*

What Does Separation of System Interaction from System Behavior Mean?

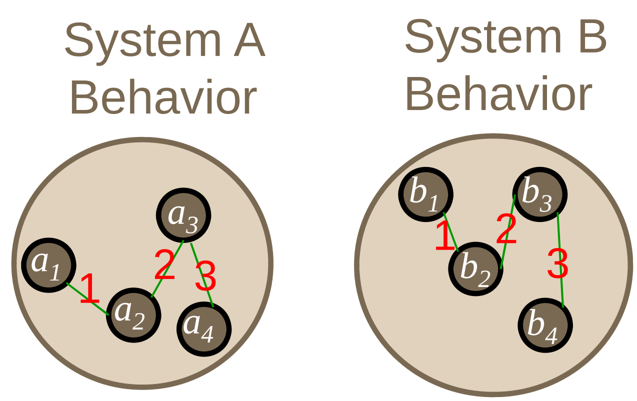
From…

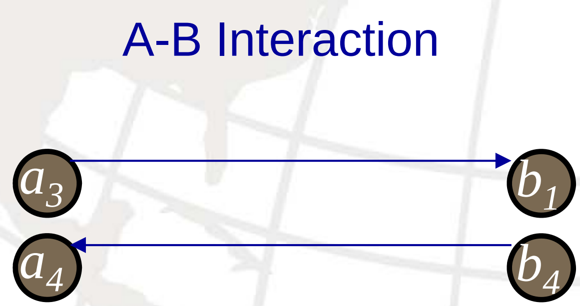


Sequencing *external* interactions in the same model as *internal* interactions

To…

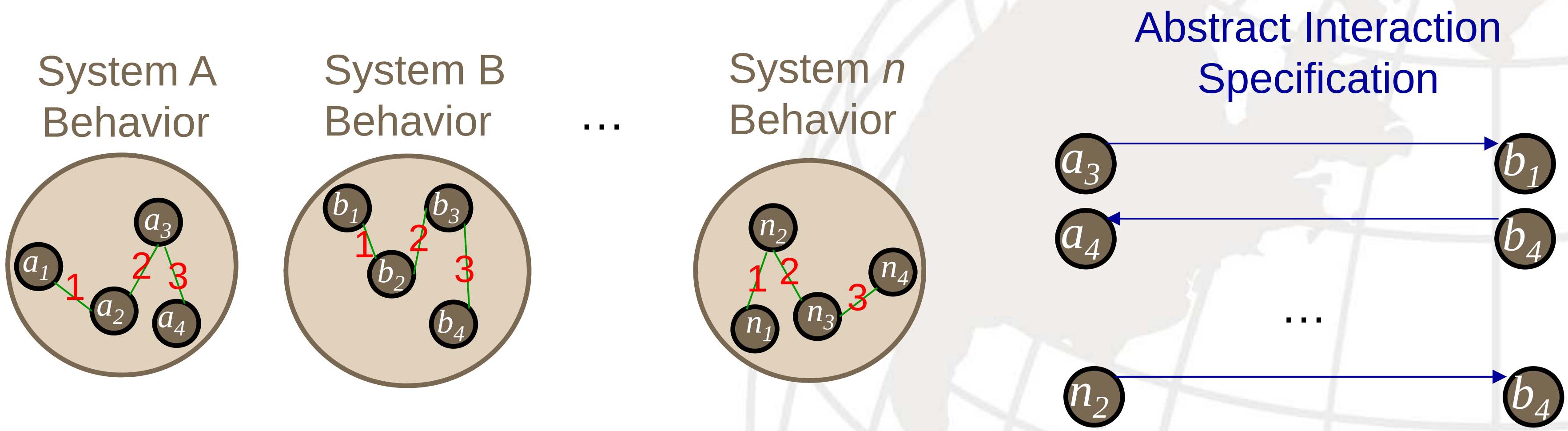
Specifying behavior of each system *separately* from interactions among those systems, and compute possible sequences from there





Why Separate System Interaction from System Behavior?

*A set or arrangement of systems that results when **independent** and task-oriented systems are **integrated** into a larger systems construct, that delivers unique capabilities and functions in support of missions that cannot be achieved by individual systems alone.*

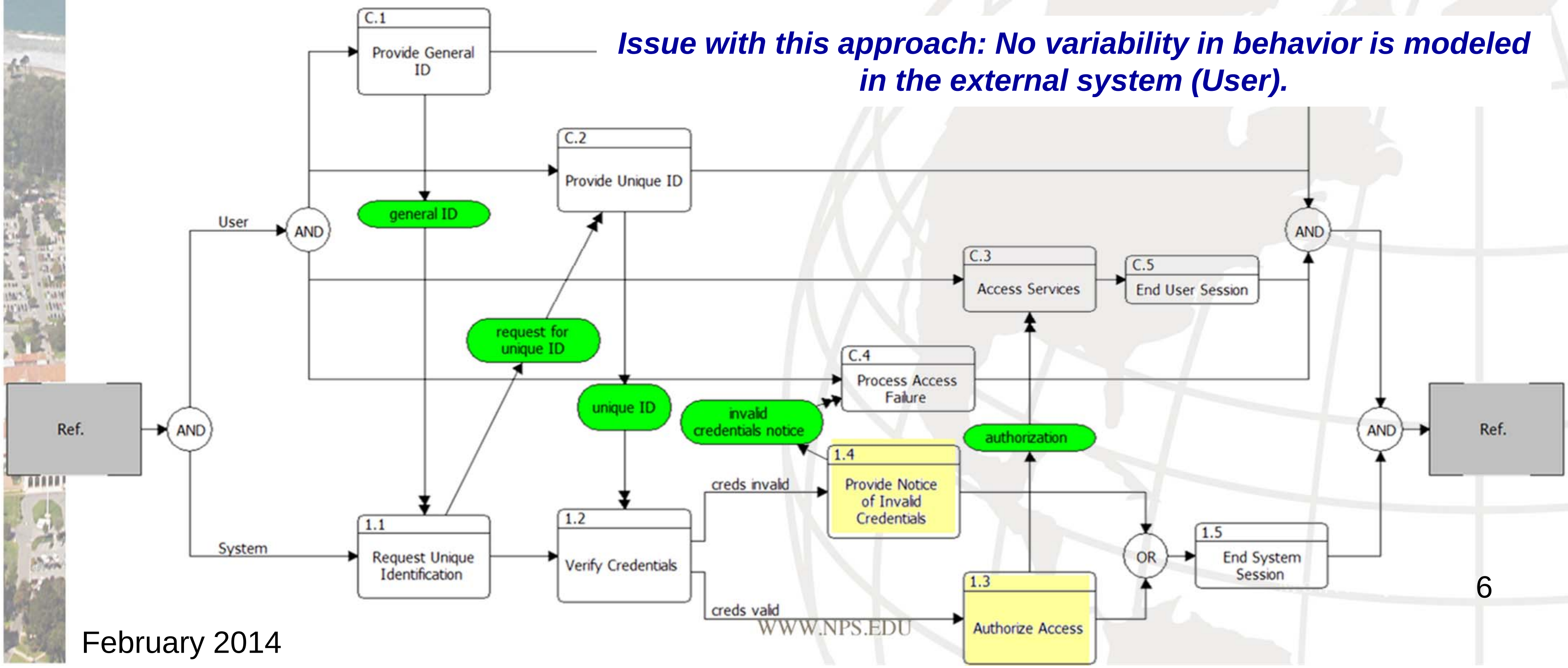


SoS Definition Ref: Vaneman, W., Jaskot, R., 2013, A Criteria-Based Framework for Establishing System of Systems Governance, Proceedings of the 7th Annual International IEEE Systems Conference, pp. 491-496, Orlando, FL, April 15-18, 2013.

Authentication Example:
Describing behavior for one system (EFFBD Notation)

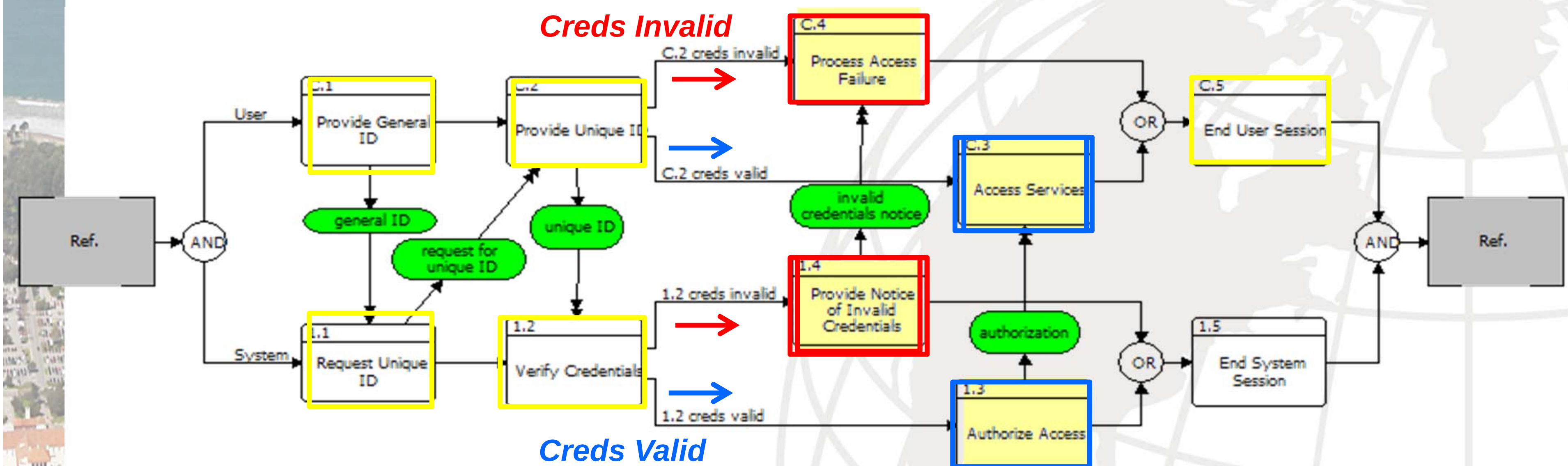
1. User provides a general identification.
2. System requests unique identification.
3. User provides a unique identification.
4. If the credentials are valid, the System authorizes the User to access the services; otherwise the System notifies the User that credentials are invalid and the user may re-attempt access up to two more times.
5. The User or the System ends the session.

Issue with this approach: No variability in behavior is modeled in the external system (User).



Authentication Example:
Describing behavior for both systems (EFFBD Notation)

1. User provides a general identification.
2. System requests unique identification.
3. User provides a unique identification.
4. If the credentials are valid, the System authorizes the User to access the services; otherwise the System notifies the User that credentials are invalid and the user may re-attempt access up to two more times.
5. The User or the System ends the session.



Issue #1: Selection of behavior (for creds valid, creds invalid) is not automatically coordinated in the notation itself.

Issue #2: Multi-actor activity diagrams do not scale well when attempting to represent a broad range of each actor's behavior.

Monterey Phoenix (MP)

- Approach to formal software and system architecture specification based on behavior models
- A view on the architecture as a high level description of possible system behaviors, emphasizing the behavior of subsystems and interactions between subsystems
- The emphasis on specifying the interaction between the system and its environment
- The behavior composition operations support architecture reuse and refinement towards design and implementation models
- Executable architecture models provide for system architecture testing and verification with tools

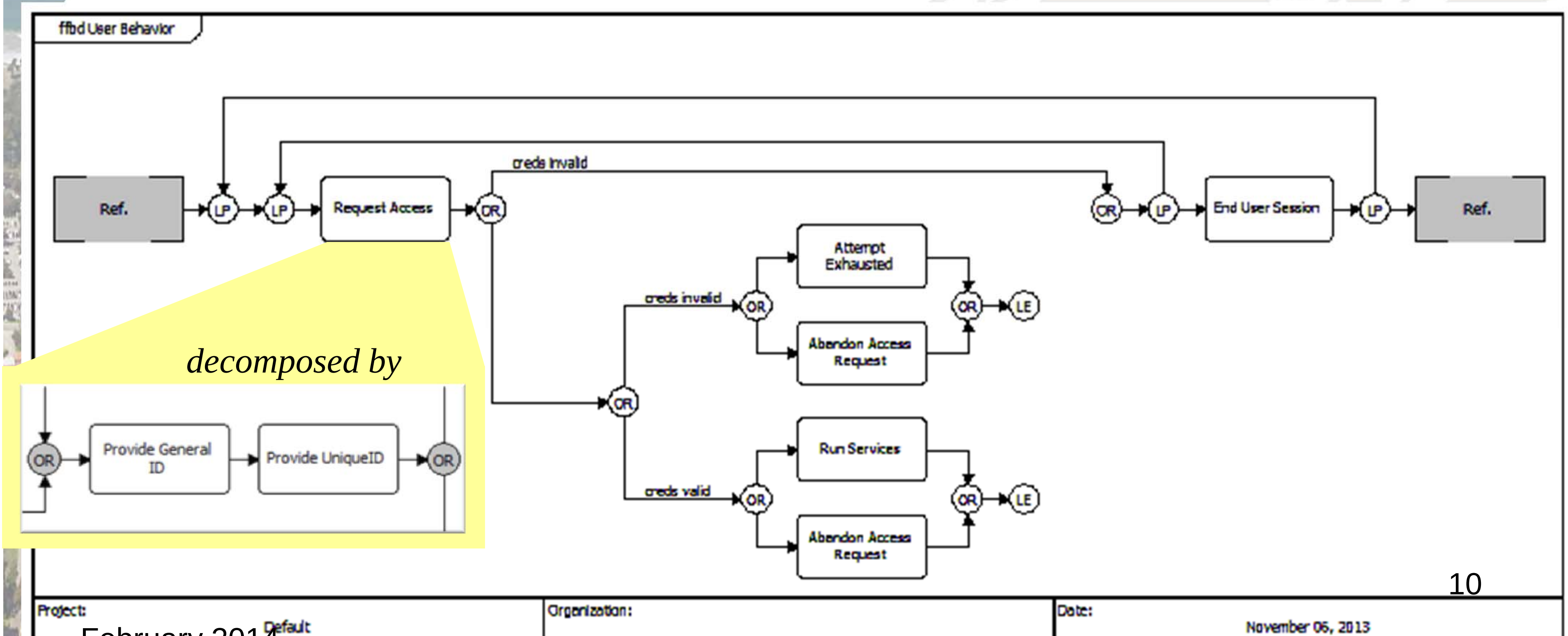
Monterey Phoenix Basic Concepts

- Event: Any detectable action in a system's or environment's behavior
- Event Trace: Set of events with two basic partial ordering relations, precedence (PRECEDES) and inclusion (IN)
- Event Grammar: Specifies the structure of possible event traces
- Schema:
  - Represents instances of behavior (event traces)
  - Contains collection of events called roots (components and connectors)
  - Composition operations specify interactions between root behaviors
  - Additional constraints on behaviors
- Behavior Model of a System: Specified as a set of all possible event traces using a schema
- Executable Architecture Model: Provides rapid prototype of system under consideration
- Automated tools: MP supports automated tools for system architecture validation and verification (MP-Alloy, MP C++ Implementation, Eagle 6)

Model Behavior of Environment: User Behavior

```
ROOT User:
(*      request_access
        (* creds_invalid request_access  *)
        (  creds_valid   (run_services   | abandon_access_request)      |
           creds_invalid (attempt_exhausted | abandon_access_request) )
        end_User_session                                               *);

request_access:  provide_general_ID  provide_unique_ID;
```



Model Behavior of Environment: System Behavior

```
ROOT System:
(* request_unique_ID
   [ creds_invalid request_unique_ID
         [ creds_invalid request_unique_ID
           [ creds_invalid  attempt_exhausted
                  invalid_creds_notice cancel_access_request ] ] ]
   [  (creds_valid   ( authorize_access   run_services            |
             long_wait_for_User   cancel_access_request )  |
       creds_invalid  long_wait_for_User   cancel_access_request      ) ]
   end_System_session     *);
```

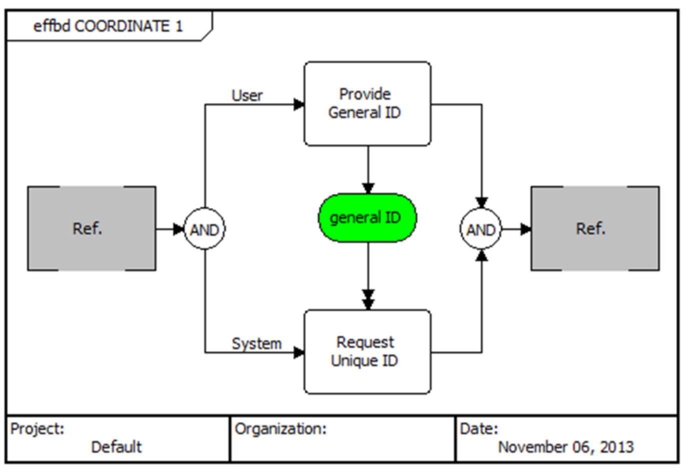
Corresponding FFBD in Backup

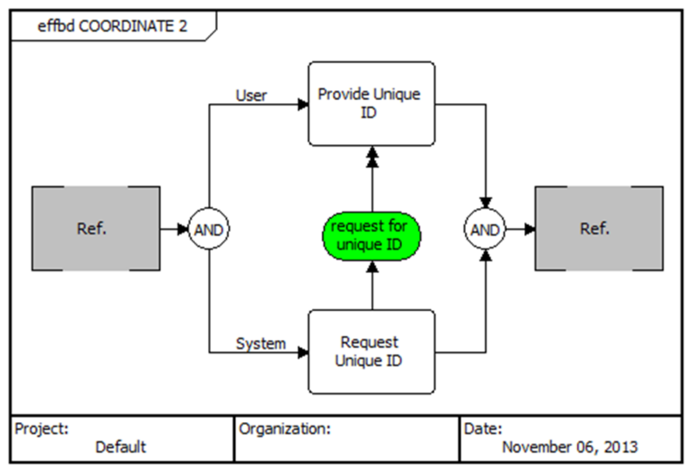
NAVAL POSTGRADUATE SCHOOL

Model Component Interactions Abstractly and Separately

```
COORDINATE        (* $x: provide_general_ID *)       FROM User,
                  (* $y: request_unique_ID *)        FROM System
                          ADD $x PRECEDES $y;
COORDINATE        (* $x: request_unique_ID *)        FROM System,
                  (* $y: provide_unique_ID *)        FROM User
 ADD $x PRECEDES $y;

User, System SHARE ALL    creds_valid, creds_invalid,
                          attempt_exhausted, run_services;
```





Branches **shared by** components:
- creds valid *shared by* User, System
- creds invalid *shared by* User, System

Functions **shared by** components:
- Attempt Exhausted *shared by* User, System
- Run Services *shared by* User, System

Authentication Model in Eagle 6

NAVAL POSTGRADUATE SCHOOL

RIVERA CONSULTING GROUP

EAGLE6 MODELING

Home | Run Eagle6 Beta | Tutorials | Manual | Contact

Eagle6 System Architectural Modeling Interface

Tested with Internet Explorer, Mozilla Firefox and Google Chrome

Code Samples

Description	Download file	Load into editor
Communication between systems A,B. A requests data from B. B sends data to A.	download	load
Communication between systems A,B,C. A requests data from B. In order to respond to A, B requests data from C. C sends data to B. B sends data to A.	download	load
Naval Gunship Model v1.01	download	load

Code Editor

```
// January 11, 2014
//-----------------------------------------------------------------------------------
//      Authentication scenario architecture model, fast version
//      from
//      Kristin Giammarco, Mikhail Auguston, Well, You didn't Say not to! A Formal Systems
//       Engineering Approach to Teaching an Unruly Architecture Good Behavior,
//       Complex Adaptive Systems Conference, November 13 - 15, 2013, Baltimore, MD
//      http://www.sciencedirect.com/science/article/pii/S1877050913010739
//-------------------------------------------------------------------------------------

ROOT    User:  (* request_access
                   (* <0-3> creds_invalid request_access  *)
                   (  ( creds_valid  (run_services  | abandon_access_request))      |
                      ( creds_invalid (attempt_exhausted | abandon_access_request))  )
                   end_User_session
                 *);

        request_access:         provide_general_ID  provide_unique_ID;

ROOT    System: (* request_unique_ID
                 [ creds_invalid request_unique_ID
                         [ creds_invalid request_unique_ID
```

Edit Code

Exhaustive Scenario Generation | Random Scenario Generation

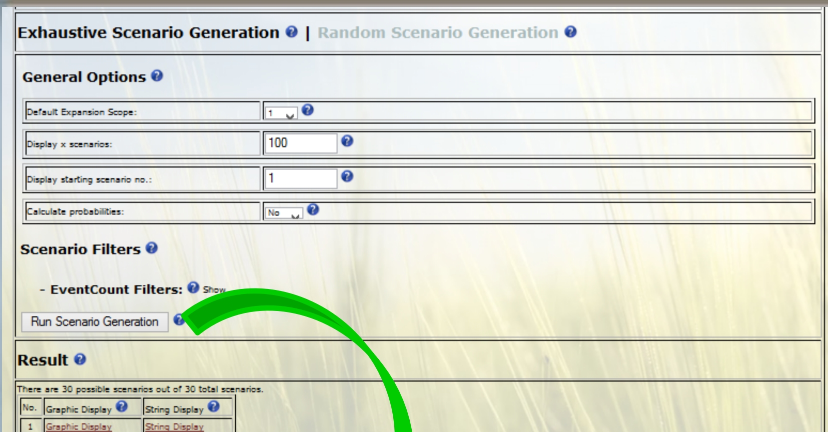
Result

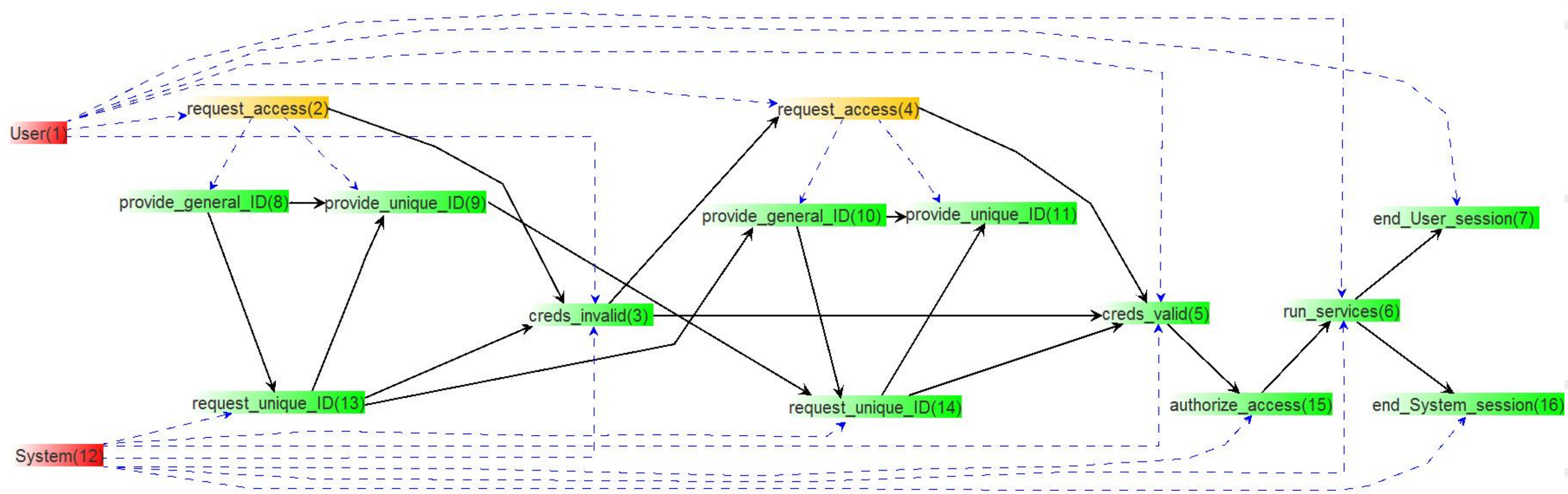
There are 30 possible scenarios out of 30 total scenarios.

No.	Graphic Display	String Display
1	Graphic Display	String Display
2	Graphic Display	String Display
3	Graphic Display	String Display
4	Graphic Display	String Display
5	Graphic Display	String Display
6	Graphic Display	String Display
7	Graphic Display	String Display
8	Graphic Display	String Display
9	Graphic Display	String Display
10	Graphic Display	String Display
11	Graphic Display	String Display
12	Graphic Display	String Display
13	Graphic Display	String Display
14	Graphic Display	String Display
15	Graphic Display	String Display
16	Graphic Display	String Display
17	Graphic Display	String Display
18	Graphic Display	String Display
19	Graphic Display	String Display
20	Graphic Display	String Display
21	Graphic Display	String Display
22	Graphic Display	String Display
23	Graphic Display	String Display
24	Graphic Display	String Display
25	Graphic Display	String Display
26	Graphic Display	String Display
27	Graphic Display	String Display
28	Graphic Display	String Display
29	Graphic Display	String Display
30	Graphic Display	String Display

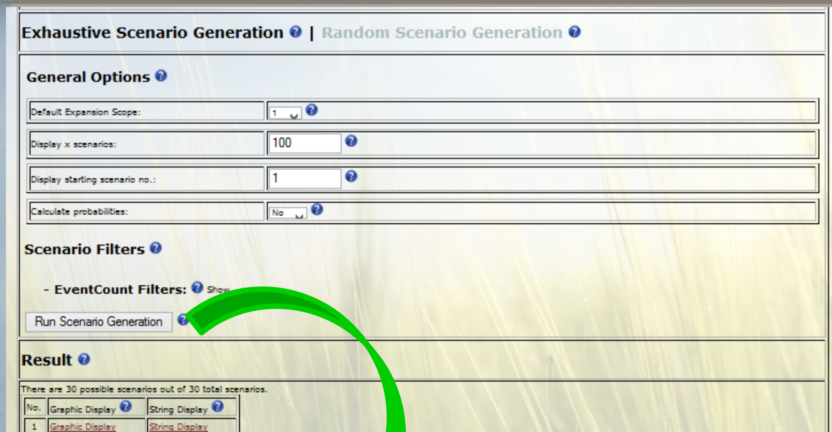
30 Unique Use Cases from the one Authentication Model in MP

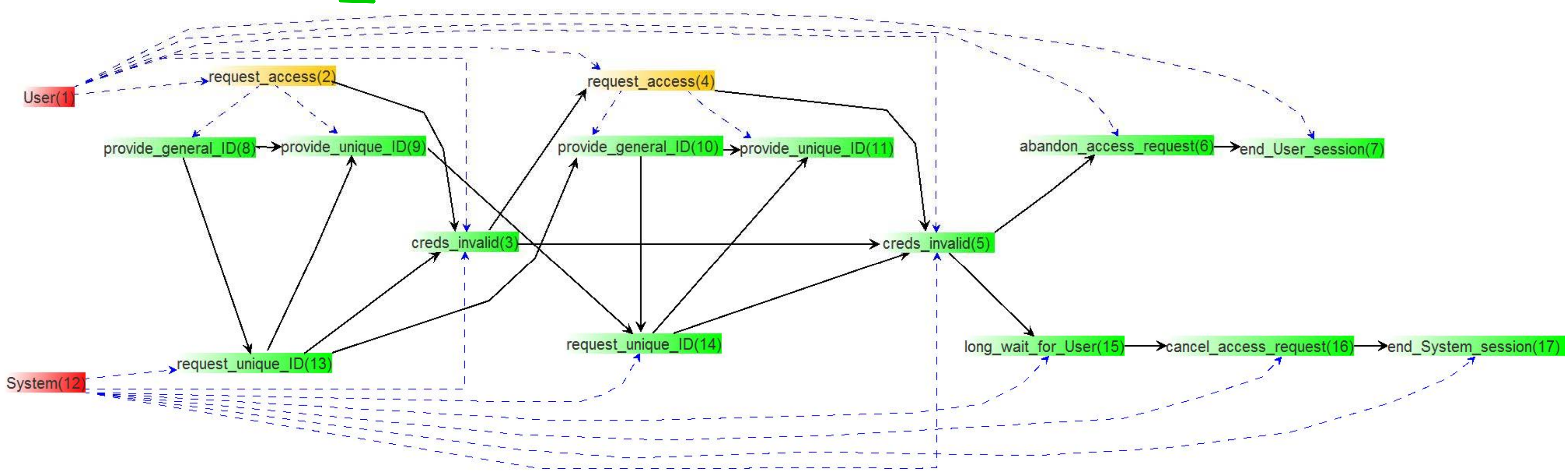
Use Case 1: User gets access after one unsuccessful attempt.





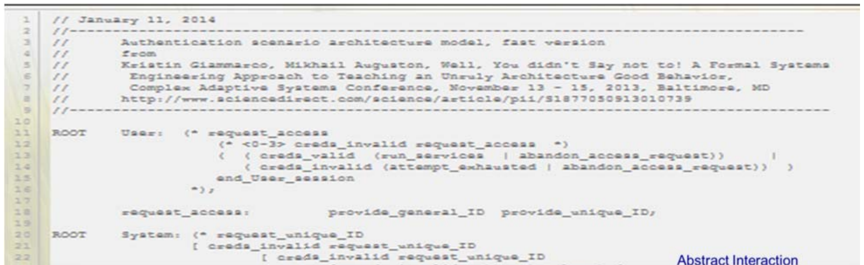
Use Case 2: User abandons access request after two unsuccessful attempts.

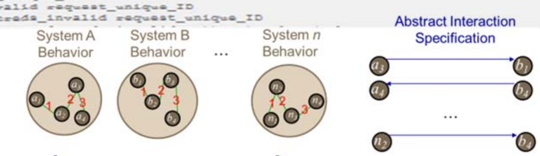




Using MP to Expose Latent Behavior Before It Happens

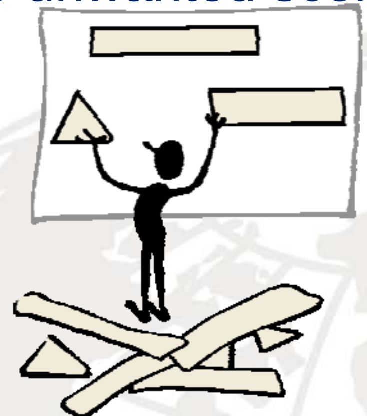
1. Behavior description for each system + abstract interaction specification



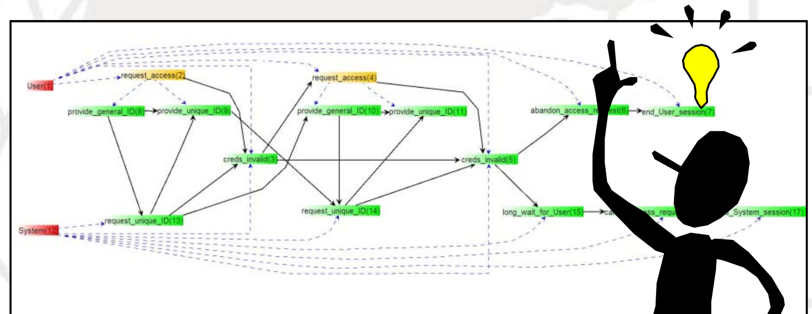


2. Scenario (use case) generation

4. Modify the design to exclude unwanted scenarios



3. Scenario visualization, inspection, and assertion checking



e.g., "Are there any instances where a user can gain access to the system after three attempts?"

NAVAL POSTGRADUATE SCHOOL

Architecture: A Bridge

User Requirements

System's Architecture

High Level Design

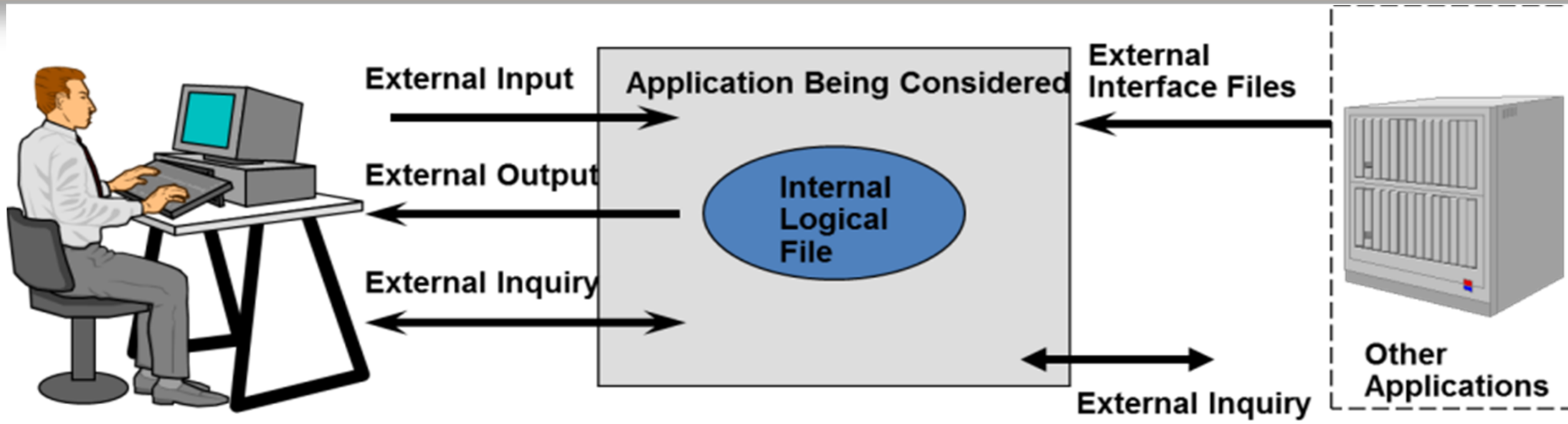
Stakeholder Examples	Typical Questions or Groups of Questions
Customer	Are user, technical, cost, and management expectations being met?
Users	Does this system do what was expected? Does it fulfill prioritized requirements?
Engineers / Designers	What implementation option(s) should be considered to meet performance expectations? What are environment interactions and constraints for each option?
Testers	What are optimal instrumentation points? What statistics should be gathered? What is the correct level of abstraction?
Cost Analysts	What is the cost of the system from requirements elicitation thru software evolution?

Resourcing Strategies Enabled By Architectural Modeling

Use Monterey Phoenix to Reason About System and Environment Behaviors, and Inform Resourcing Decisions

- Resourcing decisions must consider overall project costs
  - Include facilities, processes, staffing, tools, products, and development
  - For many projects, development effort represents the biggest component of cost
- Spectrum of estimation strategies from Excel through software parametric models
  - MP can provide a framework to support many existing resource estimation methodologies
  - Function Point analysis
    - Software size & cost estimates using well-established methodologies
  - Software Non-functional Assessment Process (SNAP) analysis
    - Nonfunctional requirement verification using events with attributes such as duration, delay

Functionality Viewed From The User's Perspective



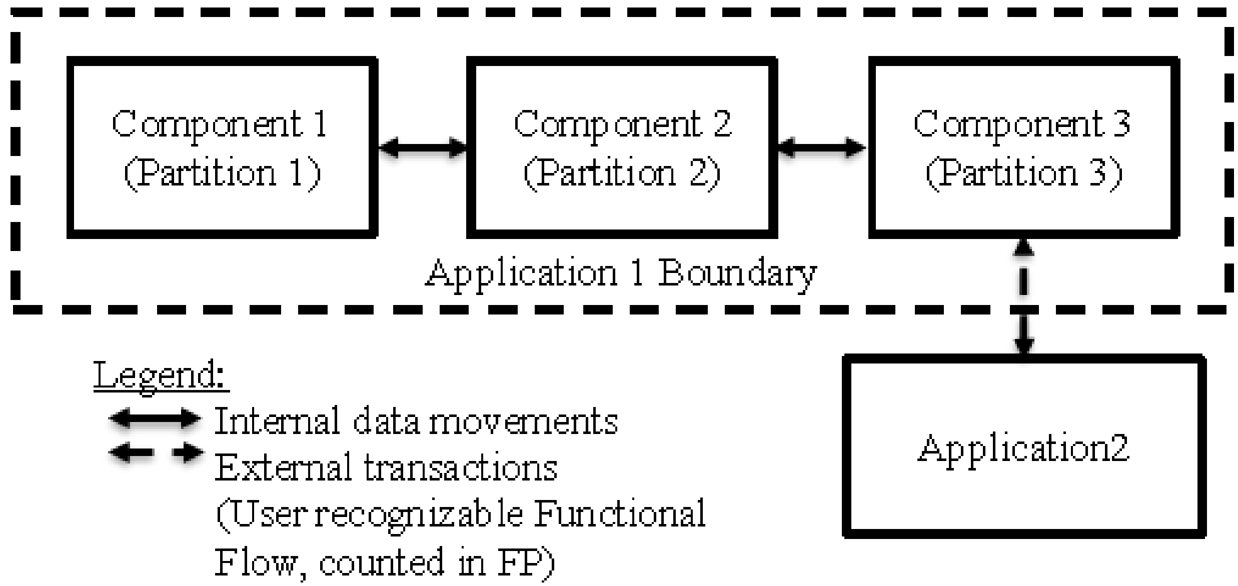
Function Point Terminology

- Describe interactions of a user, system and its environment.
- External Inputs (EI): Input data that is entering a system.
- External Outputs (EO) and External Inquires (EQ): Data that is leaving the system.
- Internal Logical Files (ILF): Data that is processed and stored within the system.
- External Interface Files (EIF): Data that is maintained outside the system but is necessary to satisfy a particular process requirement.

Function Point Analysis Practice

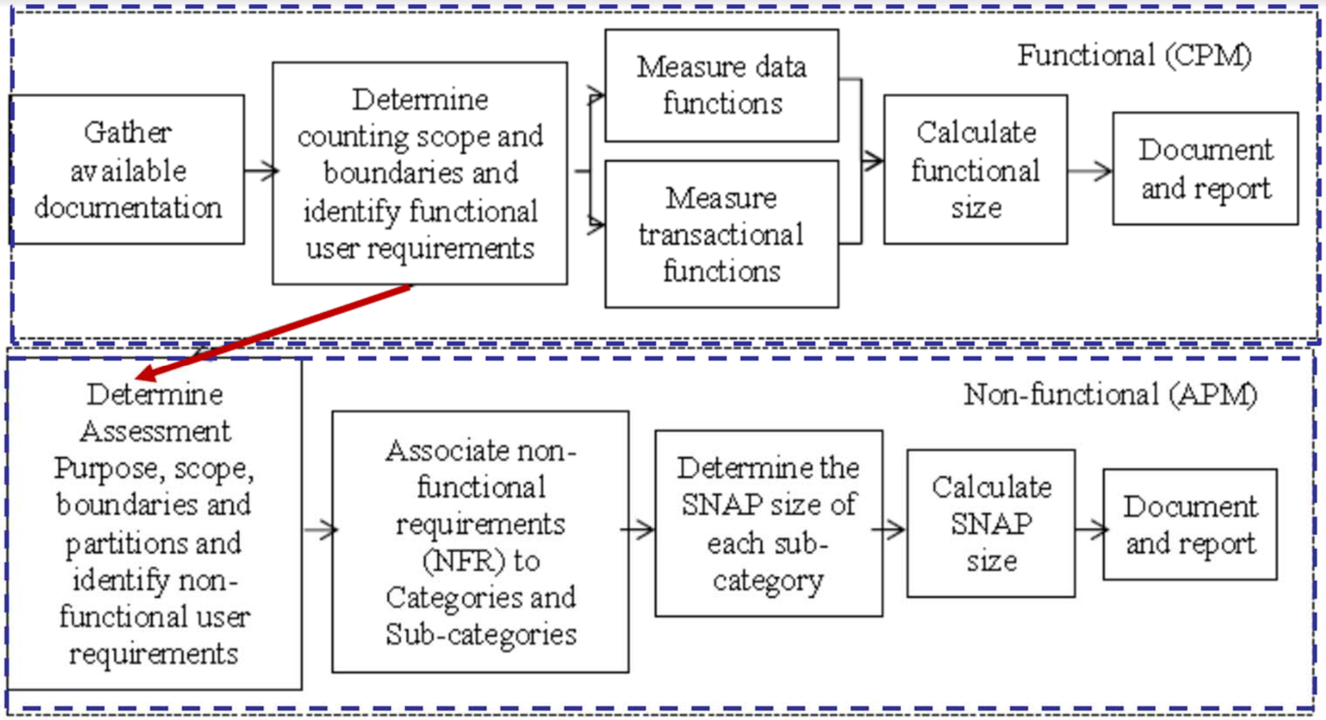
- Count Data and Transactional Function Types.
- Determine Unadjusted FP count and the Value Adjustment Factor.
- Calculate final Adjusted FP Count.

Software Non-functional Assessment Process (SNAP)



- Software Non-functional Assessment Process (SNAP)
- Apply SNAP sizing process
  - Identify application boundary.
  - Associate non-functional requirements with relevant categories and their sub-categories.
  - Size each sub-category according to type and complexity using standardized set of basic criteria (NOTE: Size is sum of sizes sub-categories.
  - Total sizes to give measure of non-functional size of the software application.

Relating MP to FP and SNAP



- Use MP to unambiguously describe boundaries and interactions of the system, user, and environment
- Apply FPA practice
- Apply SNAP sizing process
- Assess applicability of FPA and SNAP to inform project and program life cycle cost estimates

Application Boundaries must be Consistent between FPA and SNAP Processes

How Do We Implement?

- An MP prototype has been implemented as a compiler generating an Alloy relational logic model from the MP schema and then running the Alloy Analyzer to obtain event traces and to perform assertion checks
- A prototype trace generator converts MP schemas into a C++ code, compiles and runs it. Generation speed reaches 10^4 events/sec, the search space up to 10^{15} traces
- On-line demo is available at
  - http://modeling.eagle6.com
- MP model checking tool implemented at the National University of Singapore by Dr. Jin Song Dong group
- Current research efforts at Naval Postgraduate School looking at how to integrate the MP approach into the systems engineering vernacular

NAVAL POSTGRADUATE SCHOOL

MP Research Areas (Next Steps)

- Composition and reuse of MP models (e.g., for improved system of systems modeling)
- Architecture views (extracting different diagrams from the MP model, such as UML/SySML activity diagrams, functional flow diagrams, sequence diagrams, context diagrams, fit-for-purpose (custom) diagrams)
- Next implementation of the event trace generator and assertion checker
- Improved trace visualization
- Behavior patterning
- System performance estimates based on event timing attributes: duration, latency, critical paths
- Statistical system behavior simulation; system safety estimates based on interaction with environment models

MP Research Areas (Next Steps, continued)

- Methodology for using MP in a Systems Engineering problem space (e.g., requirements engineering, risk analysis, reliability/maintainability/availability analysis, architecture and design, test and evaluation, cost estimation)
- Use case scenarios for a specific system
  - Representative sets of use cases (event traces) in a domain of interest (see above for examples)
  - Use case generation methods (efficient strategies for organizing MP models)
  - MP models incorporating criteria from different stakeholders to inspect resulting use cases for contradictions
  - Application for implementation testing, maintenance, and/or documentation
  - Assertion checking for use cases leveraging Small Scope Hypothesis (testing an MP model for the presence or absence of some suspected property).

Summary

How Does Monterey Phoenix Address Issues With Traditional Architectures?

- Captures behaviors and interactions between system and environment
- Captures design decisions early, assess and modify without incurring the costs of incorrect implementations
- Enables Verification and Validation of system, focused on behaviors, interactions, and automated tools for early verification
- Enhances and extends DoDAF, UML, and SysML frameworks and notations
- Executable architecture model leveraging "lightweight" formal methods to unambiguously describe behaviors
- Provides uniform way to extract use cases from single architecture model composed of component behavior algorithms and abstract interaction specification

Why Do We Care?

- Inform quantifiable cost estimates across enterprise and lifecycle of system
- Specifically address concerns of multiple stakeholders
- Allow automatic generation of behavior examples (Use Cases) for early system architecture analysis, testing, verification, and validation

Questions Please

kmgiamma@nps.edu
monica.farah-stapleton@dha.mil
maugusto@nps.edu

Monterey Phoenix and Related Work: http://faculty.nps.edu/maugusto

Simplicity does not precede complexity, but follows it.

-Alan J. Perlis, Yale University

Perlis, Alan J., "Epigrams in Programming." SIGPLAN Notices 17(9), September 1982
More epigrams at http://www.cs.yale.edu/quotes.html

References

Aizier, B., Lizy-Destrez, S., Seidner, C., Chapurlat, V., Prun, D., Wippler, J., 2012, xFFBD: Towards a Formal Yet Functional Modeling Language for System Designers, in Proceedings of the 22nd INCOSE International Symposium, Rome, Italy, July 9-12, 2012.

Auguston, M., 2009, Software Architecture Built from Behavior Models, ACM SIGSOFT Software Engineering Notes, 34:5.

Auguston, M., Whitcomb, C., 2010, System Architecture Specification Based on Behavior Models, in Proceedings of the 15th ICCRTS Conference (International Command and Control Research and Technology Symposium), Santa Monica, CA, June 22-24, 2010.

Auguston, M., Whitcomb, C., 2012, Behavior Models and Composition for Software and Systems Architecture, Proceedings of the 24th ICSSEA Conference (International Conference on Software and Systems Engineering and their Applications), Paris, France, October 23-25 2012.

Booch, G., Jacobson, I., Rumbaugh, J., 2000, OMG Unified Modeling Language Specification, http://www.omg.org/docs/formal/00-03-01.pdf

Department of Defense, 2009, DoD Architecture Framework, Version 2.0, Washington, DC: ASD(NII)/DoD CIO, 2009.

Friedenthal, S., Moore, A., Steiner, R., 2006, OMG Systems Modeling Language (OMG SysML™) Tutorial, Presented at the 2006 INCOSE (International Council on Systems Engineering) International Symposium, Orlando, FL, July 11, 2006.

Jackson, D., 2006, Software Abstractions: Logic, Language and Analysis, Cambridge, Massachusetts: The MIT Press, 2006.

Liskov, B., Zilles, S., 1974, Programming with abstract data types, ACM SIGPLAN Notices, Vol 9 Issue 4, pp. 50 – 59

Long, D., Scott, Z., 2011, A Primer for Model-Based Systems Engineering, 2nd. Ed, lulu.com, October, 2011.

Long, J.E., 2000, Relationships between Common Graphical Representations Used in System Engineering, Proceedings of the SETE2000 Conference (Systems Engineering and Test and Evaluation), Brisbane, Queensland, November 15-17, 2000.

NASA, 2007, Systems Engineering Handbook, NASA/SP-2007-6105 Rev1. Washington, D.C., December 2007.

Object Management Group, 2012, Systems Modeling Language Specification, version 1.3, http://www.omg.org/spec/SysML/1.3/PDF

Rivera, J., 2010, Software System Architecture Modeling Methodology for Naval Gun Weapon Systems, Doctoral Thesis, Naval Postgraduate School, December 2010.

Rivera Consulting Group, Eagle 6 Modeling, http://www.eagle6.com/

Vaneman, W., Jaskot, R., 2013, A Criteria-Based Framework for Establishing System of Systems Governance, Proceedings of the 7th Annual International IEEE Systems Conference, pp. 491-496, Orlando, FL, April 15-18, 2013.

Complex Adaptive Systems (CAS) Conference Presentation

Monterey Phoenix was presented at CAS on November 15, 2013. The full paper can be accessed from: http://www.sciencedirect.com/science/article/pii/S1877050913010739

SJPS Meeting Presentation

Monterey Phoenix was an invited presentation topic at the Southern Jersey Professional Societies meeting on January 15, 2014.

The Southern Jersey Professional Societies

American Institute of Aeronautics and Astronautics (AIAA), Institute of Electrical and Electronics Engineers (IEEE), International Council of Systems Engineers (INCOSE), International Test and Evaluation Association (ITEA) and National Society of Black Engineers (NSBE)

AIAA IEEE INCOSE ITEA

Presents

"Well, you didn't say not to! A formal systems engineering approach to teaching an unruly architecture good behavior"
Kristin Giammarco, PhD, Naval Postgraduate School

This presentation introduces a new formal modeling approach known as Monterey Phoenix (MP), which has features that enable prediction of emergent reactive system and system of systems (SoS) behaviors that result from interactions among subsystems and among the system and its environment. The approach emphasizes specification of component behavior and component interaction as separate concerns at the architectural level, consistent with well-accepted definitions of SoS. Implementing this separation of concerns in modeled architectures substantially increases the number of behaviors appearing in simulations for better prediction of design flaws and other latent behaviors. MP provides a new capability for automatically verifying system behaviors early in the lifecycle, when design flaws are most easily and inexpensively corrected. MP extends existing frameworks and allows multiple visualizations for different stakeholders, and has potential for application in multiple domains.

Wednesday Jan 15, 2014

Call in your reservations by noon on Tuesday, Jan 14th.

Mays Landing Golf and Country Club, Frasier Room
1855 Cates Rd. Mays Landing, NJ (641-4411)
Check their website for directions: mayslandinggolf.com

INCOSE SoS WG Presentation

NAVAL POSTGRADUATE SCHOOL

Monterey Phoenix was presented to the INCOSE SoS WG on January 17, 2014.

Systems of Systems Working Group
Webinar Series on Systems of Systems

Friday, 17 January 2014
11:00am – 12:00pm EDT

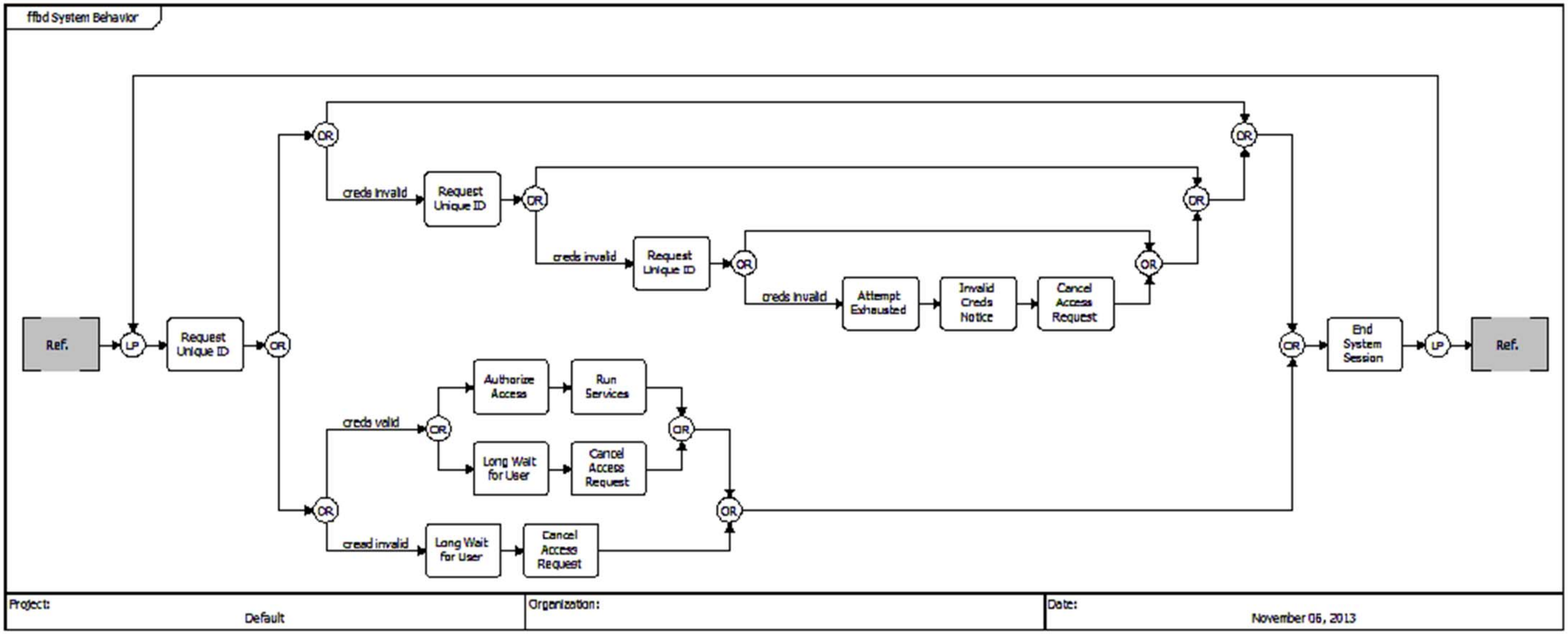
Behavioral Modeling of System Architectures

Architectural modeling and analysis are mechanisms that allow the capture of design decisions early in the process, so that they can be assessed and modified without incurring the costs of incorrect implementations. This presentation introduces a new formal modeling approach known as Monterey Phoenix (MP). MP has features that enable prediction of emergent reactive system and system of systems (SoS) behaviors that result from interactions among subsystems and among the system and its environment. It also provides a new capability for automatically verifying system behaviors early in the lifecycle, when design flaws are most easily and inexpensively corrected. The approach emphasizes specification of component behavior and component interaction as separate concerns at the architectural level, consistent with well-accepted definitions of SoS. Implementing this separation of concerns in modeled architectures substantially increases the number of behaviors appearing in simulations for better prediction of design flaws and other latent behaviors. An example is employed to highlight fundamental limitations of current modeling languages and approaches that hinder prediction of emergent behavior, and demonstrate how the application of MP enhances SoS modeling capability. This presentation will also address how MP can be used to inform quantifiable cost estimates (e.g. Function Point analysis) and ultimately project, program, and enterprise level resourcing decisions. MP is focused on behaviors, interactions, and automated tools, thereby enhancing and extending the software and systems engineer's toolkit of DoDAF, UML, and SysML to inform technical and resourcing decisions. For additional information on Monterey Phoenix and related work, the reader is invited to visit

- http://faculty.nps.edu/maugusto
- http://www.sciencedirect.com/science/article/pii/S1877050913010727
- http://www.sciencedirect.com/science/article/pii/S1877050913010739
- http://modeling.eagle6.com/,

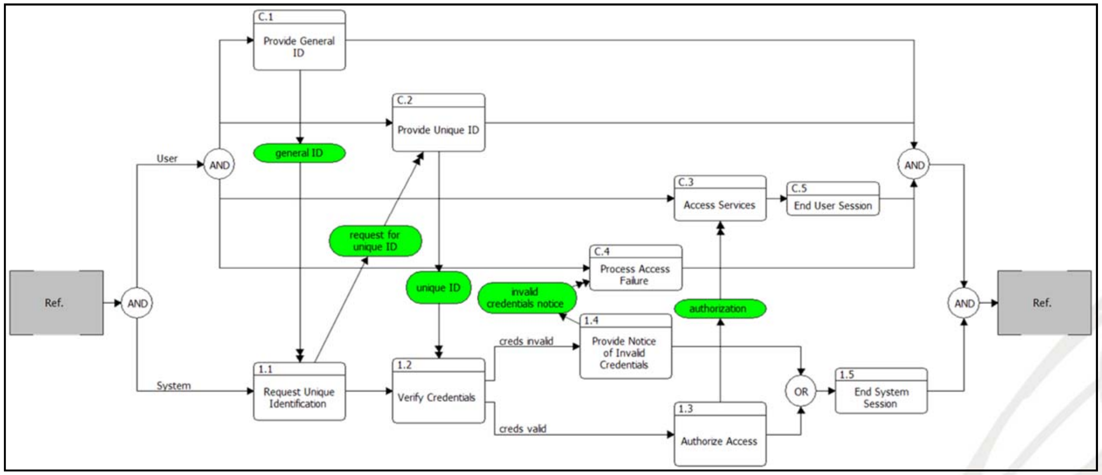
and to contact the presenters directly.

System Behavior (FFBD Corresponding to ROOT System MP Code))



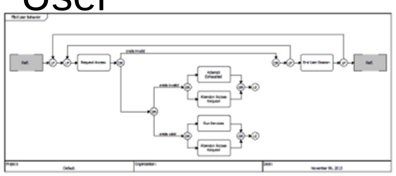
It Makes a Big Difference

Modeling the System in the Context of Its Environment

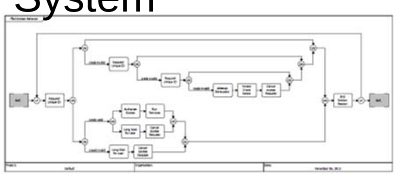


Modeling the System and Its Environment in the Context of One Another

User



System



User/System Interaction Specification

Branches shared by components:
creds valid *shared by* User and System
creds invalid *shared by* User and System
Functions shared by components:
Attempt Exhausted *shared by* User and System
Run Services *shared by* User and System

creds valid → **creds invalid** →

Attempt Exhausted **Run Services**