Calhoun: The NPS Institutional Archive

DSpace Repository

Faculty and Researchers | Faculty and Researchers Collection

2009

Monterey Phoenix: modeling software and systems architecture

Auguston, Mikhail

Monterey, California, Naval Postgraduate School

http://hdl.handle.net/10945/36771

Downloaded from NPS Archive: Calhoun

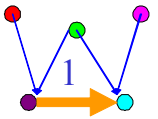
http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

Monterey Phoenix: modeling Software and Systems Architecture

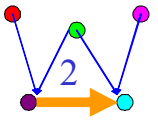
Mikhail Auguston
Computer Science Department
Naval Postgraduate School
Monterey, California, USA



One of the major concerns in System and Software Architecture design is the question of the behavior of the system.

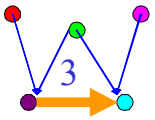
For system's architecture one of the challenges is to predict emerging behavior of system of systems.

We suggest an approach for building system behavior models based on the concepts of event and event traces and tools for architecture verification.



Monterey Phoenix

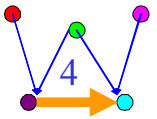
- An approach to formal software and system architecture specification based on **behavior models**.
- The behavior of the system is defined as a set of events (**event trace**) with two basic relations: **precedence** and **inclusion**.
- The structure of event trace is specified using **event grammars** and other constraints organized into schemas.
- The **schema** framework is amenable to stepwise architecture refinement, reuse, composition, visualization, and application of automated tools for consistency checks.



Basic Concepts

Event - any detectable action in system's or environment's behavior

Event trace - set of events with two basic relations, **precedence** (PRECEDES) and **inclusion** (IN)



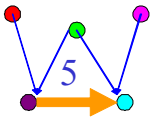
Data items as behaviors

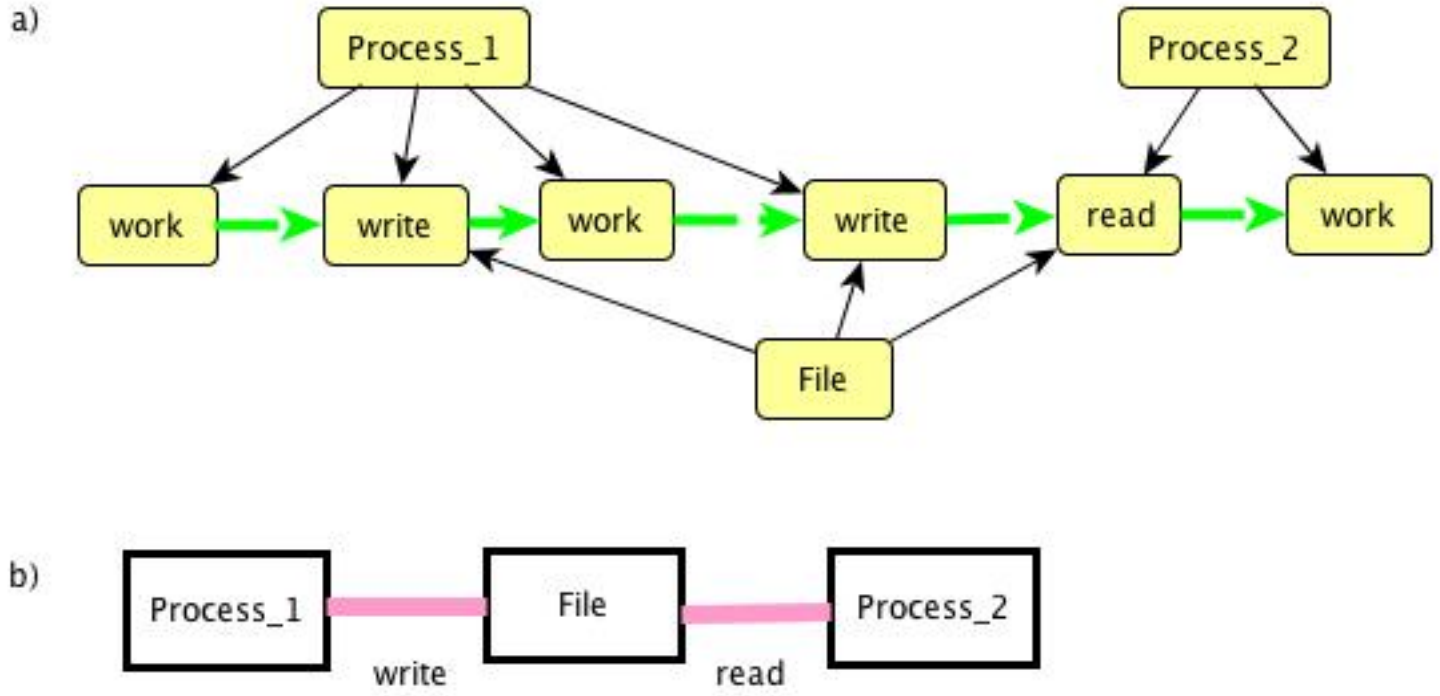
```
SCHEMA Data_flow

ROOT Process_1:      (*  work   write  *);
ROOT Process_2:      (* ( read | work ) *);
ROOT File:           (* write *)  (* read *);

Process_1, File  SHARE ALL write;
Process_2, File  SHARE ALL read;
```

The schema defines a set of event traces,
i.e. the **behavior model**



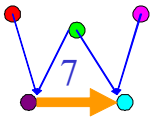


a) An example of composed event trace for the Data_flow schema.
b) An architecture view for the Data_flow schema.

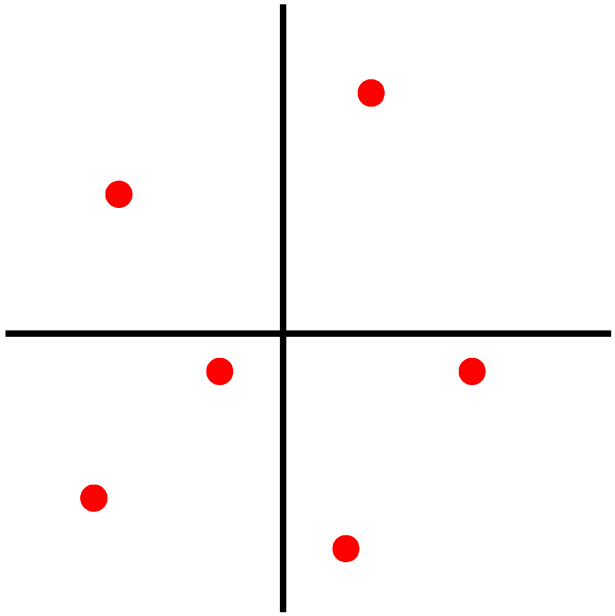
Architecture verification

Advantages of Monterey Phoenix approach compared with the common simulation tools are as follows.

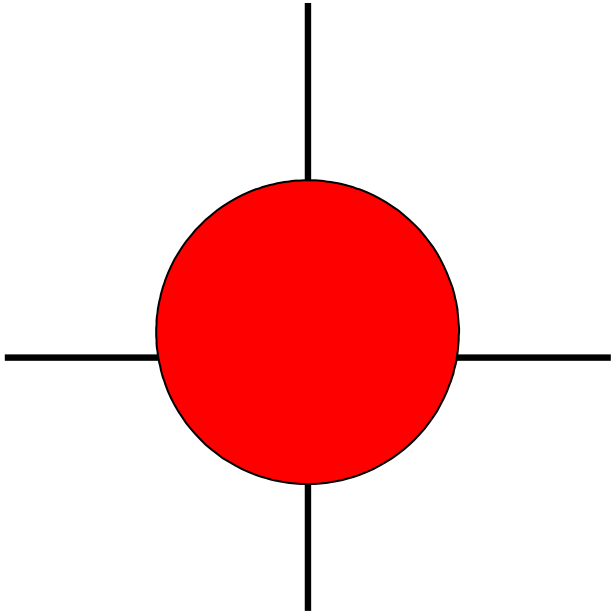
- Means to write **assertions** about the system behavior and tools to verify those assertions.
- **Exhaustive search** through all possible scenarios (up to the scope limit). **The small scope hypothesis** states that most of errors can be demonstrated on small examples.
- The support for **verifiable refinement** of the architecture model, up to design and implementation models.
- Integration of the architecture models with **environment models** for defining typical scenarios (use cases) and verifying system's behavior for those scenarios.



Architecture verification within limited scope



Testing:
A few cases of arbitrary size



Scope-complete:
All cases within a small bound

A model of system interacting with its environment and assertions about the system's behavior.

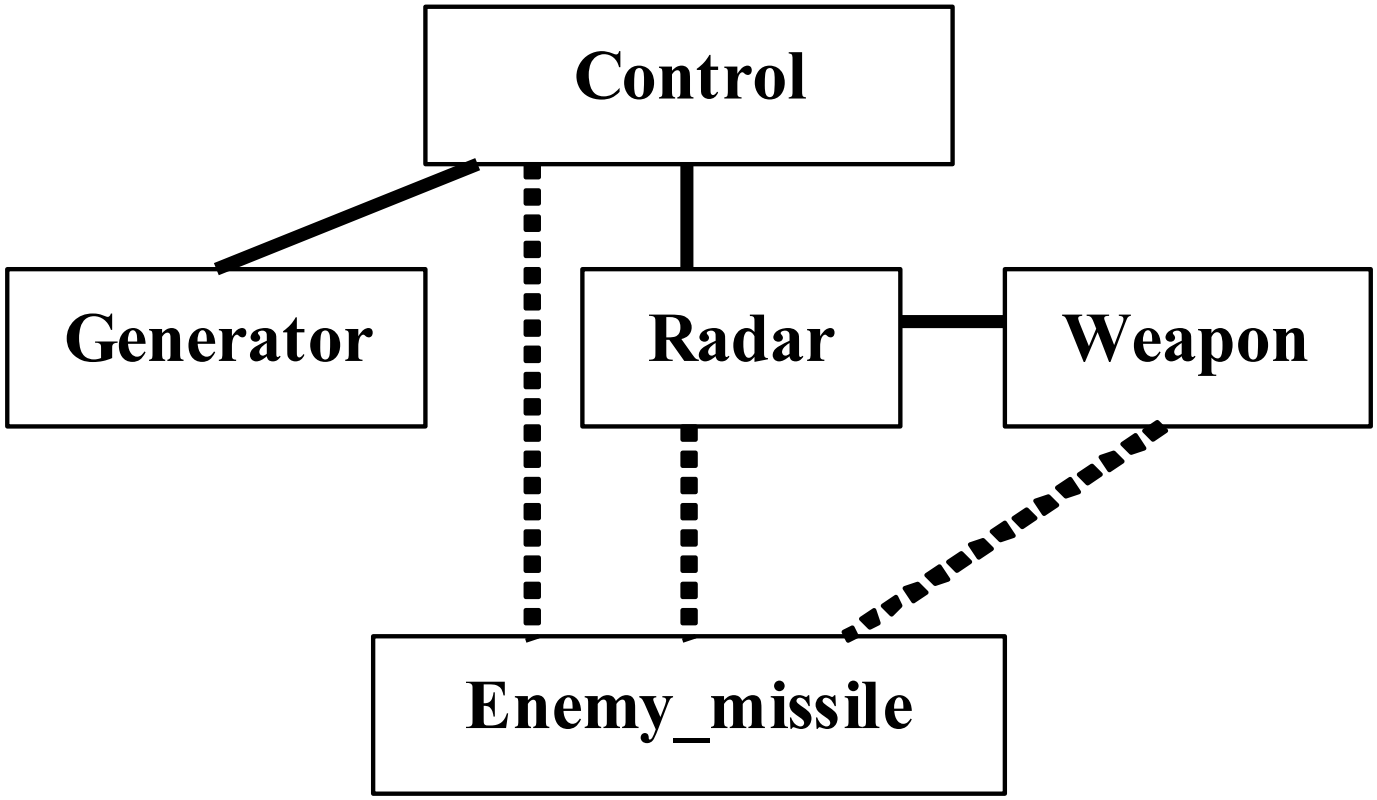
```
ROOT Generator: (*      Idle  Generator_On
                  Generating  Generator_Off *)
        WHEN{ Generator_hit =>
              Generator_Off  Repair   RESTART };

ROOT Radar: (*  Idle    Radar_On
                Radar_Working  Radar_Off  *)
        WHEN{ Radar_hit =>
                Radar_Off  Repair  RESTART };

Radar_Working: (*(Target_detected | No_target)*);

Target_detected: Weapon_On;
… …   …
```

A view of the system's components and environment interaction



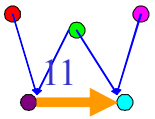
Assertions about the system's behavior

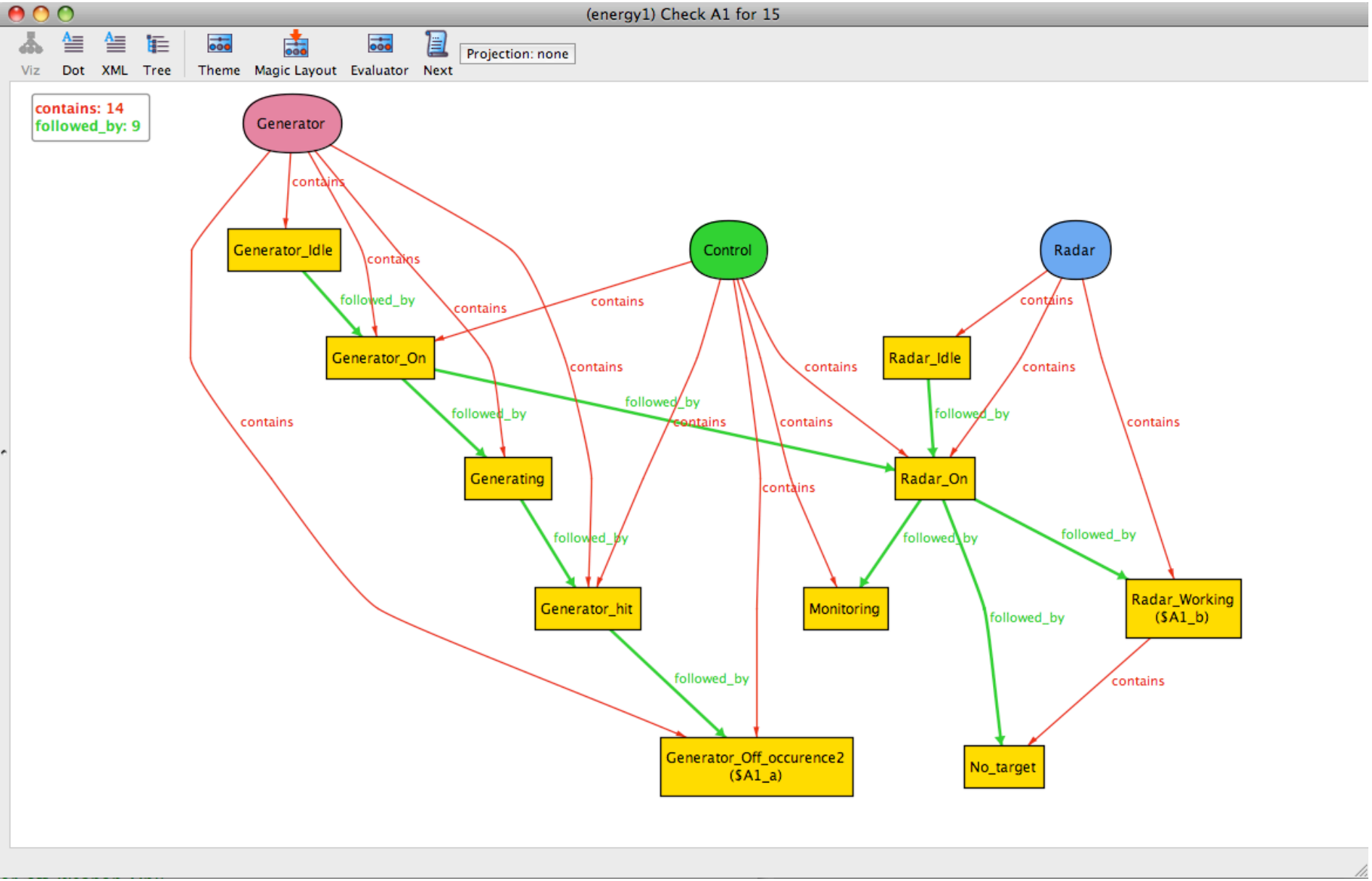
Assertion 1.

```
not exists Slice(Generator_off, Radar_Working);
```

The assertion above can be refuted on a relatively small counterexample of a trace.

The following counterexample for Assertion 1 has been found by Alloy Analyzer in less than 2 sec on iMac workstation with 2.8 GHz processor.




(energy1) Check A1 for 15
Viz
Dot
XML
Tree
Theme
Magic Layout
Evaluator
Next
Projection: none
contains: 14
followed_by: 9
Generator
Generator_Idle
Generator_On
Generating
Generator_hit
Generator_Off_occurence2
($A1_a)
Control
Monitoring
Radar
Radar_Idle
Radar_On
Radar_Working
($A1_b)
No_target
contains
followed_by

Work in progress

- Implementation of MP -> C++ generator to support **generation and analysis of event traces** with hundreds and thousands of events
- Methodology, guidelines, and a representative collection of **reusable templates** for Phoenix framework.
- A broad and flexible assortment of schema **composition operators.**
- Libraries of predefined predicates, functions, and tools to extract and **visualize views**.
- Given duration and frequency estimates for events within components and connectors it becomes possible to **estimate throughput and latency**.
- **Environment models** and Business Process Models for Systems Engineering.
- Potential of applying **model checking** power to a reasonable subset of Phoenix schemas.
- **Dynamic** and evolving architectures
- **Meta-architecture**, software product lines, domain-specific architectures

On-line MP prototype Eagle6: **http://www.Eagle6.com**