Model Comparison for Fault and Attack Detection

Dr. Christopher Landauer
Software Systems Analysis Department
The Aerospace Corporation

Computer Science Division / Software Engineering Subdivision
07 November 2012

Summary

- We show how to detect anomalous behavior in software-intensive systems by comparing models of observed and expected behavior.

chris.landauer@aero.org

Issues

- Determining expected behavior
- Determining actual behavior
- Detecting and understanding behavior differences
- Diagnosing differences as fluctuations or trends
  - *Fluctuations can be attacks or transients*
  - *Trends can be attacks or normal degradation*
  - *All of these are anomalies*
  - *Anomalies are not necessarily errors or attacks*

chris.landauer@aero.org

Outline

- Anomalous Behavior Is Only Defined in Reference to Expected Behavior
  - *Expectation Modeling*
  - *Behavior Modeling*
- Model Comparison
  - *Anomaly Detection*
  - *Attack Detection*
- Enabling Technologies
  - *Computational Reflection*
  - *Inductive Inference*
  - *Model Comparison*
- Experiments with CARS Testbed
- Conclusions and Prospects
- Research Topics
- References

chris.landauer@aero.org

Anomaly Detection in Software-Intensive Systems

- An Anomaly is Something Unusual or Unexpected
  - *Anomalies are not necessarily errors or attacks*
- Anomalous Behavior Can Only be Defined in Reference to Expected Behavior
  - *Expected could mean required behavior*
  - *Expected could mean normal behavior*
  - *Both require behavior observation*
  - *Both require behavior models*
- Many Different Kinds of Methods to Produce Expected Behavior
  - *Imposed models*
  - *Logical methods*
  - *Empirical methods - see reference cprb*

chris.landauer@aero.org

Expectation Modeling

- System Specification and CONOPS Map into Behavioral Expectations
  - *Expected situations*
  - *Expected response behavior*
    - Can be written as context-sensitive rules
  - *Prohibited or deprecated behavior*
- This Generally Requires Better Specifications than We Currently Use
  - *More abstract - intensional instead of extensional*
  - *More precise about applicability conditions*
- Need an Appropriate Dynamic Behavior Notation
  - *Include component definitions and structures*
    - Connections and interferences
  - *Include temporal constraints*

chris.landauer@aero.org

Behavior Modeling in Software-Intensive Systems

- Observe Internal and Interface Behavior
  - *Infer models of behavior sequences*
    - Maybe even partially ordered sets
  - *Also Construct Stochastic Models when Necessary*
- Data Required
  - *We need deep and detailed instrumentation of the system and its environment*
- Processes Required
  - *We need good model assessment methods to determine whether the models reflect the behavior properly*
- We Can Use the Same Components as in the Expectations
  - *The issue is not to compare the system structure*
  - *It is to compare the behavior of that structure*

chris.landauer@aero.org

Model Comparison

- Event Structure Matching
  - *Examine and explain differences*
- Detect Model Differences
  - *Differences point to potential problems*
  - *Unexpected events*
  - *Expected events that did not occur when they were expected*
  - *Unexpected inactivity (partial deadlock)*
- Predictable Explanations
  - *Many of these differences have predictable explanations*
  - *Incorrect conditions for an event path or pattern*
  - *Insufficient conditions for continuing an event path*
- Identifiable Conditions
  - *The conditions that did (or did not) occur can be identified*
  - *Sometimes the system can make the changes; sometimes not*
  - *Either way it can announce the problem*

chris.landauer@aero.org

Enabling Technologies

- Specifications Provide the Expectation Targets
  - *Deduction and elaboration of expectations into activities*
- Computational Reflection Provides the Basic Data
  - *We can know more than just what was expected*
  - *We can know why*
- Inductive Inference Provides the Empirical Models
  - *Identify commonalities and distinctions*
  - *Infer hidden states and discriminators*
- Model Comparison
  - *Detect (easy) and explain (hard) what is different*

chris.landauer@aero.org

Computational Reflection

- This Approach Avoids the Problem of ``quis custodiet ipsos custodes'' (who will watch the watchers)
- Because there is no Separate Watcher
  - *The system watches itself*
- This is Enabled by the Wrapping integration Infrastructure - see reference gppe
- Self-Modeling Systems Model all Computational Processes
  - *They model the infrastructure processes also - see reference sms*
- The System Contains Models of Its Own Structure and Behavior
  - *Including processes that examine and manipulate those models*
  - *All expected behavior comes from the models via model interpreters*
  - *The interpreters are resources and are Wrapped and replaceable*

chris.landauer@aero.org

Wrapping Integration Infrastructure

- Problem Posing Programming Paradigm
  - *Separate functionality into ``problem'' and ``resource''*
    - Information service request
    - Information service provider
  - *Still need a method to map one to the other*
- Wrapping Knowledge Bases
  - *Map one to the other using a Knowledge Base*
  - *Problem Specification and Context-dependent*
- Problem Managers
  - *Read WKB and select and apply resources to problems*
  - *They are also resources; they are also Wrapped*
- Many Applications
  - *Code ReUse without modification*
  - *Migration to standards*

chris.landauer@aero.org

Scenario-Based Engineering Process

- SBEP is an Ideal Development Process for This Approach
  - *See references sbep and sbep(more)*
  - *Expectations are explicitly retained*
    - Users, developers, and other stakeholders
    - Concepts of operation derived from scenarios of expected use
  - *Roles and responsibilities*
    - Functional components have roles in cooperation and responsibilities locally
    - These are NOT necessarily the same as structural components
  - *Activities*
    - Behavioral units demonstrably fulfilling the roles and responsibilities
- Then Map Activities to Problems and Construct Resources
  - *The resources are the structural components*
- We have shown this to be effective for cyber-physical systems - see reference cars

chris.landauer@aero.org

Inductive Inference

- Many Methods are Known (we have used them in many systems)
  - *Sequence structure models*
    - Regular and context-free grammars
  - *Stochastic Models with Effective Inference Processes*
    - Stochastic state machines, Hidden Markov models
- Sequence Models - Grammatical Inference
  - *Encapsulate common sequences (syntagmatic)*
  - *Generalize common events across multiple instances (paradigmatic)*
    - Including alternatives in similar contexts
- Use Context as a Discriminator
  - *Need context measurement and identification*
  - *Intention can be part of context*
- Timing and Event Correlation
  - *Sequence and partially ordered event pattern correlations*

chris.landauer@aero.org

Semiotics and the ``Get Stuck'' Theorems

- Semiotics
  - *All of these modeling methods depend on the choice of basic symbols*
  - *Computational Semiotics is the study of the use of symbols by computing systems*
- How to Label Events Appropriately
  - *Defined by activity expectations or observations*
  - *Different partitions of events into classes*
    - Clustering by co-occurrence or proximity
  - *Different model alphabets mean different structures*
- The ``Get Stuck'' Theorems
  - *Limit how much variety a fixed finite notation can express usefully*
    - Eventually it becomes too unwieldy to compute (it gets ``stuck'')
  - *Limit how much knowledge a finite representational mechanism can express*
    - Even when humans do the modeling
  - *Both theorems cause us to focus attention on the act of making models and symbols for them*

chris.landauer@aero.org

Model Comparison

- Some Methods are Known (we have used them on some systems)
  - *Structure Matching*
  - *Dynamic time warping (used for speech recognition)*
  - *Differential Diagnosis*
- Event Correlation
  - *The whole process is simplified here because the basic observational data and the activity use the same terms (the time-stamped resource applications)*
- Table Parser (a primitive chart parser)
  - *Match potential goals to observed events using a general context-free grammar*
  - *Generalized to partially ordered sets of event patterns*
  - *Immediate notification of missing or unassigned events*
  - *Can also be used for model assessment*
- Correction Process
  - *Detect Differences, Announce Them, Fix them if possible (leads to a notion of error-correcting structures)*

chris.landauer@aero.org

CARS Testbed

- Computational Architectures for Reflective Systems
  - *A testbed for distributed cooperative behavior of embedded systems - see reference cars*
  - *A collection of radio-controlled toy cars with extra sensors and computing*

- Even though cars are not spacecraft, the important issues of what modeling is appropriate, how to generate models from goals and from observations, and how to compare dynamic models are all the same (and the cars are MUCH cheaper).

- CARS System Structure
  - *Theoretical and experimental platform*
    - Software in the cars is identical in the simulation, written in *wrex*
  - *Multiple levels of simulation fidelity*
  - *Continual model validation*

chris.landauer@aero.org

Experiments with CARS (1)

- We are conducting experiments in model inference in CARS
  - *First in the simulator (theoretical platform)*
  - *Eventually in the field (experimental platform)*
- In the context of simple movement in a relatively flat space
  - *Just mapping and modeling the environment as the car explores*
    - Various movable objects and fixed obstructions
  - *Are the expectation models we provide consistent with the empirical models it infers?*
  - *Are the predictive models it builds substantiated by the subsequent behavior?*
  - *What inference methods are most effective for this (relatively benign) environment?*
  - *Are surprises properly identified and handled?*

chris.landauer@aero.org

Experiments with CARS (2)

- In the context of several cooperative games
  - *Tag is the first one we use*
    - One car is It; the others are NotIt
    - Once tagged, the roles interchange (but the new It must wait a few seconds)
  - *Do the same model building methods work in this more complex environment?*
  - *All the same questions as above*
  - *Other games are follow the leader, soccer practice, and push the BIG box – see reference cars*

chris.landauer@aero.org

Conclusions and Prospects

- We have observed that many techniques for inference of information from data have been developed
- We believe that they are ready to be used in and by autonomous embedded computing systems
  - *To help us help them behave appropriately in complex environments*
  - *To help us understand the difficult parts of making models autonomously*
- We described some of these methods and the kinds of models they can produce
  - *What kinds of data they expect*
  - *What assessment methods they require*
  - *How to implement them within our computational systems using Wrappings*
- The SBEP produces our expectation models as a byproduct of system development
  - *Models of resource usage patterns in context*
- We are planning and conducting experiments in a real system testbed (CARS)

chris.landauer@aero.org

Research Topics

- There are many aspects of modeling cyber-physical systems that need more research and better methods
- Formal Modeling Notations
  - *Events and activities*
  - *Intentions*
  - *Hypotheticals and counterfactuals*
- Model Inference Methods
  - *More expressive model definition formalisms*
  - *More powerful model inference mechanisms*
- Other Computational Methods
  - *Computational abstraction, naming and other semiotic processes*
  - *Differential and / or deficiency diagnostics*
- System Designers
  - *System design process descriptions*
  - *Guidance on specifications*
  - *Guidance on instrumentation*

chris.landauer@aero.org

Some References

- **cars:** Dr. Kirstie L. Bellman, Dr. Christopher Landauer, Dr. Phyllis R. Nelson, ``Managing Variable and Cooperative Time Behavior'', Proceedings SORT 2010: The First IEEE Workshop on Self-Organizing Real-Time Systems, 05 May, part of ISORC 2010: The 13th IEEE Intern. Symposium on Object / component / service-oriented Real-time distributed Computing, 05-06 May 2010, Carmona, Spain (2010)

- **cprb:** Christopher Landauer, ``Correctness Principles for Rule-Based Expert Systems'', pp. 291-316 in Chris Culbert (ed.), Special Issue: Verification and Validation of Knowledge Based Systems, Expert Systems With Applications Journal, Volume 1, Number 3 (1990)

- **gppe:** Christopher Landauer, Kirstie L. Bellman, ``Generic Programming, Partial Evaluation, and a New Programming Paradigm'', Chapter 8, pp. 108-154 in Gene McGuire (ed.), Software Process Improvement, Idea Group Publishing (1999)

chris.landauer@aero.org

Some More References

- **sms:** Christopher Landauer, Kirstie L. Bellman, ``Self-Modeling Systems'', p. 238-256 in R. Laddaga, H. Shrobe (eds.), ``Self-Adaptive Software'', Springer Lecture Notes in Computer Science, Volume 2614 (2002)

- **sbep:** Karen McGraw and Karan Harbison, User-centered Requirements: The Scenario-Based Engineering Process, Lawrence Erlbaum (1997)

- **sbep (more):** Dr. Christopher Landauer, ``Problem Posing as a System Engineering Paradigm'', Proc. ICSEng 2011: The 21st Intern. Conf. on Systems Engineering, 16-18 August 2011, Las Vegas, Nevada (2011)

chris.landauer@aero.org

AEROSPACE
Assuring Space Mission Success

Questions?