

Using Requirements Engineering to Track Down Medical Errors

Lori A. Clarke

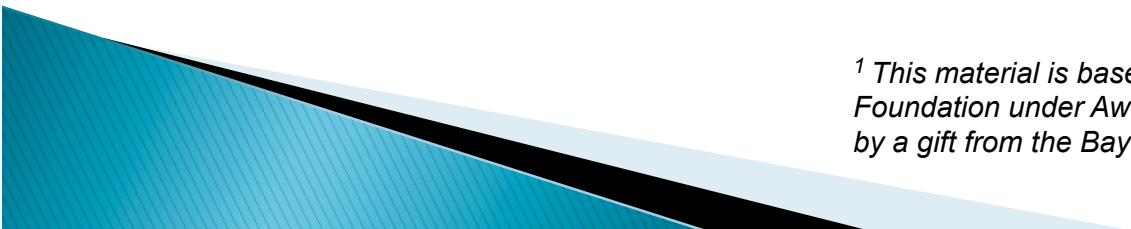
Department of Computer Science

University of Massachusetts Amherst

In collaboration with

George S. Avrunin and Leon J. Osterweil

¹ This material is based upon work supported by the National Science Foundation under Awards CCF-0820198, CCF-0905530 and IIS-0705772, and by a gift from the Baystate Medical Center, Rays of Hope Foundation.



A Crisis in Health Care

- ▶ ~ 100,000 deaths per year due to **avoidable** errors in the US [IOM1999]
- ▶ Widely believed to be an underestimate
 - Perhaps by a factor of two
- ▶ Hundreds of billions of dollars wasted
- ▶ Incalculable amount of human pain and suffering



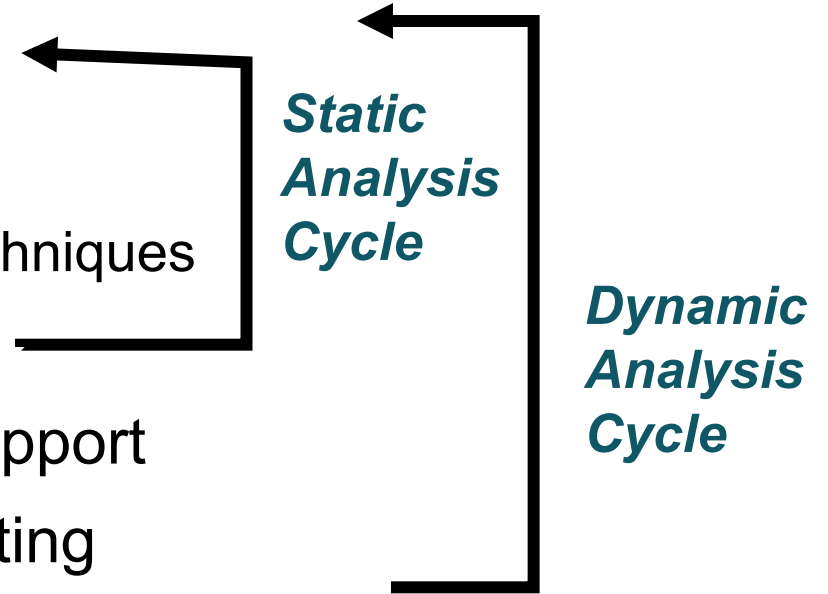
Current trends

- ▶ Little improvement since 1999
- ▶ 2009 National Research Council Report
 - *"persistent problems do not reflect incompetence on the part of health care professionals - rather, they are a consequence of the **inherent intellectual complexity** of health care taken as a whole and a medical care environment that has not been adequately structured to help clinicians avoid mistakes or to systematically improve their decision making and practice."*
- ▶ As the use of technology increases,
 - Some kinds of errors are expected to decrease (RFID scanners to verify patient ID)
 - But others are expected to lead to even more complexity

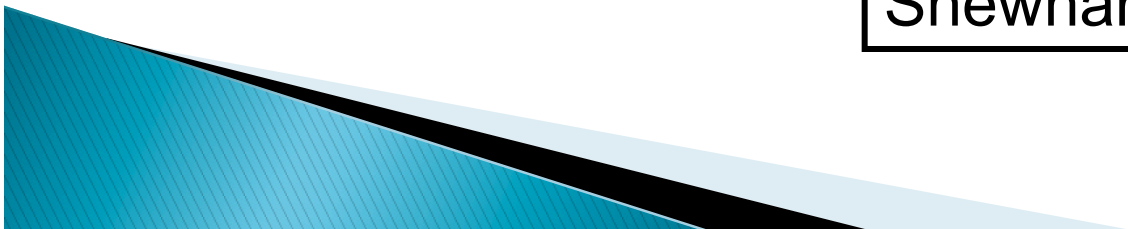


Our Approach: Technology Support for *Continuous Process Improvement*

- ▶ Model processes
- ▶ Evaluate them
 - Using a wide variety of analysis techniques
- ▶ Propose modifications
- ▶ Deploy them: Process-guided support
- ▶ Reevaluate based on clinical setting data and iterate



Shewhart/Deming Cycle



Static Analysis Cycle: Employ Technologies Designed to Model and Analyze Complex Software Systems

- ▶ **Process programming to model medical processes**
 - Little-JIL process programming language
- ▶ **Requirements engineering to capture properties**
 - PROPEL (property elucidation system)
- ▶ **Model checking to detect errors**
 - FLAVERS (Flow Analysis for Verifying Systems) and SPIN
- ▶ **Safety analysis to reveal vulnerabilities**
 - Failure Mode and Effects Analysis
 - Fault tree Analysis
- ▶ **Discrete event simulation to improve efficiency**



On-going Case Studies

- ▶ Breast Cancer Chemotherapy
 - Wilson Mertens, Director, Baystate Medical Center (BMC) Oncology Dept
 - ▶ In-Patient Blood Transfusion Process
 - Beth Henneman, UMass School of Nursing
 - ▶ Emergency Department Patient Flow
 - Phil Henneman, BMC ED physician, former director
 - ▶ Patient-controlled Analgesia Infusion Pump
 - Julian Goldman, Partners Health and Mass General Hospital
 - ▶ Coordination of Care
 - Kate Rankin, Memory and Aging Center, U. CA San Francisco
 - ▶ Cardiac Surgery
 - Marco Zenati, VA and Mass General Hospital
- 

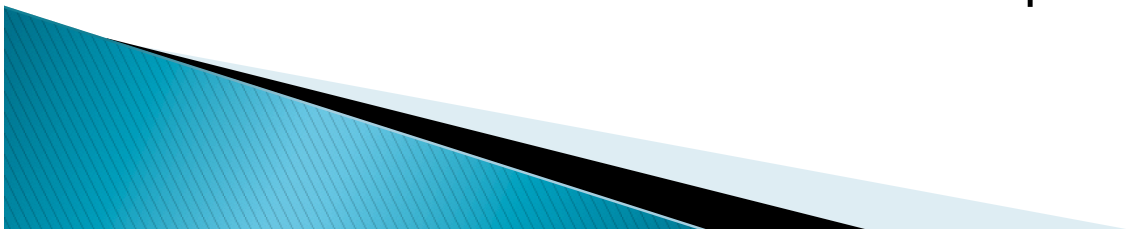
Defining Processes

▶ **Medical processes**

- Complex, concurrent, and exception-rich
- Need to support human choice, flexibility
- Each task may involve multi-processors and each processor is involved in multiple tasks

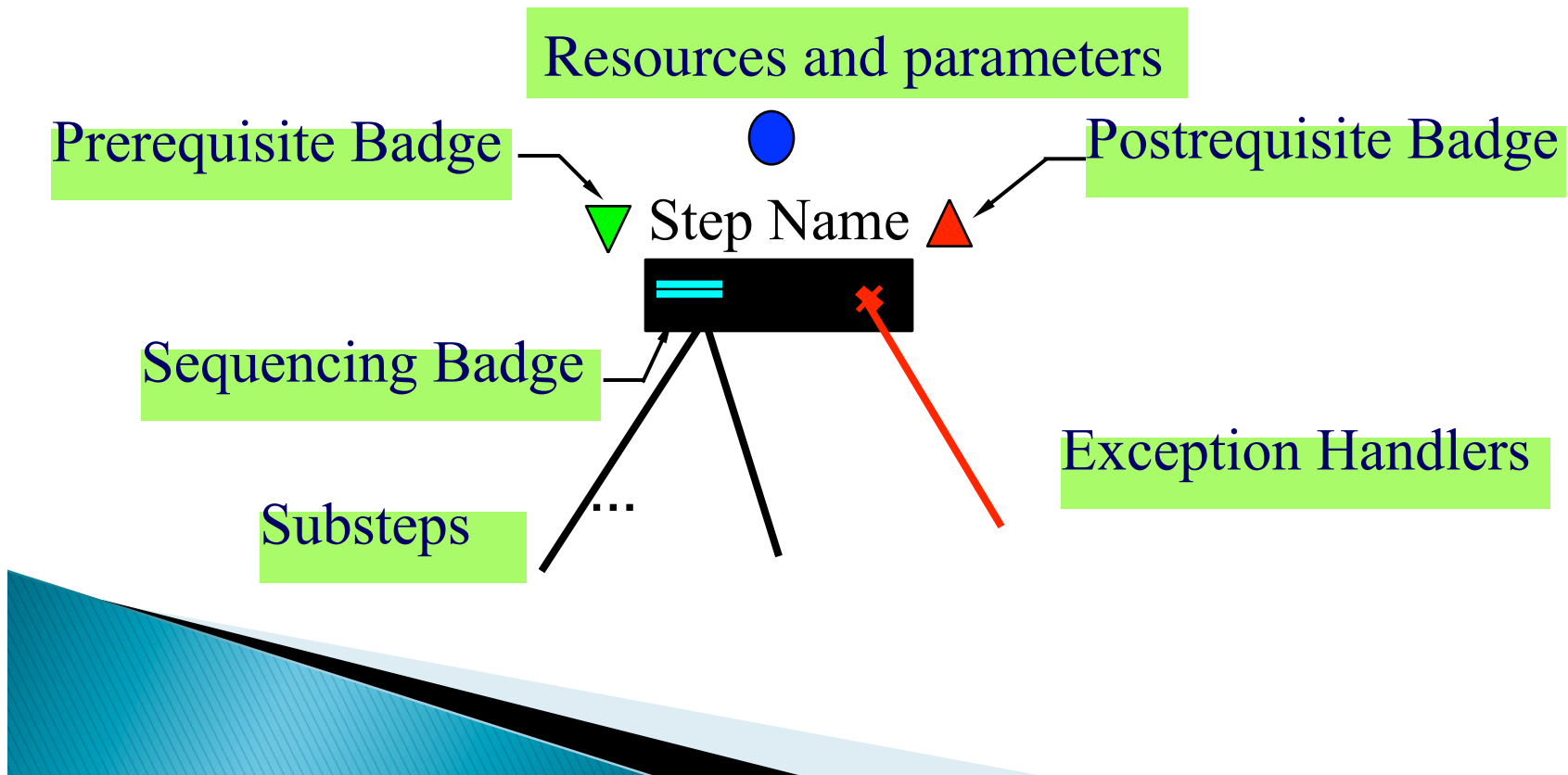
▶ **Process language requirements**

- Capture complexity of medical processes clearly, cleanly
- Rich semantics
(e.g. functionality, timing, resource utilization, exceptions)
- Precise enough to support static analysis, simulations, and executions
- Understandable to a medical professional

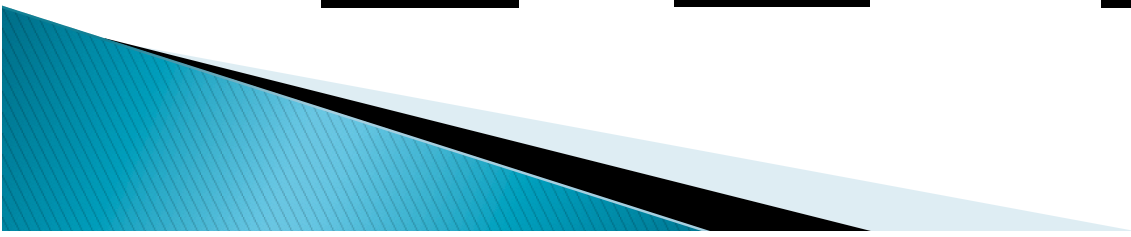
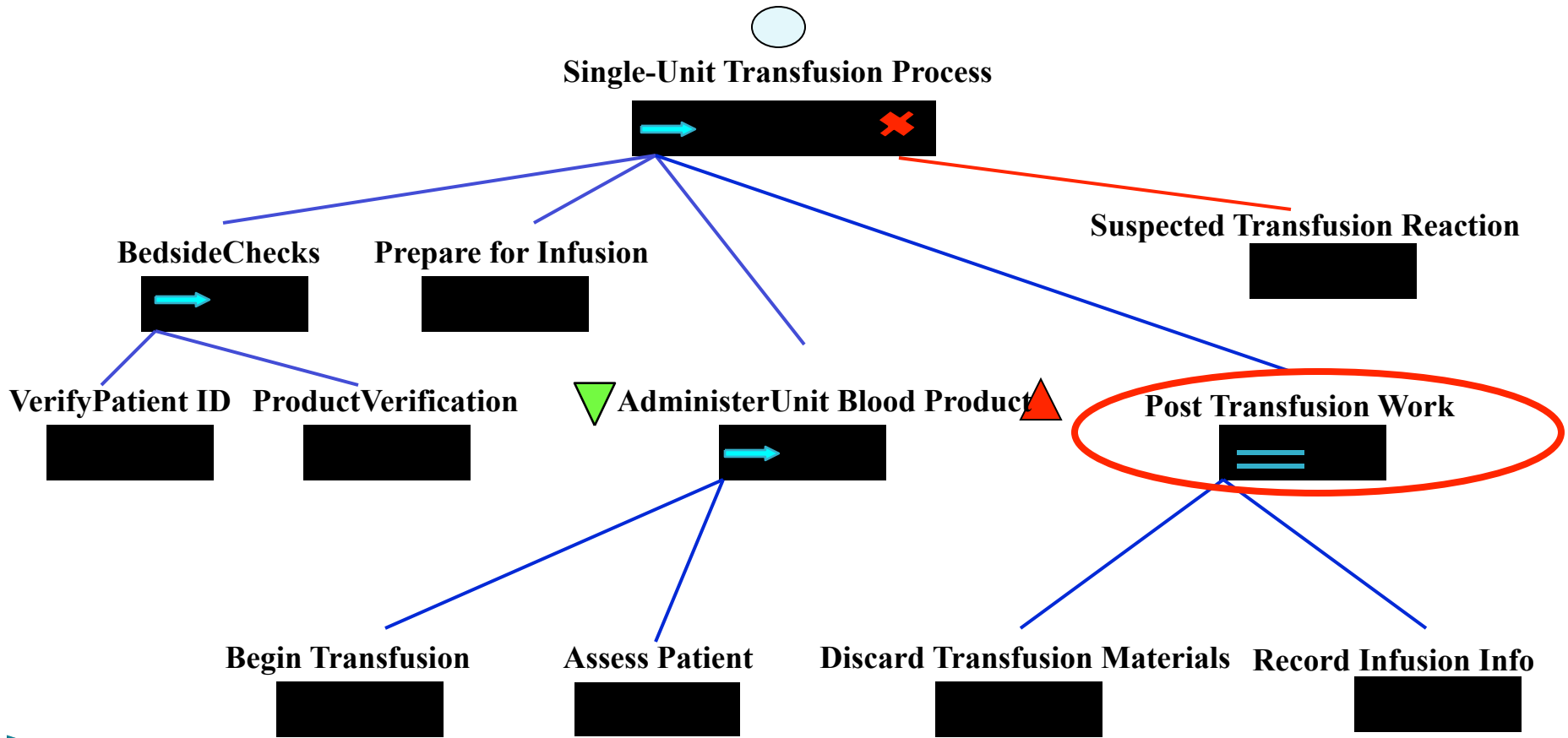


Little-JIL Overview

- ▶ Visual language for coordinating tasks
- ▶ Uses hierarchical step decomposition
- ▶ Step icon



Single-Unit Transfusion Process



Narrative View and ToC

3. Order Test(s)

3.1 order test(s) on computer

3.1.1 log into computer

3.2.1 select patient record

3.2.1.1 look for patient name on the alphabetical list

3.2.1.2 match additional info as needed (age, gender, complaint, location...)

- ...

3. **Order Test(s)**(part of [perform Blood Specimen Labeling process](#))

To perform this step the *Provider* must have the *patient-name*.

The *Provider* should first [order test\(s\) on computer](#),
and then [order test\(s\) on patient chart](#).

During any of these steps, if the required resources are not available, *order test(s)* is considered to have failed.

Upon successful completion of this step,
continue to [perform Blood Specimen Labeling process](#) by proceeding to the next step in the sequence.

3.1 **Order Test(s) on Computer** (part of [order test\(s\)](#))

To perform this step *the Provider* must have *the patient-name* and the *CIS system*.

To order test(s) on computer the *Provider* should perform, in order, each of the following:

[log into computer](#)

[select patient record in DB](#)

[verify the selected patient exactly matches desired patient](#)

[select test to order](#) at least once

[digitally sign the order\(s\)](#)

During any of these steps, if the required resources are not available, *order test(s) on computer* is considered to have failed.

Upon successful completion of this step, continue [order test\(s\)](#) by proceeding to the [order test\(s\) on patient chart](#) step.

Process Model Elicitation

- ▶ **Processes are not well-understood**
 - Individuals know their process, but misunderstand how it relates to others
e.g., Artifacts created but not used
- ▶ **Even “simple” processes may be very complex**
 - E.g., Verify Patient ID
 - Need abstraction and hierarchical decomposition
- ▶ **Clinicians are not computer scientists**
 - Think in terms of scenarios, not in terms of the general process
 - Initially are scared by the language but many end up discussing the lowest-level details “comfortably”
 - Like the natural language representation (and so do the computer scientists)



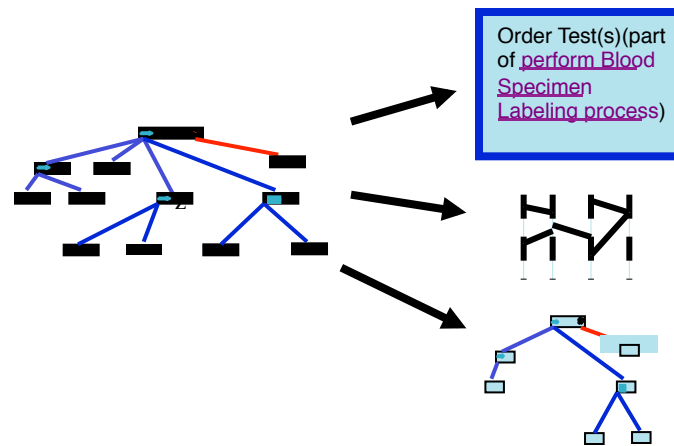
Process Model Elicitation

- ▶ **Aspects of the language helped guide the elicitation**
 - E.g., What exceptions can arise and how are they treated?
- ▶ **Important to tie down terminology**
 - Use the same term in different ways
e.g., “transfuse blood”
 - Use different terms to mean the same thing
e.g., verify, check, confirm, match
- ▶ **Takes many iterations to model a process**
 - Must determine upper and lower bounds of the scope
 - Must determine granularity of tasks
 - 2 elicitors - help keep discussion on track; listen for requirements; record all sessions

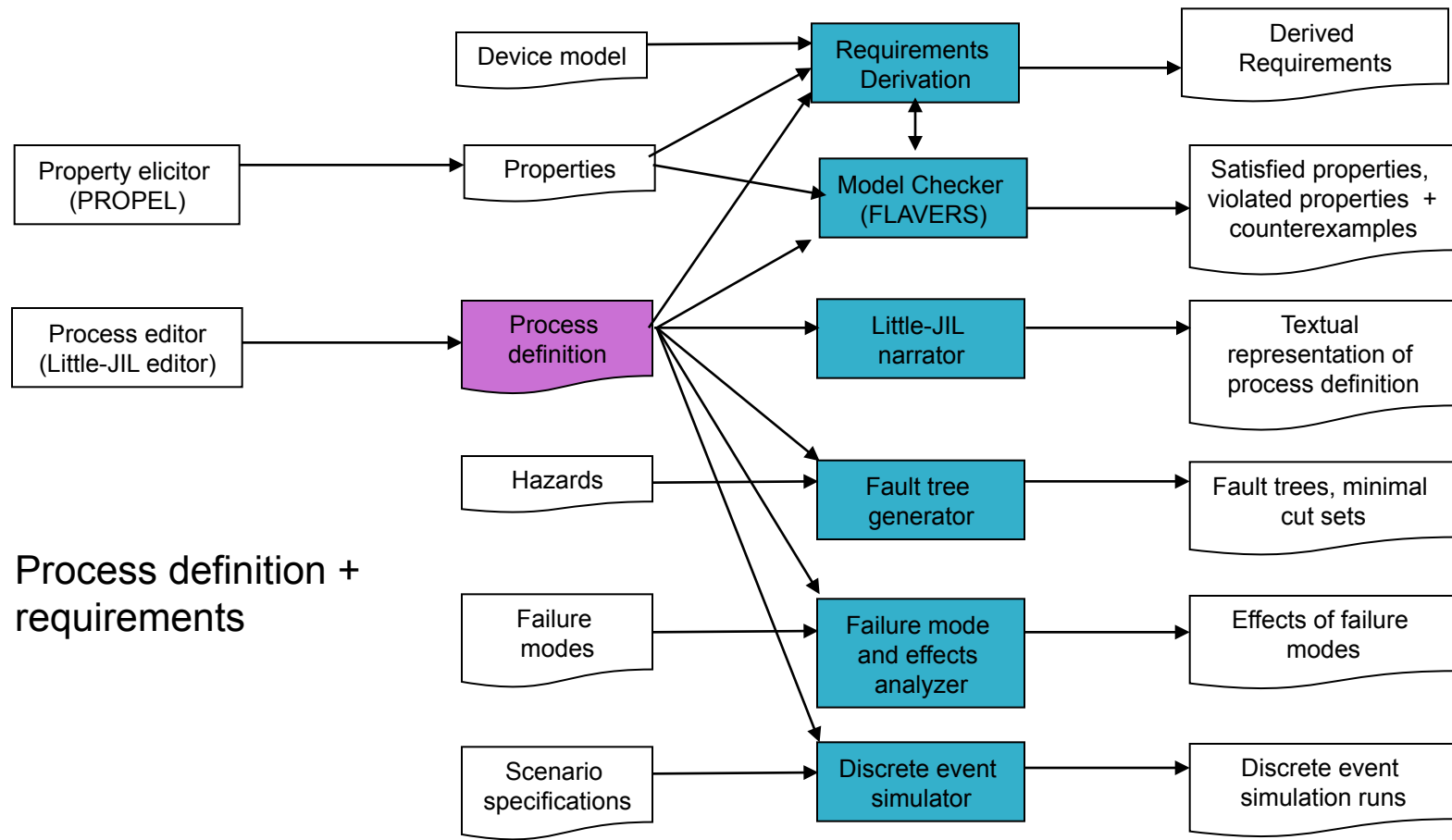


Process Modeling Observations

- ▶ There is no **best** process modeling representation
 - Large graphical or textual representations are hard to comprehend
- ▶ Want a representation that will
 - **Be the basis for answering questions about the model**
 - Support the creation of other representations
 - E.g., dependency matrix, data flow diagrams, role-based descriptions, textual descriptions, scenarios...



Process Improvement Environment



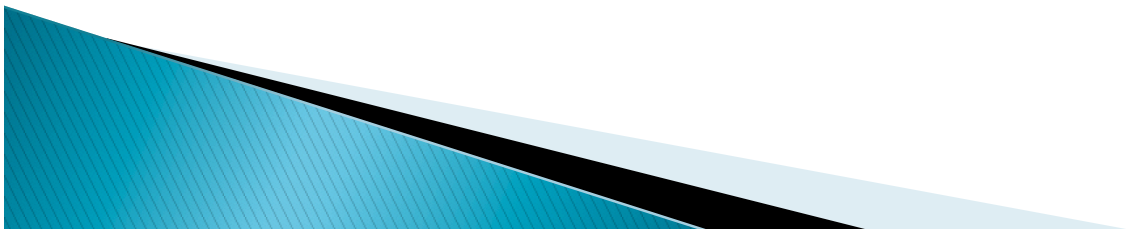
Process definition + requirements

Static Analysis

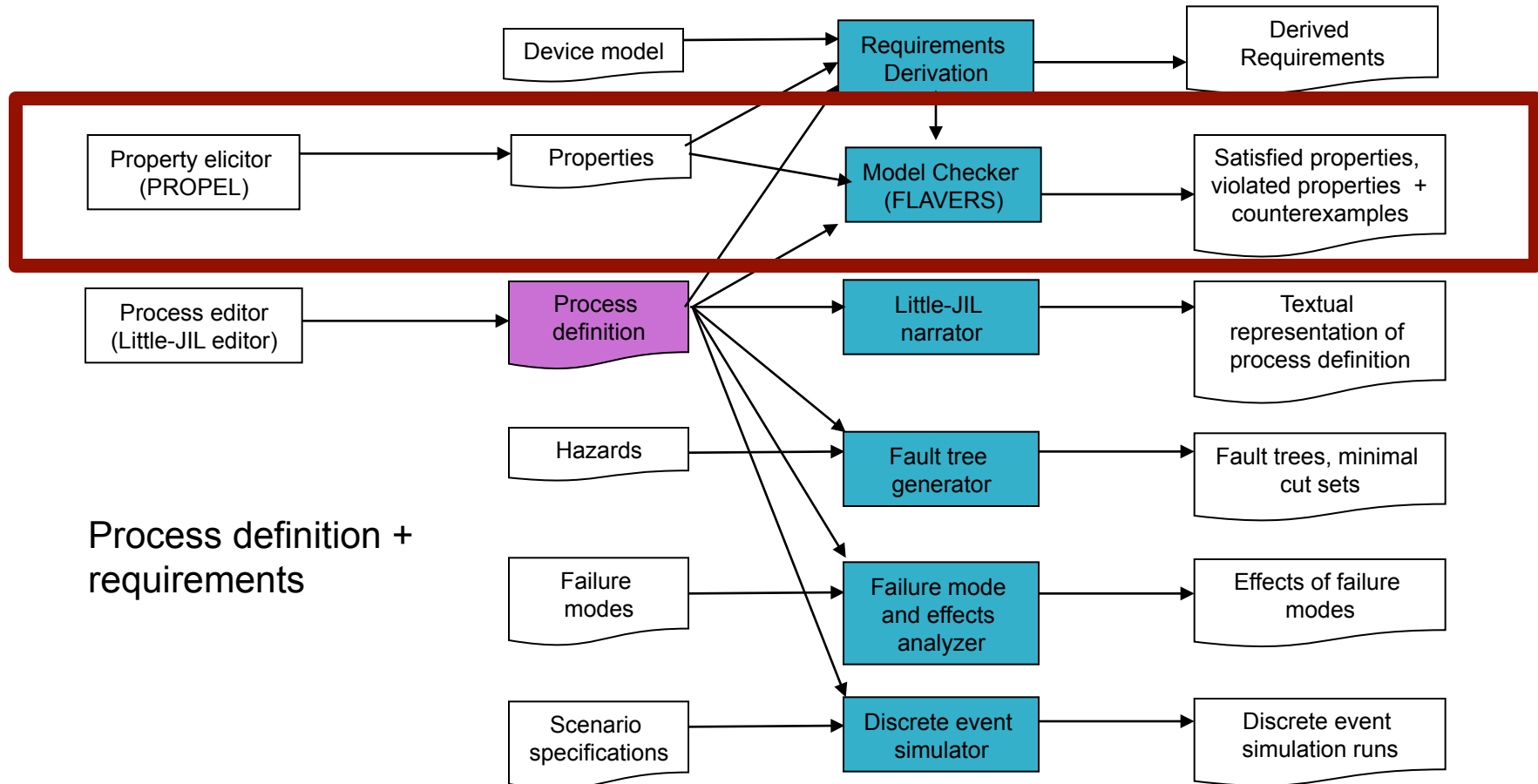


Validating and Verifying the Process Model

- ▶ **How do we know if the model is consistent with the real process?**
 - Review textual/graphical descriptions
 - Examine scenarios (could be automatically generated)
 - Shadow process performers (e.g., with eye trackers)
 - Not the focus of this presentation
- ▶ **Does this process (as captured by the model) allow errors or vulnerabilities to occur?**
 - Want to leverage the investment in creating the model

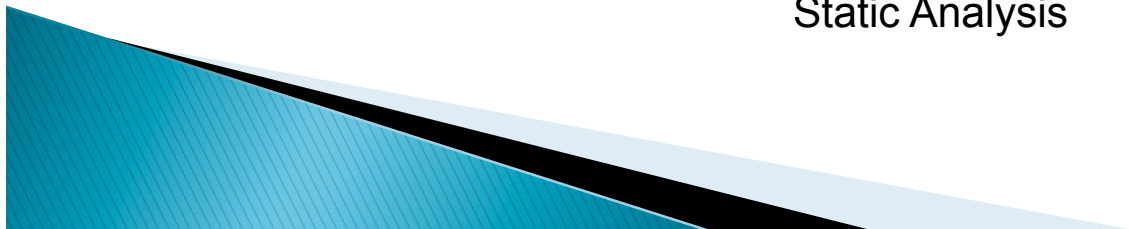


Process Improvement Environment

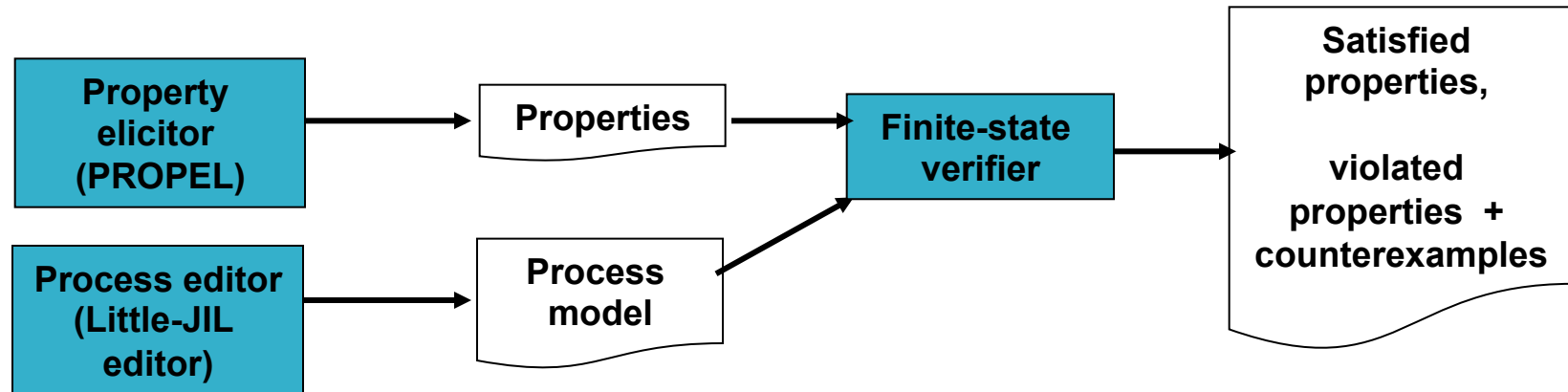


Process definition + requirements

Static Analysis



Model Checking

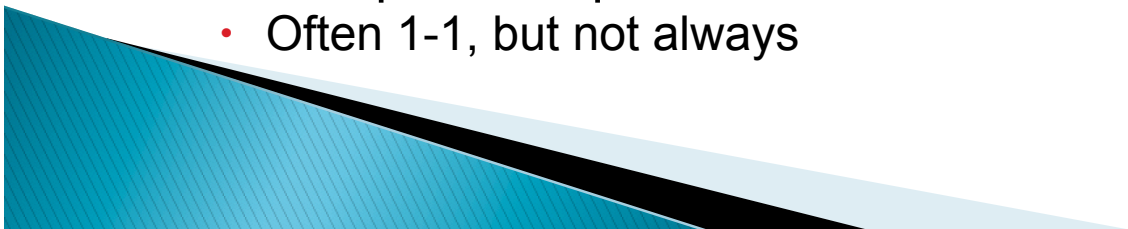


- Are there **any** traces through the process model that will violate a property?
 - e.g., is it possible for a required sequence of events to ever be missed or done out of order?
 - If so, provide counterexample traces



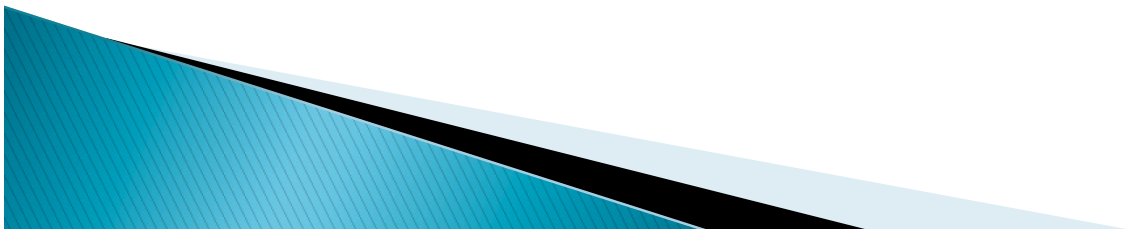
RE Phases

- ▶ Capture and elaborate stated goals
 - Often inconsistent, incomplete, ambiguous,...
 - Helps in understanding the problem, risks, constraints, ...
- ▶ Refine into definitive, natural language statements
 - Develop structural organization of the collection
 - And/Or relationships
 - Generalization, specialization, and refinement
 - Refine glossary
- ▶ Refine into mathematically precise properties that can be used as the basis for verification
 - Bridge between the abstract goals and the process model
 - Create bindings between the events stated in the requirements and the steps in the process model
 - Often 1-1, but not always



PROPEL Templates

- ▶ Provides templates that explicitly indicate the options associated with each Property Pattern (Dwyer, Avrunin, and Corbett)
- ▶ Three coordinated representations
 - Question Tree
 - Helps select the appropriate pattern
 - Guides in the selection of options
 - Disciplined Natural Language (DNL)
 - Specifier selects from given optional phrases
 - Fully instantiated template is a sequence of English sentences
 - Extended Finite-State Automaton
 - Graphical FSA with optional transitions, labels, and accepting states
 - Fully instantiated template is a FSA defining a language of desirable sequences of events; basis for Model Checking



Question Tree View

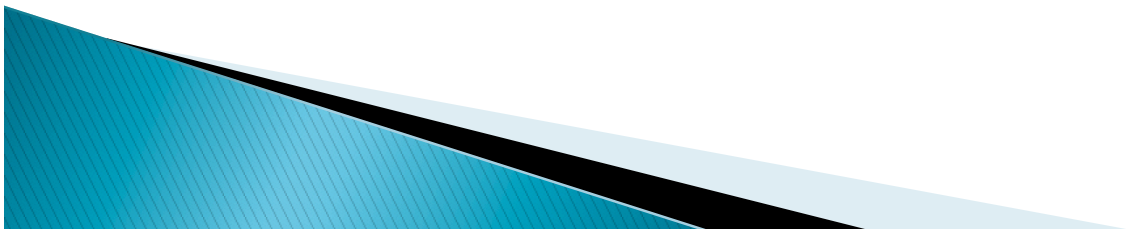
How many events of primary interest are there?

- One: event verify-patient-ID

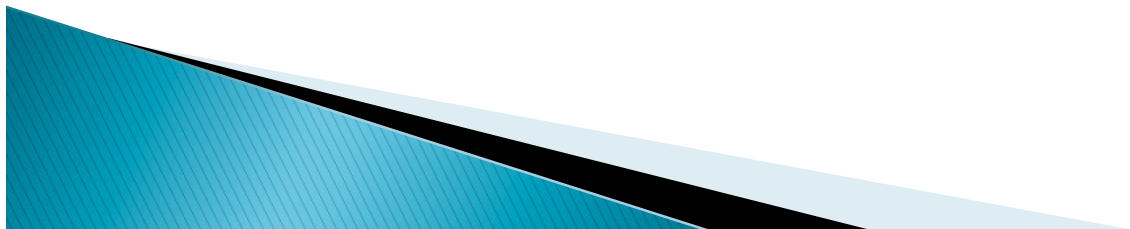
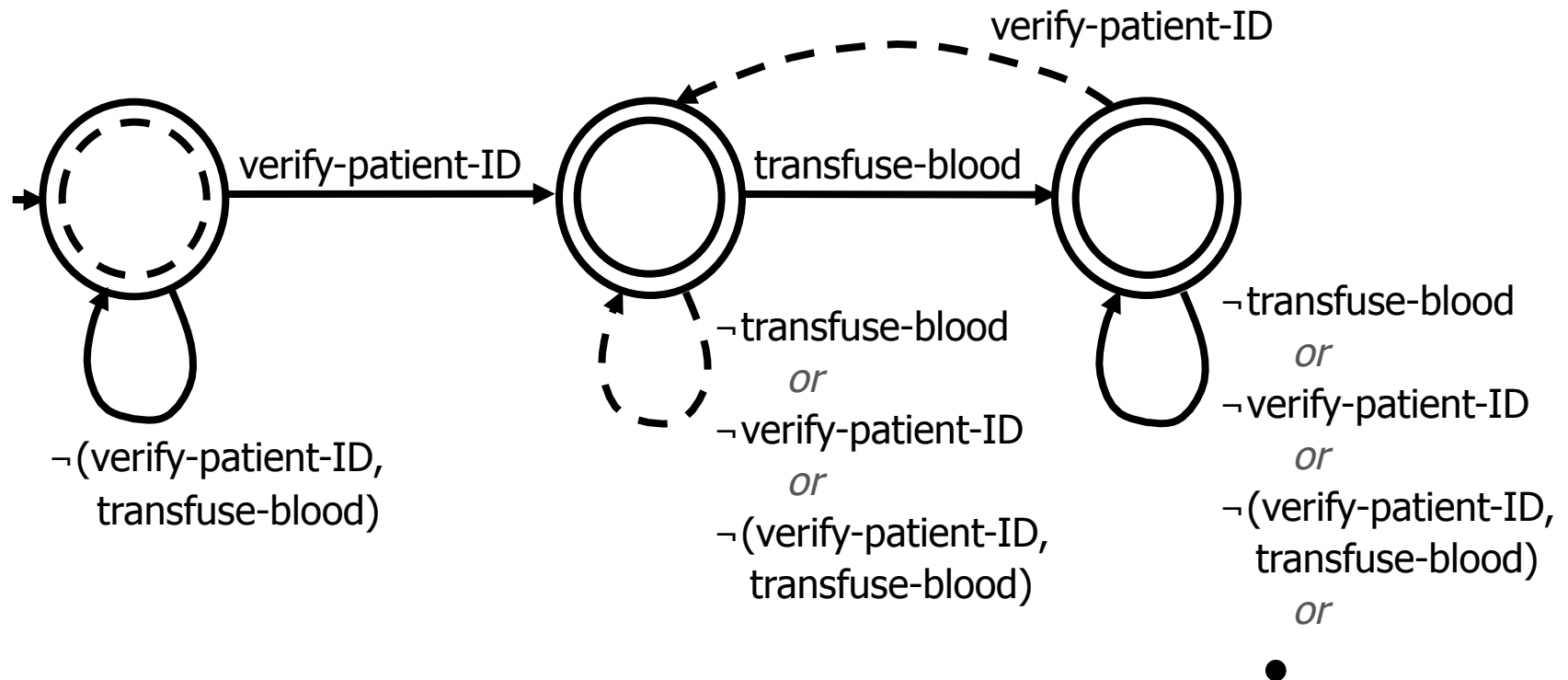
- Two: events verify-patient-ID and transfuse-blood

- After verify-patient-ID occurs, transfuse-blood is required to occur

- transfuse-blood cannot occur until after verify-patient-ID has occurred



Precedence FSA Template



Precedence DNL Template

▼ Behavior & Scope Disciplined English View

transfuse-blood cannot occur unless **verify-patient-ID** has already occurred.

▼ **transfuse-blood** is not required to occur.

Before the first **verify-patient-ID** occurs, the events in the alphabet of this property, other than **transfuse-blood** , can occur any number of times.

After **verify-patient-ID** occurs and before the first subsequent **transfuse-blood** occurs:

▼

After the first subsequent **transfuse-blood** occurs:

▼

Precedence DNL Template

▼ Behavior & Scope Disciplined English View

transfuse-blood cannot occur unless **verify-patient-ID** has already occurred.

▼ **transfuse-blood** is not required to

verify-patient-ID is required to occur, but

verify-patient-ID is not required to occur, however,

It is acceptable if **verify-patient-ID** does not occur, however,

transfuse-blood, can occur any number of times. abet of this property, other than

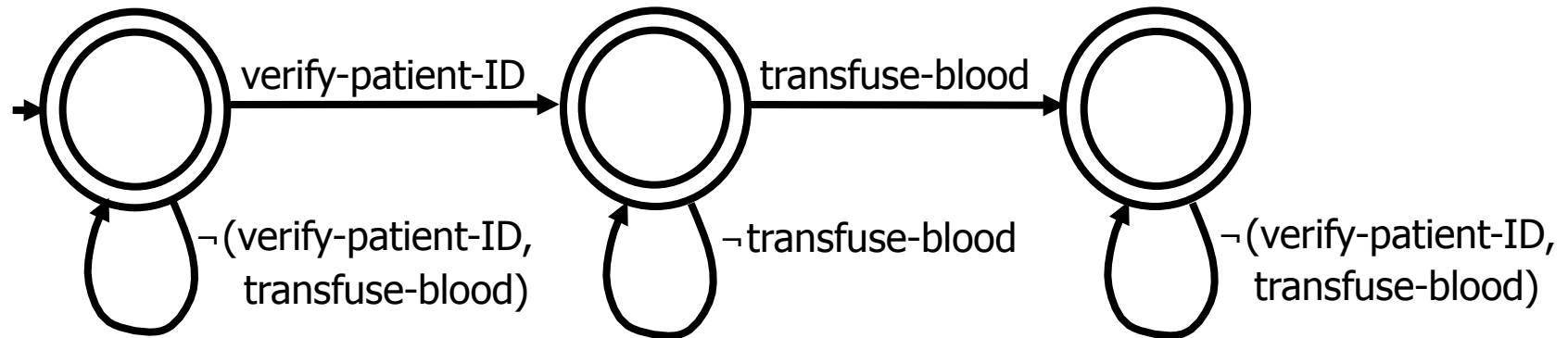
After **verify-patient-ID** occurs and before the first subsequent **transfuse-blood** occurs:

▼

After the first subsequent **transfuse-blood** occurs:

▼

Example Behavior



transfuse-blood cannot occur unless verify-patient-ID has already occurred.

It is acceptable for verify-patient-ID to not occur, but if it does not occur then transfuse-blood can not occur. Even if verify-patient-ID does occur, transfuse-blood is not required to occur.

Before the first verify-patient-ID occurs, the events in this property, other than transfuse-blood, can occur any number of times.

After verify-patient-ID occurs and before the first subsequent transfuse-blood occurs:

- the events in this property, including verify-patient-ID but not transfuse-blood, can occur any number of times.

After the first subsequent transfuse-blood occurs:

- the events in this property, other than verify-patient-ID or transfuse-blood, could occur any number of times;
- neither verify-patient-ID nor transfuse-blood can occur again.

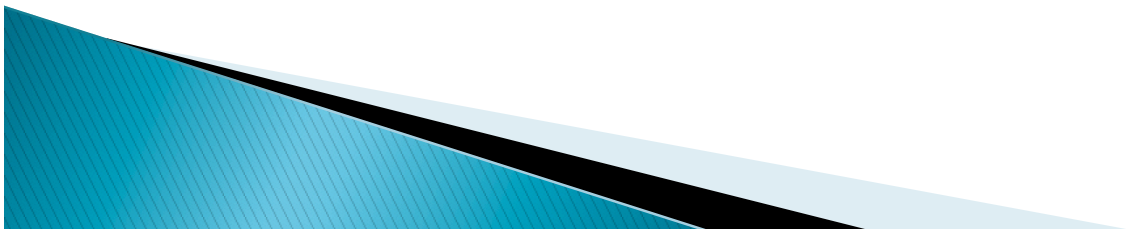
Observations about Specifying Properties

- ▶ Specifying the properties helped determine the scope/granularity of the process
- ▶ Need to specify properties in the context of exceptions
 - PropA is true unless exception X1 or X2 occurs
- ▶ Difficult for clinicians to understand the difference between a process model, a property, and a scenario
 - Scenario describes a particular situation
 - Process prescribes what to do for all situations
 - Property describes what should be true no matter how the process is defined or implemented



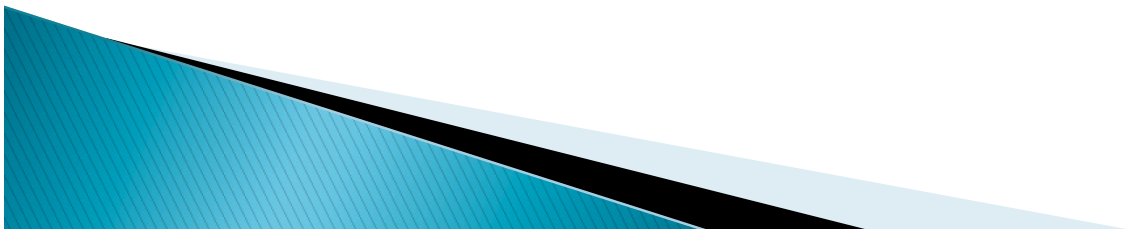
Observations about Verifying the Models

- ▶ Just doing the modeling helped uncover errors in the processes
- ▶ Initially we mostly found errors in the process models and the properties
- ▶ After fixing the modeling errors, we found errors in the real processes
 - E.g.,
 - Deadlock
 - Stale height and weight



Observations about Verifying the Models

- ▶ Our process models were amenable to model checking
 - Few infeasible paths and most of our properties were event based
 - => few false positives
 - Scalability still a concern
- ▶ Fixing the errors often led to other errors
- ▶ *If processes are complex enough to be modeled, the models must be carefully validated!*



Faults versus Vulnerabilities

- ▶ **Model checking** assumes that the stated tasks are done correctly, but tries to determine if the tasks are always done in the right order with the right values
- ▶ **Safety analysis** tries to determine what harm might be done if the tasks are not done correctly
 - **Failure mode and effects analysis**
 - What hazards might arise, if there is a failure in the system?
 - **Fault tree analysis**
 - What are the ways in which a particular hazard might occur
 - E.g., single points of failure

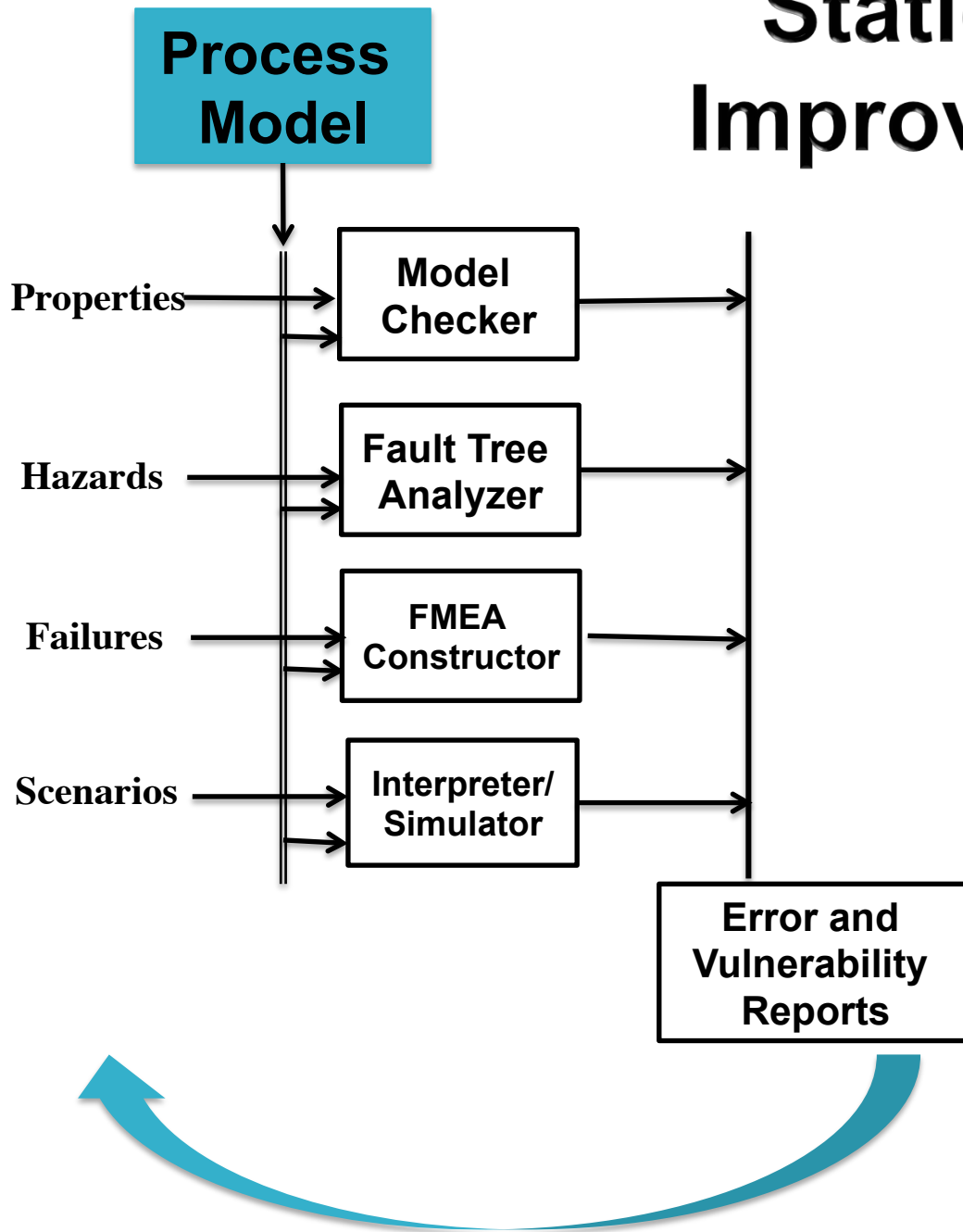


Overall Observations

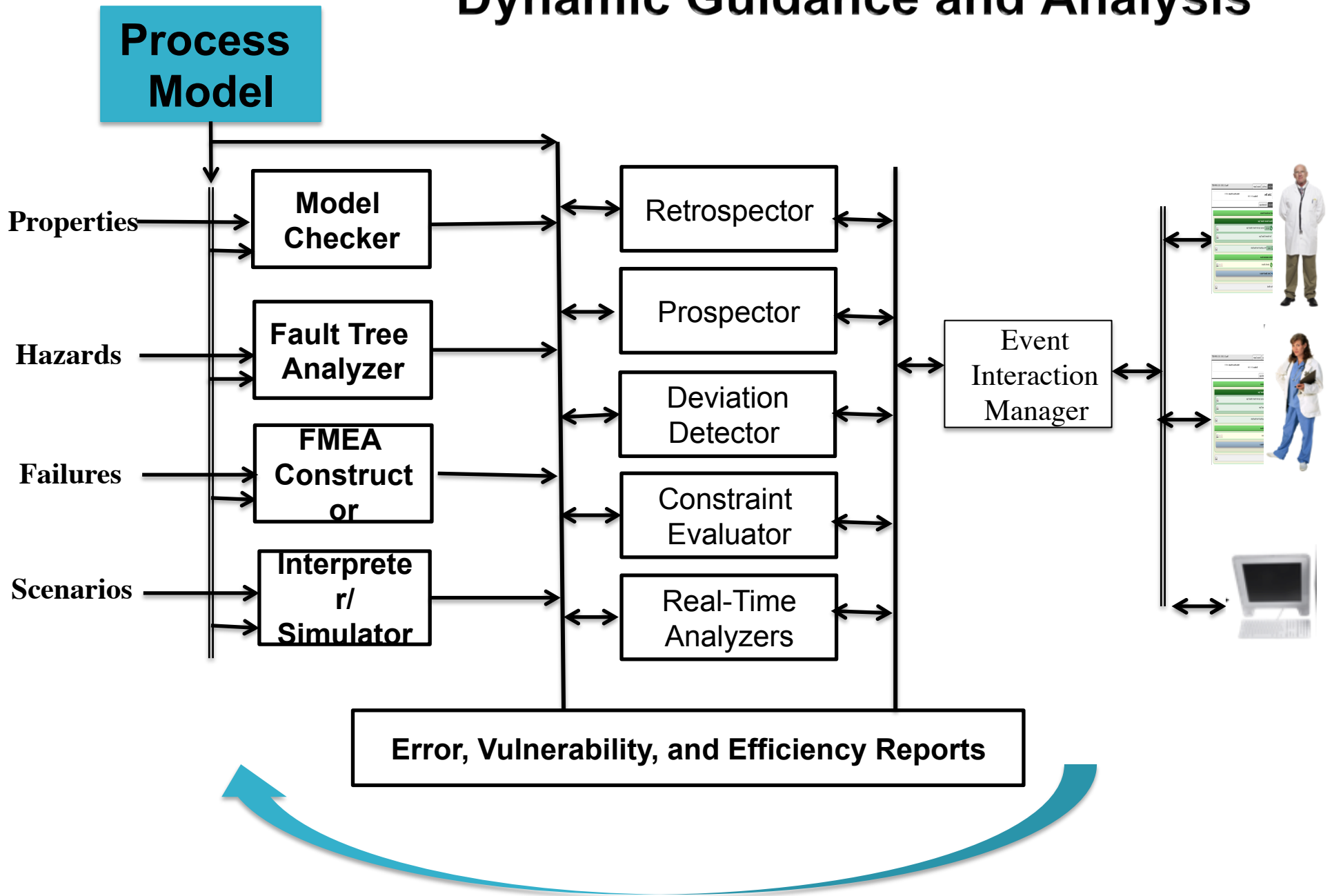
- ▶ Found several important errors and inefficiencies in the processes that we modeled
 - Sequence errors, deadlocks, single points of failure
- ▶ Testimonials
 - Medical colleagues claim that this approach has changed the way they view their processes, the terms they use, and how they teach their disciplines
- ▶ **Chemotherapy process saw a 70% reduction in errors that reach the patient**



Static Analysis Improvement Loop



Dynamic Guidance and Analysis



Concisely Communicate Process State: Smart Checklists

- ▶ **Retrospection, Prospection, and Current Context** based on accurate monitoring of process steps
- ▶ Exploit process model info, such as subprocesses and iteration
- ▶ Provide alternative views that highlight what is important to an agent
- ▶ Support queries about the past
- ▶ Use simulations to predict future alternatives

The screenshot shows a 'Smart Checklist' application window. At the top, it displays patient information for John Doe: Gender: MALE, Birthdate: 1993-03-03, Age: 20, and MRN: 12345678. Below this, the main process is titled 'PROCESS "set up infusion pump"'. The checklist is organized into several steps, each with a status indicator (checkmark or calendar icon) and a timestamp:

- retrieve pump** (12:09) - Status: Not completed
- assess infusion-related hazards and configure pump** (12:10) - Status: Completed
- assess infusion-related hazards** (12:10) - Status: Completed
- configure pump when there are infusion-related hazards** (12:11) - Status: Completed
- perform baseline assessment test** (12:11) - Status: Completed
- document and recommend frequent monitoring** (12:11) - Status: Completed
- configure pump** (12:11) - Status: Completed
- load drug container into pump** (12:11) - Status: Completed

A tooltip for the 'load drug container into pump' step lists the required inputs: drug container and tubing.

Conclusions

- ▶ Approach not restricted to medical processes
 - Systematic, analysis-based approach to process improvement
 - Effective for finding errors and vulnerabilities in complex, *human-intensive processes*
- ▶ Future work: monitoring and guidance based on validated process models
 - Basis for real-time deviation detection
 - Framework for accumulating operational data, applying probabilistic analysis, and proposing **evidence-based** process improvements

-

