

Analysis Support for Continuous Process Improvement

Lori A. Clarke

Department of Computer Science
University of Massachusetts, Amherst
clarke@cs.umass.edu

In collaboration with
Leon J. Osterweil and George S. Avrunin

Our Vision

- **Model processes**
- **Evaluate them**
 - **Using a wide variety of analysis techniques**
- **Propose modifications**
- **Deploy them: Process-guided support**
- **Reevaluate in the deployed setting**
- **Propose modifications**

Process Analysis Technologies

- View generation to improve understandability
- Requirements engineering to capture properties
 - PROPEL (PROPErty ELucidation)
 - Can specify sequences of events that should and should not happen
- Finite-state verification to detect errors
 - FLAVERS (FLow Analysis for VERifying Systems) and SPIN
 - Determines if the properties are always true for the process model and, if not, show counter example traces
- Hazard analysis to reveal vulnerabilities
 - Fault tree and Failure Mode Effects Analysis (FTA/FMEA)
 - Evaluates if tasks are not done correctly, what damage could result
- Simulation to evaluate efficiency

Process Modeling Language Requirements

- **Capture complexity of processes**
 - Support concurrency, synchronization, exceptions
 - Support abstraction and decomposition
- **Flexible enough to support human creativity**
- **Precise enough to support static analysis and to drive simulations and executions**
- **Understandable to non-computer scientists**

Many views of the same process

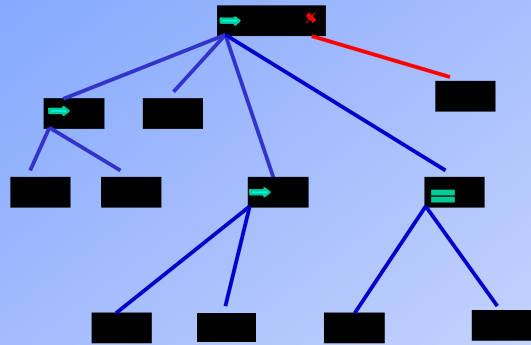
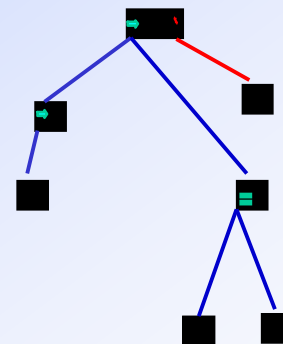
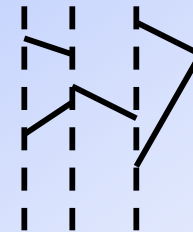


Table Of Contents	Legend	Index of step names
<ul style="list-style-type: none">perform chemotherapy processprepare for and administer chemotherapyperform consultation and assessment<ul style="list-style-type: none">fill out medical history formsmeasure and record height and weightconfirm pathology report indicates cancerperform patient consultationcreate treatment plan and orderscreate treatment	<ul style="list-style-type: none">Before beginning to "perform chemotherapy process", the step "refer patient to an oncologist" must be completed successfully.To "perform chemotherapy process", the following need to be done in any order (including simultaneously), <u>prepare for and administer chemotherapy</u> and <u>create and process consult note</u>.To "prepare for and administer chemotherapy", the following need to be done in the listed order<ul style="list-style-type: none">perform consultation and assessmentperform initial review of patient recordsperform pharmacy tasks	<h3>Perform Chemotherapy Process</h3> <h3>Prepare For And Administer Chemotherapy</h3>



Textual and Hyperlinked View

Table Of Contents

- = [perform chemotherapy process](#)
 - [prepare for and administer chemotherapy](#)
 - [perform consultation and assessment](#)
 - [fill out medical history forms](#)
 - [measure and record height and weight](#)
 - [confirm pathology report indicates cancer](#)
 - [perform patient consultation](#)
 - = [create treatment plan and orders](#)
 - [create treatment](#)

[Legend](#) [Index of step names](#)

Perform Chemotherapy Process

▼ Before beginning to "perform chemotherapy process", the step "refer patient to an oncologist" must be completed successfully.

= To "perform chemotherapy process", the following need to be done in any order (including simultaneously), [prepare for and administer chemotherapy](#) and [create and process consult note](#).

Prepare For And Administer Chemotherapy

→ To "prepare for and administer chemotherapy", the following need to be done in the listed order

- [perform consultation and assessment](#)
- [perform initial review of patient records](#)
- [perform pharmacy tasks](#)

Process Modeling Observations

- **Even “simple” processes can be very complex**
 - E.g., verify patient ID
- **Modeling processes results in the discovery of errors in the process**
 - Artifacts created and not used
 - Inconsistent units
- **Multiple views of the process definition help detect errors in the process/process model**

Process Modeling Observations

- There is no best process modeling representation
 - Large graphical representations are hard to comprehend
 - Large textual representations are hard for following complex flow
- Want a representation that will
 - Be the basis for answering questions about the model

Validating Process Definitions

- How do we know if a process definition correctly describes the process?
 - Reviews
 - textual/graphical descriptions
 - Review scenarios or role-based views (could be automatically generated)
 - Shadowing with think aloud or eye-tracking support
- How do we know if the process, as defined, satisfies the requirements?
 - Show that all traces through the process definition satisfy important properties
 - But, need to represent these properties

Property Specification (Requirements Engineering)

- **State Initial Goals**
 - Often inconsistent, incomplete, ambiguous,...
- **Refine into definitive, natural language statements and informal models**
 - Develop structural organization of the collection
 - Define glossary
- **Refine into mathematically precise properties that can be used as the basis for validation**
 - Drive testing and verification
 - Bridge between the **abstract goals** and the **process model**

Getting the Details Right

- For verification, need a mathematically precise specification
 - Temporal logics difficult to use or understand
 - Finite-state automata easier to comprehend, but still difficult to capture the details of the actual intent
- PROPEL: Property Elucidation
 - Extends the Property Pattern work of Dwyer, Avrunin, and Corbett
 - Common property patterns
 - Provides more detailed templates that explicitly indicate the options associated with each pattern
 - Provides alternative views of these templates

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 FSV

Example Property

The patient's identification must be verified prior to transfusing each unit of blood product.

EVENT-A: verify-patient-ID

EVENT-B: transfuse-blood

Question Tree View

▼ Behavior Question Tree View

- How many events of primary interest are there?
 - One: event `verify-patient-ID`
 - Two: events `verify-patient-ID` and `transfuse-blood`
 - How do `verify-patient-ID` and `transfuse-blood` interact?
 - `verify-patient-ID` causes `transfuse-blood` to occur
 - `transfuse-blood` cannot occur until after `verify-patient-ID` has occurred
 - Is `verify-patient-ID` required to occur at least once?
 - Yes, `verify-patient-ID` is required to occur at least once
 - No, `verify-patient-ID` is not required to occur at least once
 - After `verify-patient-ID` occurs, can `verify-patient-ID` occur again before the first subsequent `transfuse-blood` occurs?
 - Yes, `verify-patient-ID` can occur multiple times before the first subsequent `transfuse-blood` occurs
 - No, `verify-patient-ID` cannot occur again before the first subsequent `transfuse-blood` occurs
 - After `verify-patient-ID` occurs, can events in the alphabet of this property other than

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, label of this property, other than

transfuse-blood, can occur any number of times.

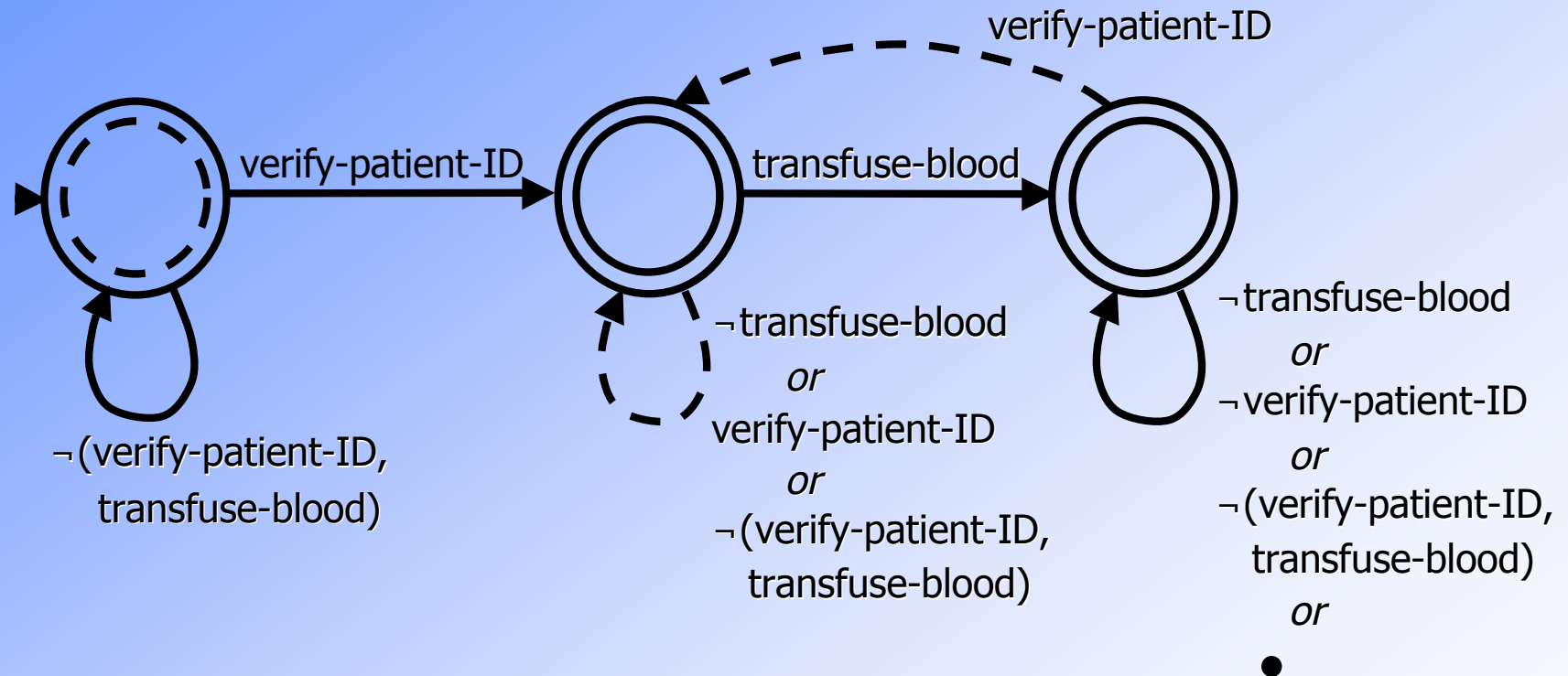
After **verify-patient-ID** occurs and before the first subsequent **transfuse-blood** occurs:

▼

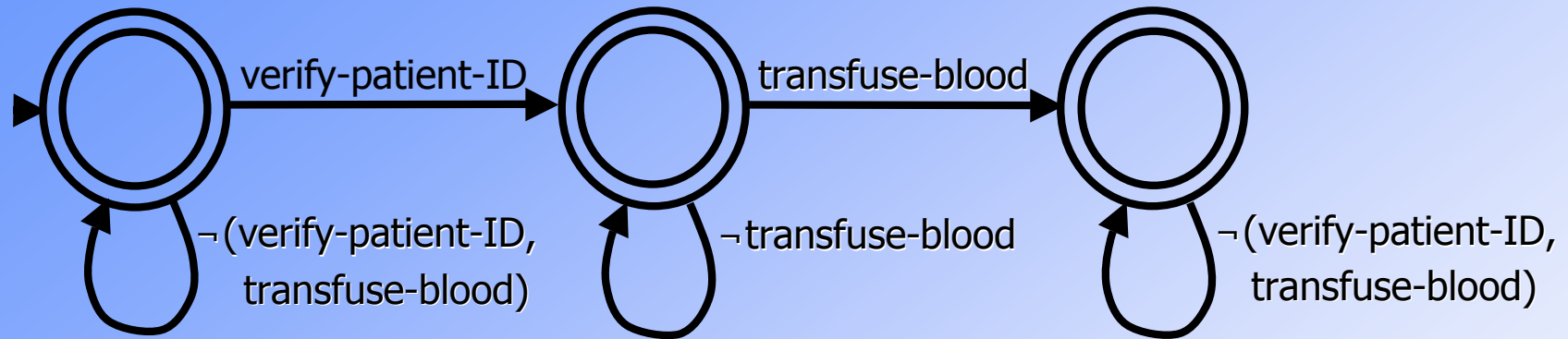
After the first subsequent **transfuse-blood** occurs:

▼

Precedence FSA Template



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 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:

- the events in the alphabet of 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 the alphabet of 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
- Just specifying the properties helped find errors in the process definitions
 - Errors of omissions

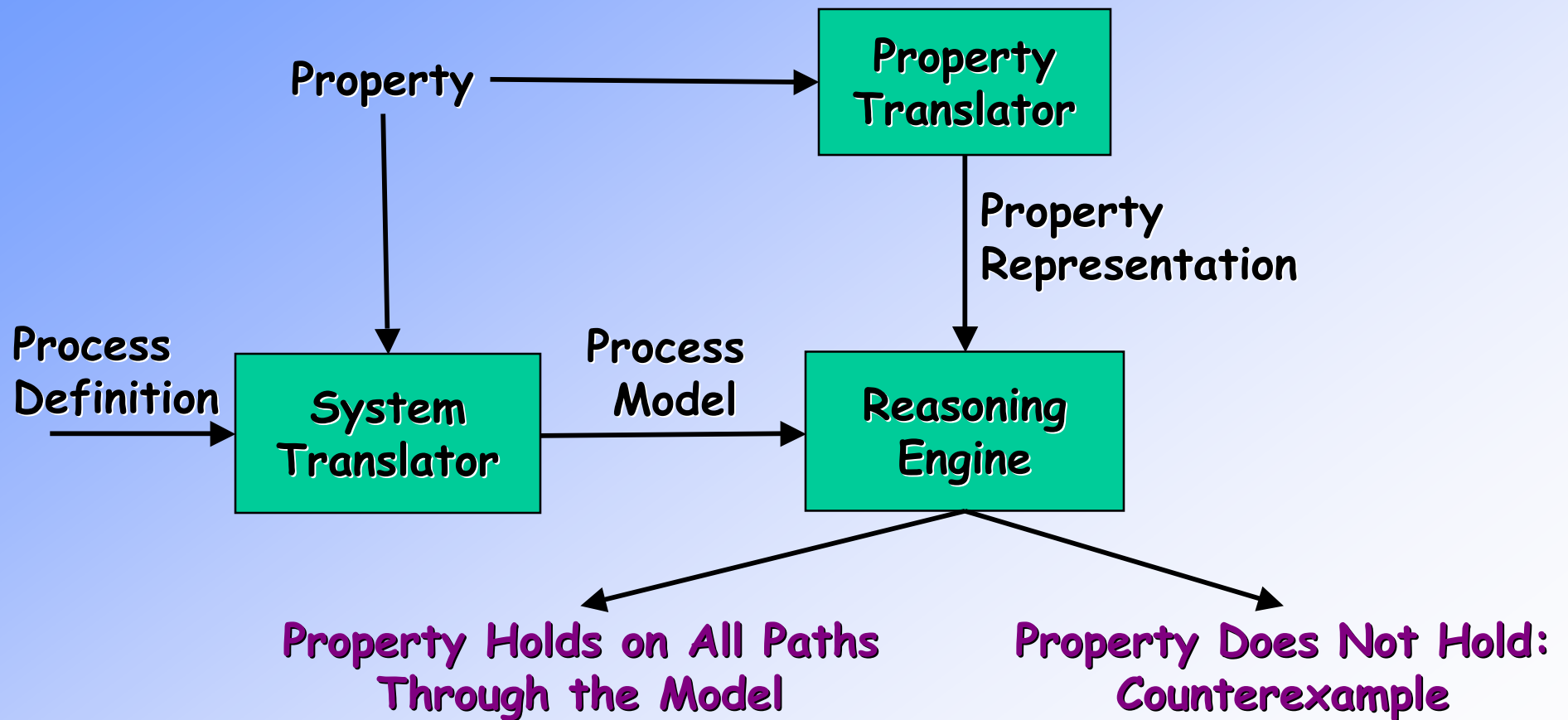
Observations about Specifying Properties

- Difficult to specify properties in the context of exceptions
 - PropA is true unless exception X1 or X2 occurs
- Collection of requirements a valuable asset for revalidation
 - Continually grown and improved
 - Continually reused

Finite-State Verification of Process Definitions

- **Determines if the process definition is consistent with a property**
 - Considers every trace through the process
- **If a property does not hold, the verification tool will provide a counterexample trace**
 - Process improvement: change process, property, or both
 - Re-verify until satisfied

Finite-State Verification



Observations about FSV of processes

- **FLAVERS a nice fit for Little-JIL**
 - Event based
 - Specialized optimizations for Little-JIL
- **There is a tension between expressiveness and analyzability**
 - Some of the more expressive constructs in Little-JIL are difficult to model e.g., choice step
- **FSV reveals errors**
 - Process definition errors and process errors
- **Interesting processes are so complex that verification techniques must be applied**

Process Analysis Technologies

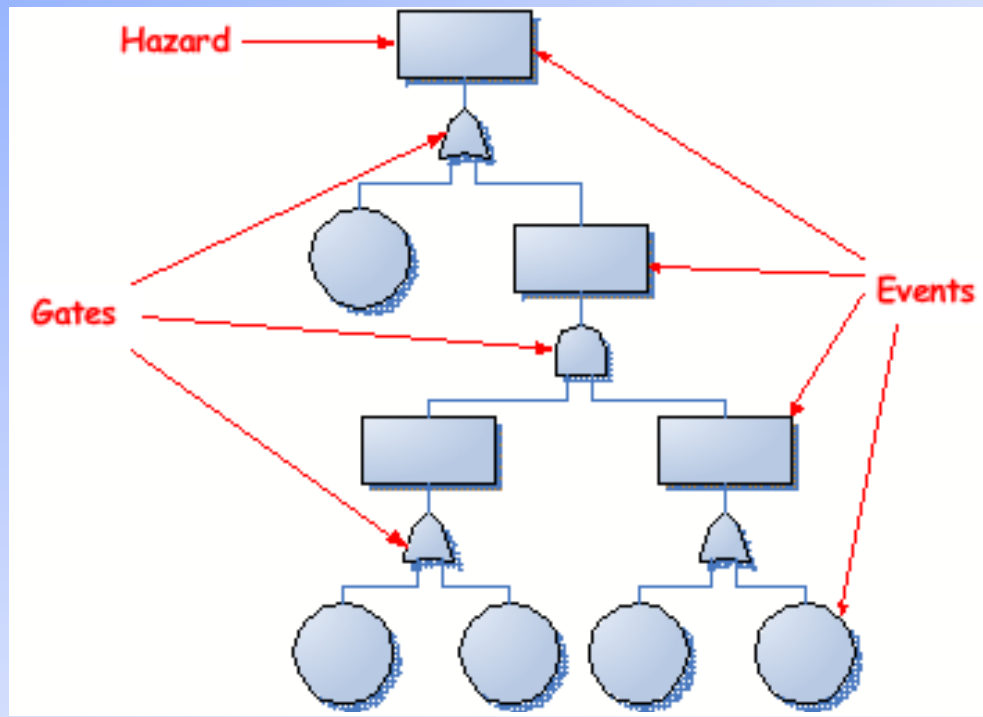
- Requirements engineering to capture properties
 - PROPEL (property elucidation system)
- Finite-state verification to detect errors
 - FLAVERS (Flow Analysis for Verifying Systems) and SPIN
- Hazard analysis to reveal vulnerabilities
 - Fault-tree Analysis and Failure Mode and Effect Analysis to reveal vulnerabilities
- Simulation to improve efficiency

Fault Tree Analysis (FTA)

- A well accepted and widely practiced hazard analysis technique
 - Hazard - condition that can result in a significant loss
- Systematically identify and reason about all possible events that could lead to a given hazard
 - Create fault tree for a hazard
 - Analyze each fault tree
- Analysis results can be used to improve the process

Fault Tree

A fault tree captures all the combinations of events that could lead to a given hazard



Problem

- Difficult to manually develop correct fault trees for large processes
 - Requires deep understanding of processes
 - Time-consuming
 - Error-prone
 - Errors in fault trees affect the validity of decisions made to improve processes

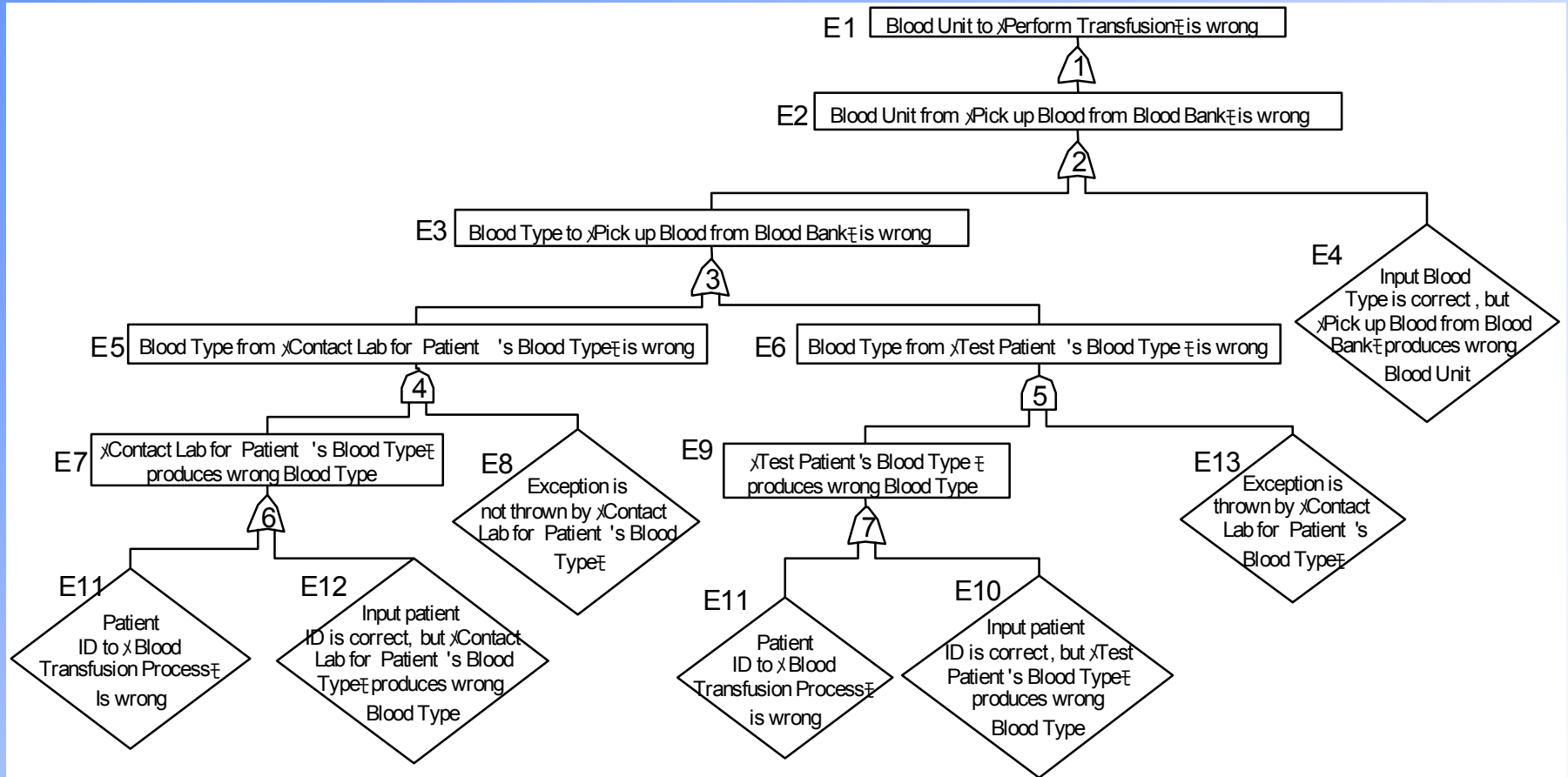
Our Approach

- **Automatically develop fault trees from process definitions**
 - requires the process be formally defined
 - 17 templates are defined based on Little-JIL semantics
 - defines the cause-consequence relationships between events and intermediate event
- **Identify process vulnerabilities using FTA**

Analyze Fault Trees - Calculate Minimal Cut Sets (MCSs)

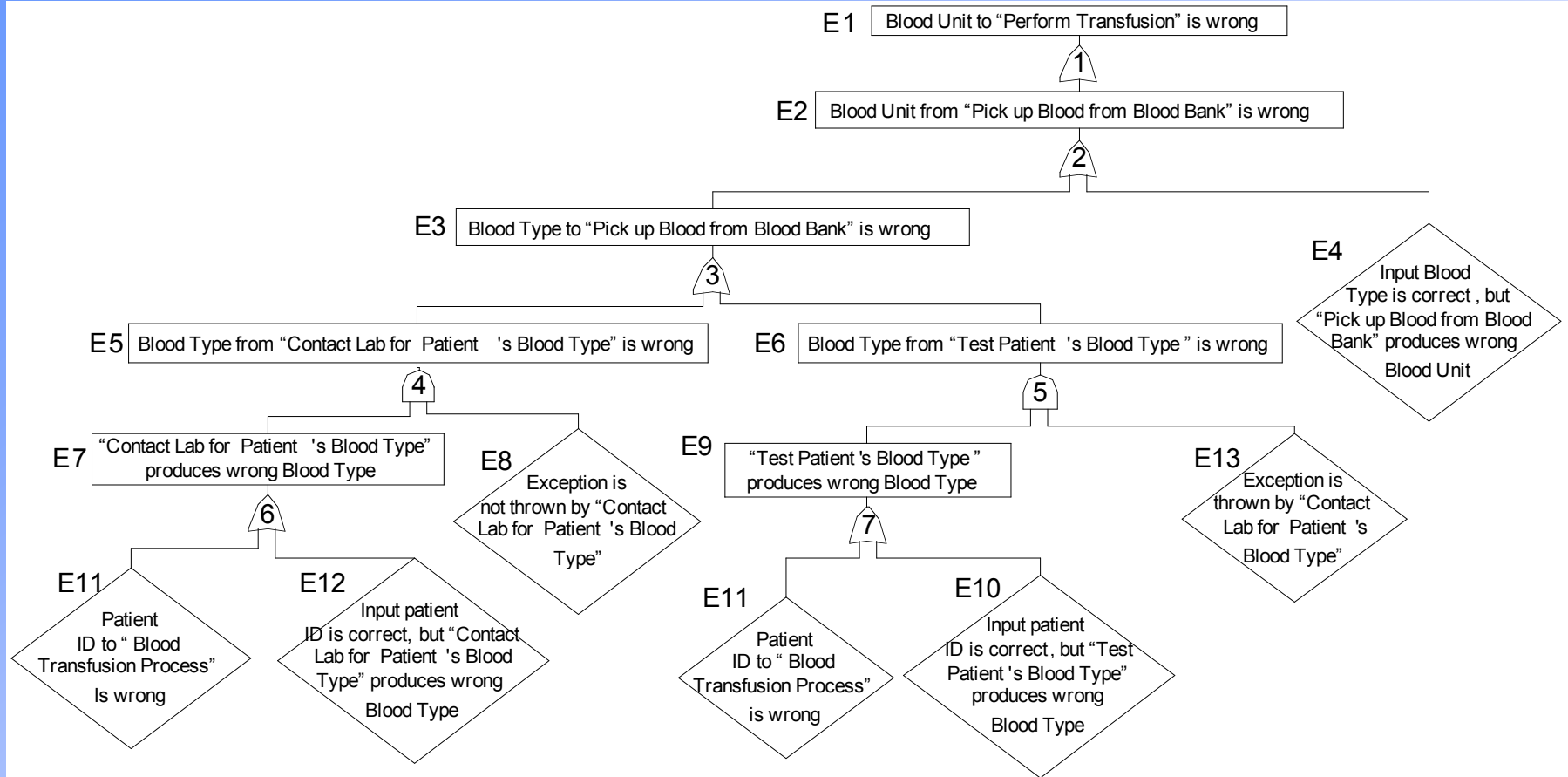
- Cut set - a set of basic events and/or undeveloped events whose occurrence ensures that the TOP event occurs
- MCS - a cut set that cannot be further reduced
- MCSs can be automatically calculated from a fault tree using Boolean algebra

Calculate MCSs



Each gate corresponds to an equation

Calculate MCSs

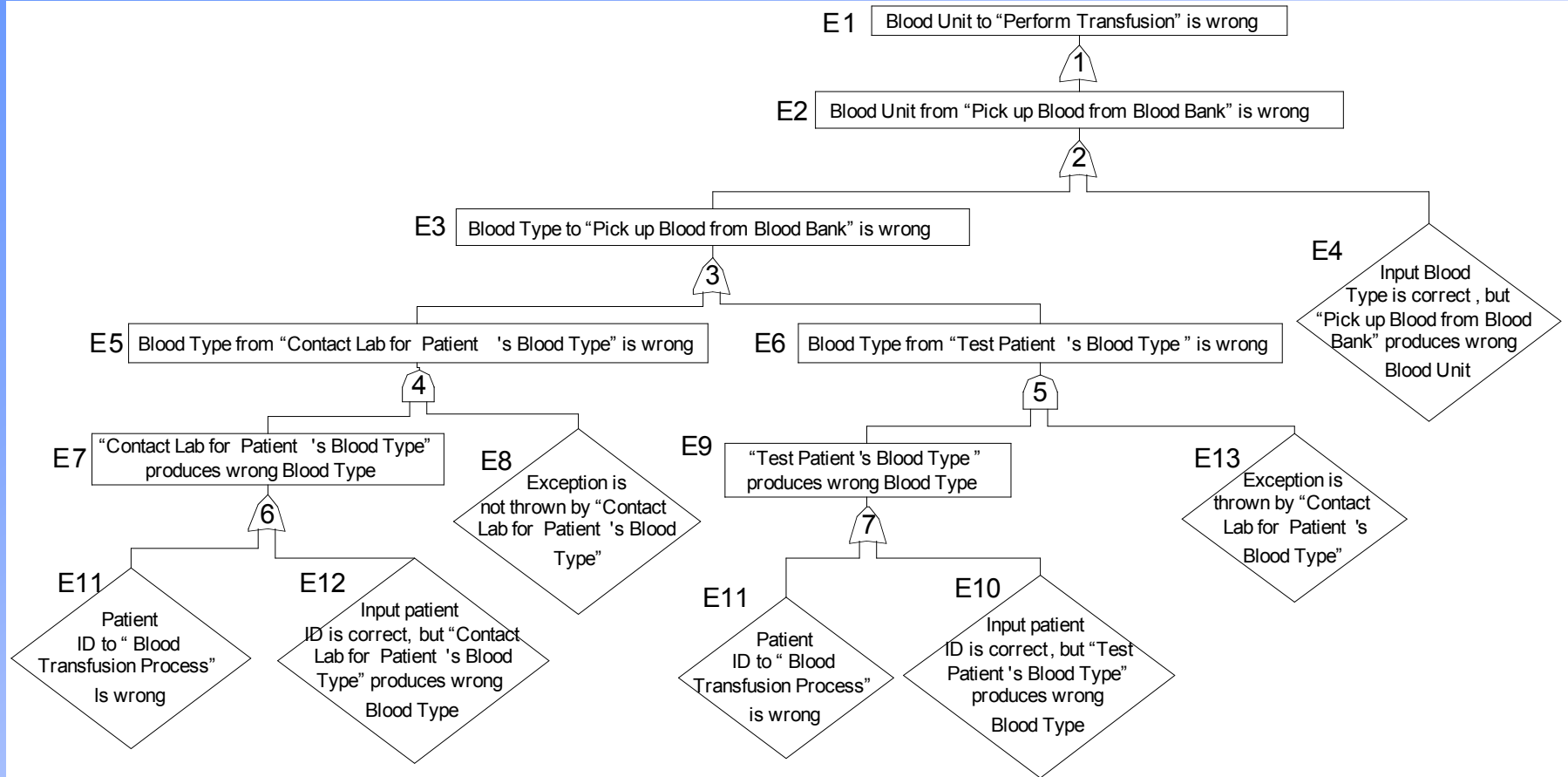


Each gate corresponds to an equation

$$1: E1 = E2 \quad 2: E2 = E3 + E4 \quad 3: E3 = E5 + E6 \quad 4: E5 = E7 \cdot E8$$

$$5: E6 = E9 \cdot E13 \quad 6: E7 = E11 + E12 \quad 7: E9 = E11 + E10$$

Calculate MCSs



Derive an equation for E1 by eliminating and substituting the other intermediate events:

$$E1 = (E4) + (E11) + (E12 \cdot E8) + (E10 \cdot E13)$$

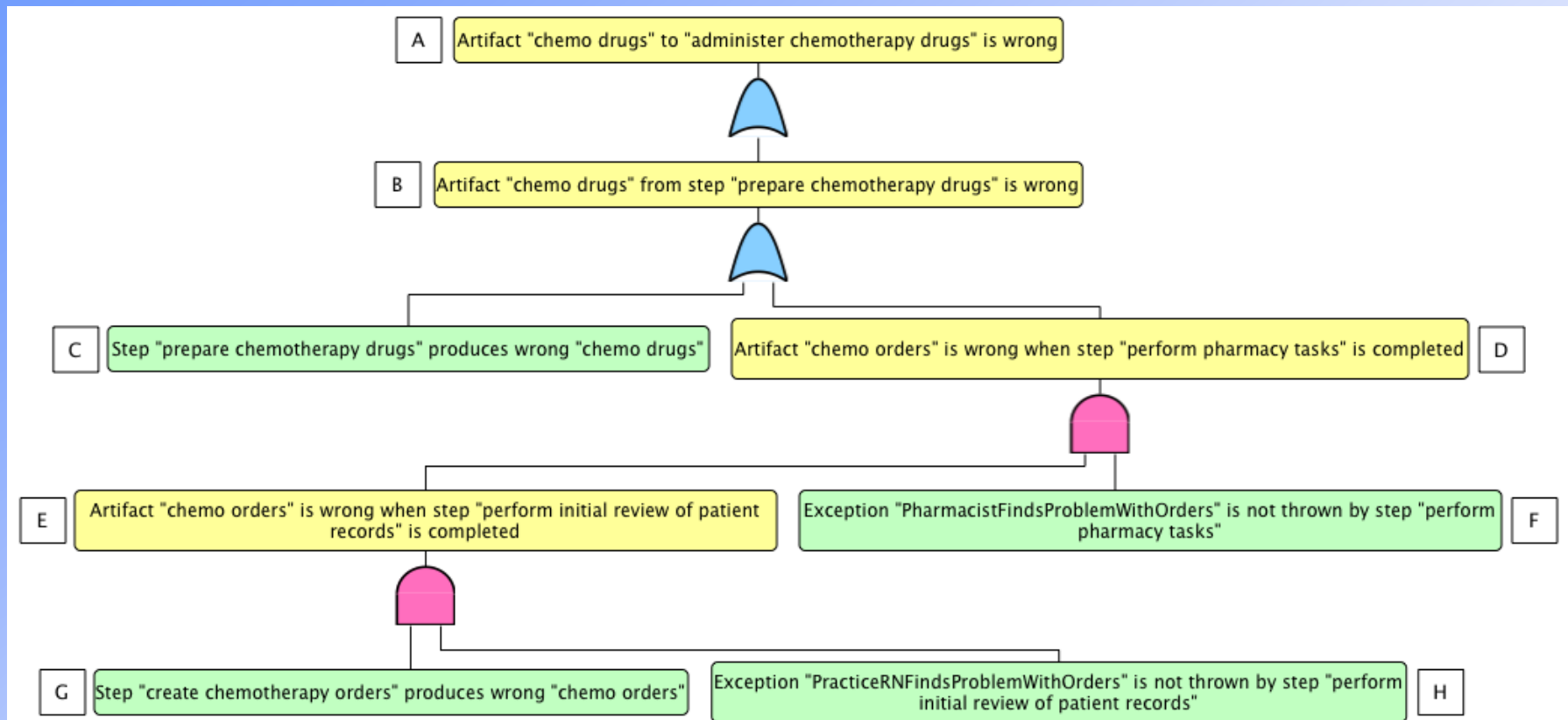
Using MCS to Drive Process Improvements

- To control or eliminate a hazard, several options could be applied
 - A more failure-resistant agent could be assigned to some steps where major faults could occur
 - Consistency check steps could be added to well-chosen places in the process to stop the propagation of faults
- The effectiveness of an option can be decided by the reduction in the probability of the hazard, if the probabilities of primary events are known

Observations about Fault Tree Analysis

- The completeness and correctness of a derived fault tree depends on the completeness and correctness of the process definition
 - Easier to construct a process definition than a fault tree
 - **Many** fault trees can be derived from a single process definition

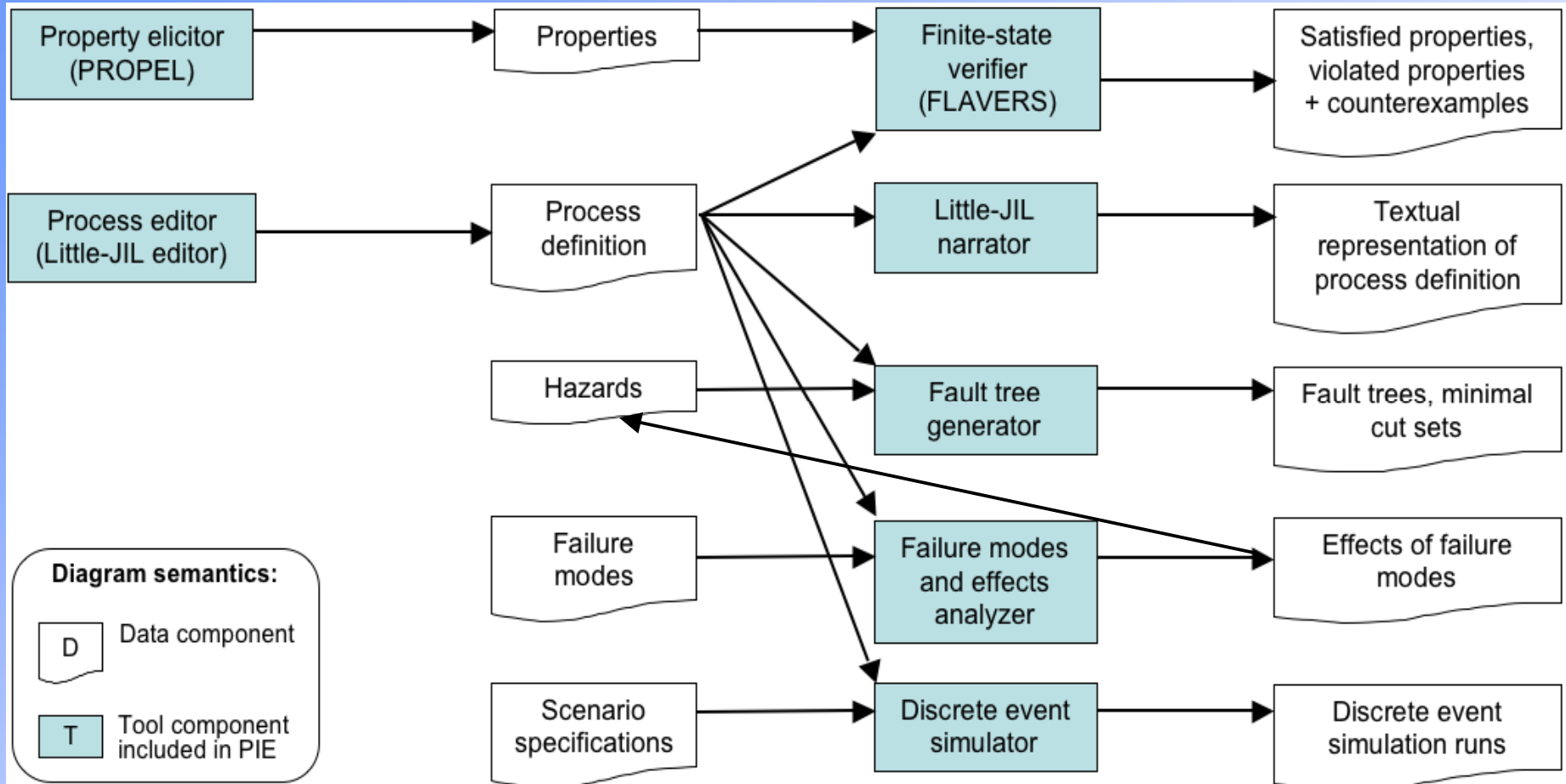
A Fault Tree for the Chemo Process



Complementary Analysis Techniques

- **Finite State Verification** assumes tasks are done correctly, but detects when the order of events can lead to problems (as indicated in a property specification)
- **Fault Tree Analysis** assumes that the tasks/artifacts might be wrong and shows where the process is vulnerable if such problems arise
 - **Failure Mode and Effect Analysis** shows the hazards that might result from failures

Little-JIL Process Improvement Environment



Future Work

- **Process-guided execution**
 - Recognize relevant and non-relevant events
 - Recognize the start and end of a task
 - Resynchronize after process deviations
 - Guidance with support for overrides
 - Gather longitudinal data
 - Probabilities of different tasks

Conclusions

- **Approach is not domain specific**
 - Dealing with complex, human-intensive systems
 - Humans provide creative insight
 - Complex human and device/computer interactions
 - Other process definition languages and tools might also be effective
- **Have found important errors**
 - Single points of failure
 - Inconsistent interfaces
 - Deadlock

Conclusion

- **Process Models and Properties need to be validated**
 - Both are very error prone
 - If a process is worth modeling, then the model requires significant validation