Engineering Medical Processes to Improve Their Safety

Leon J. Osterweil, George S. Avrunin, Bin Chen, Lori A. Clarke, Rachel Cobleigh

Lab. for Advanced Software Engineering Research

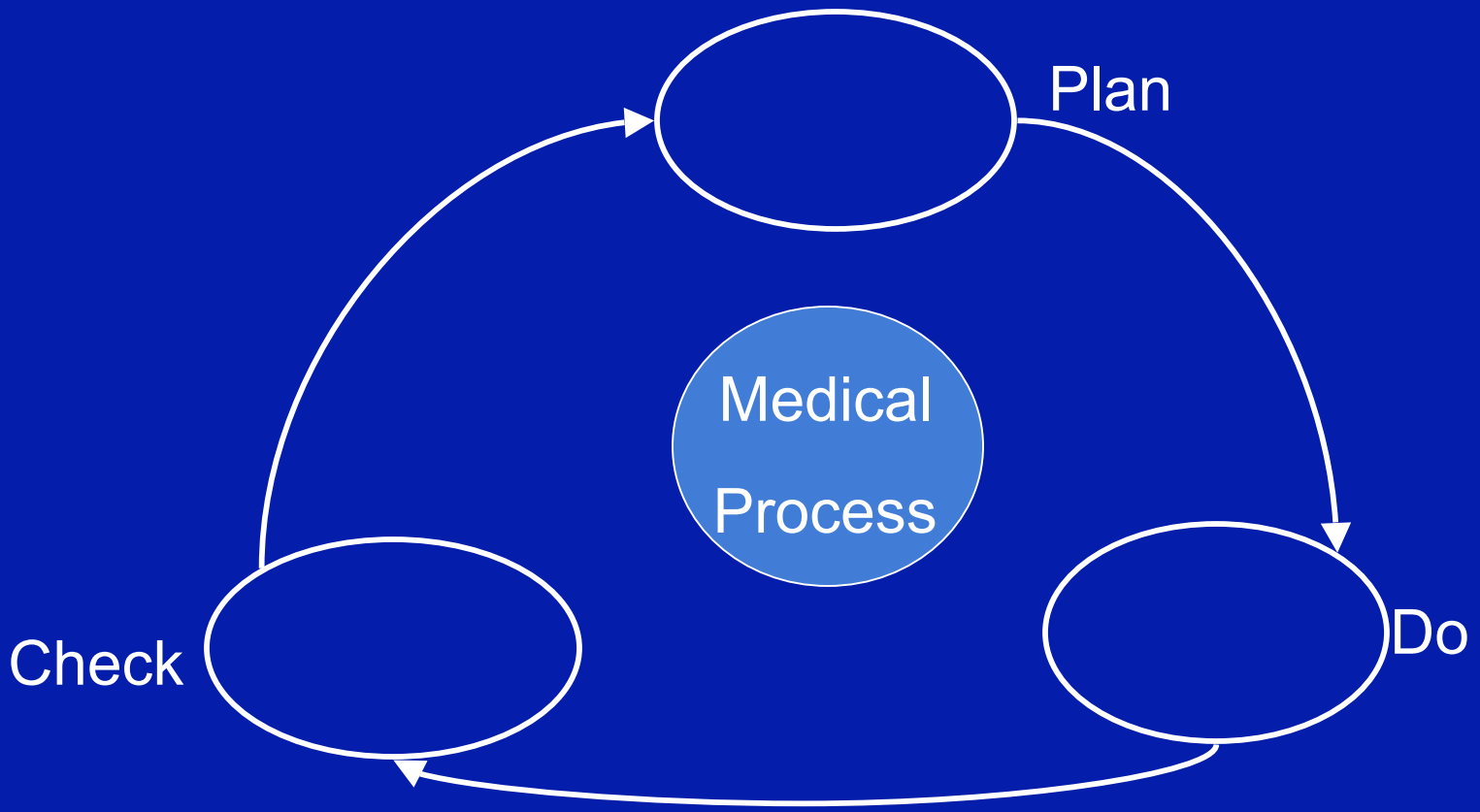
http://laser.cs.umass.edu

University of Massachusetts, Amherst, MA 01003 USA

Goals of the Work

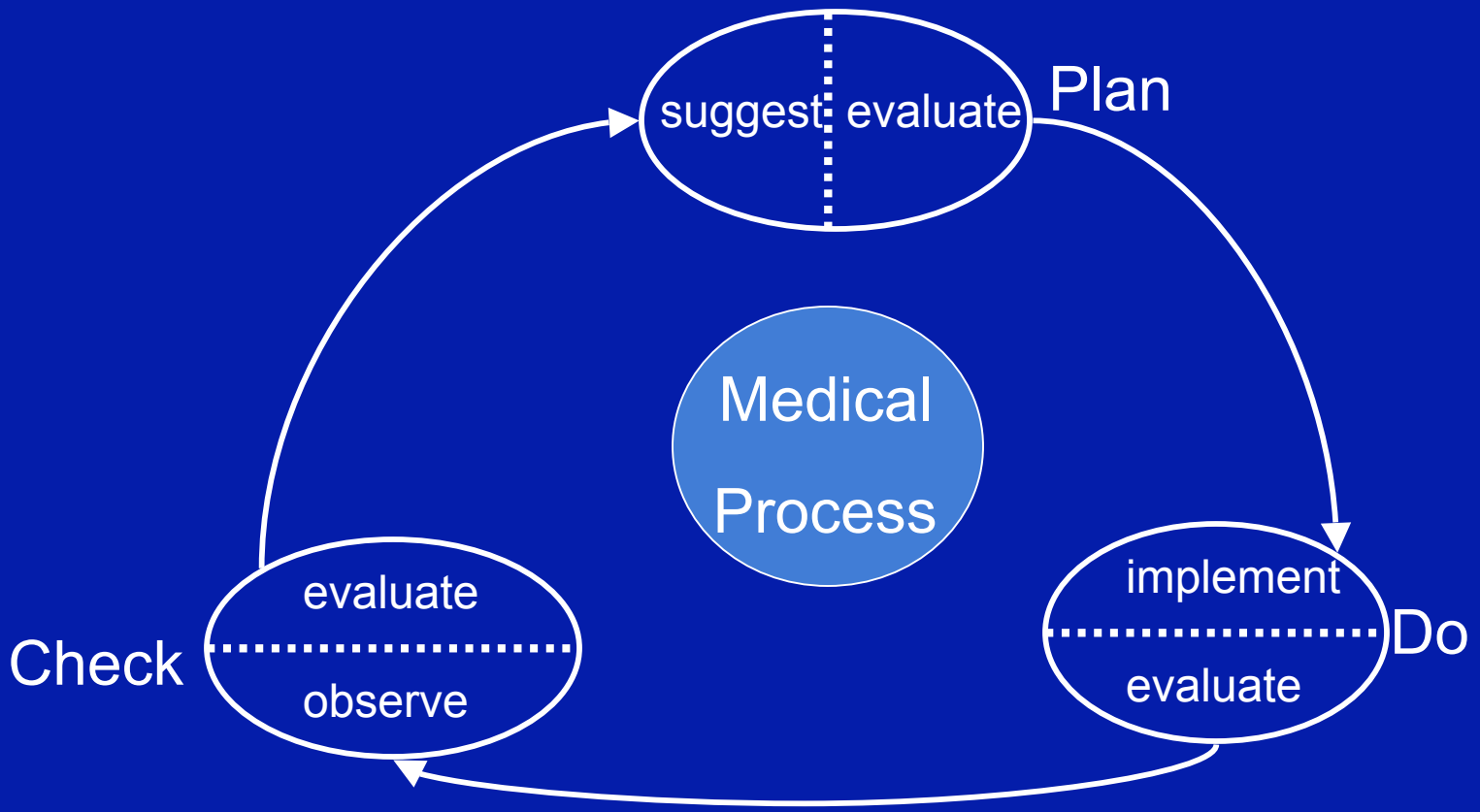
- Improve healthcare processes
  - By finding and removing defects
  - By automating (parts of) them
- Support better training of nurses
- Evaluate applicability of software engineering technology to processes
  - Little-JIL process definition language
  - Propel property definition tool
  - FLAVERS finite state verification system

Continuous (Medical) Process Improvement



Continuous (Medical) Process Improvement

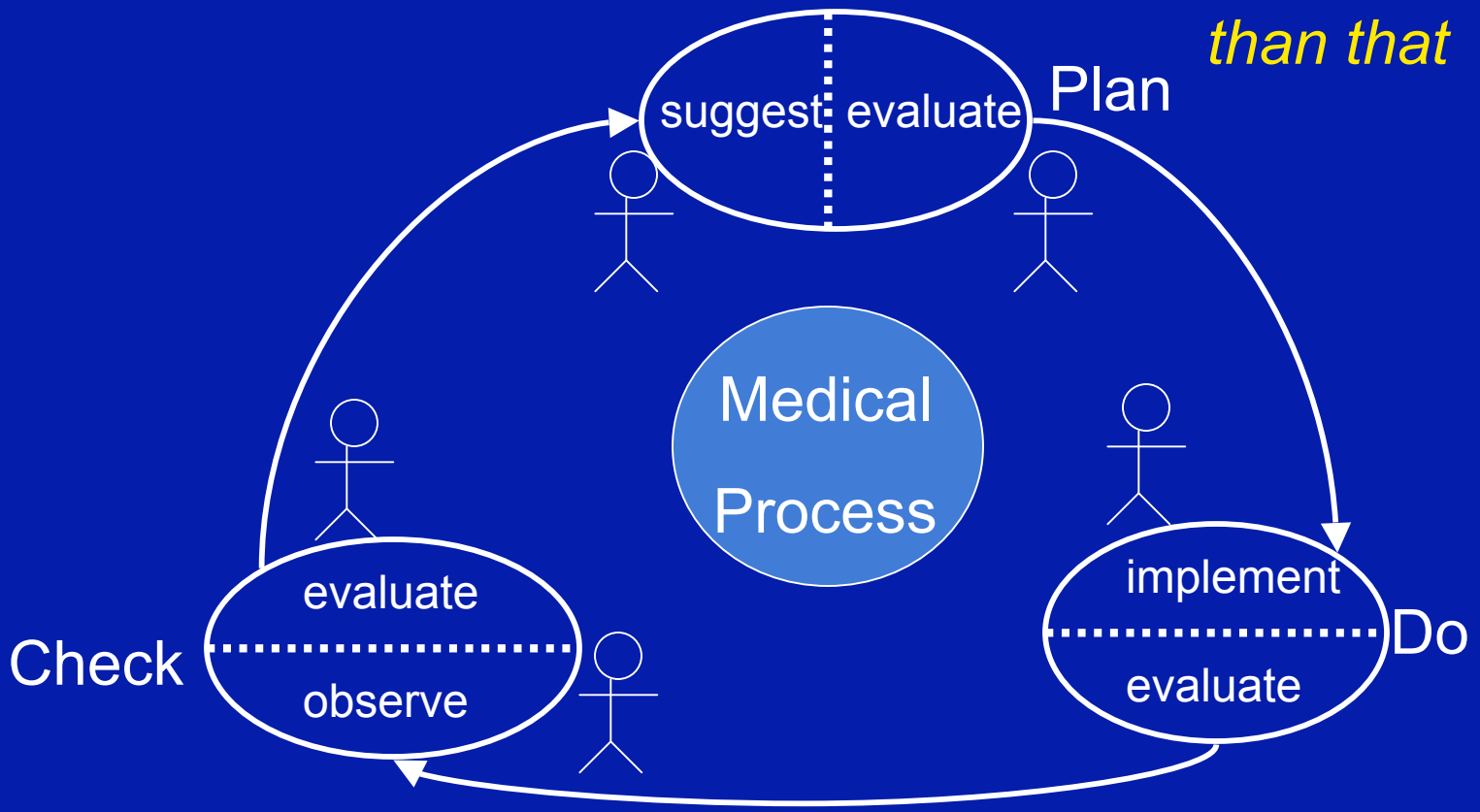
It is Actually more complicated



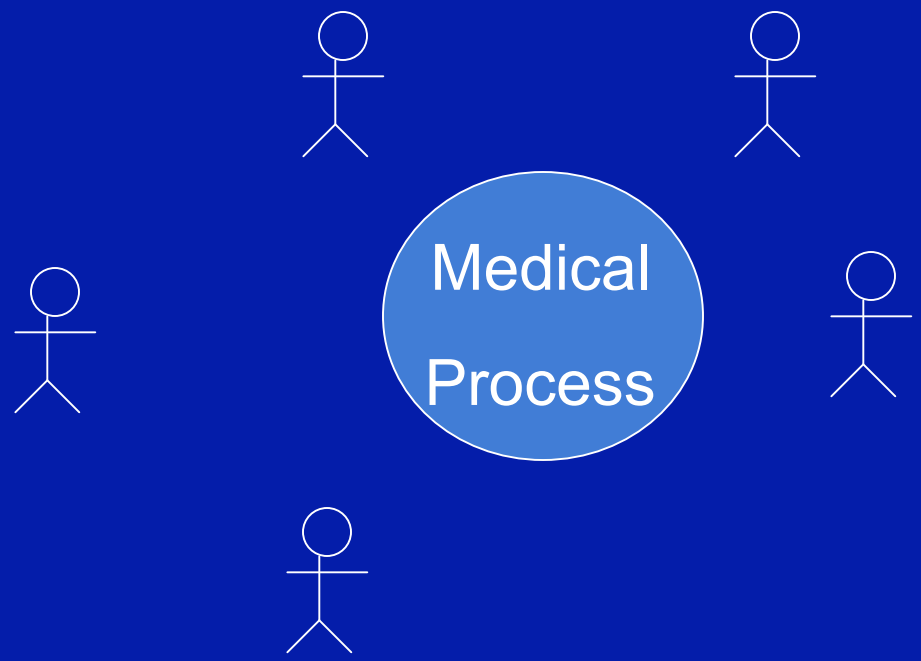
Continuous (Medical) Process Improvement

It is Actually more complicated

and even more complicated than that

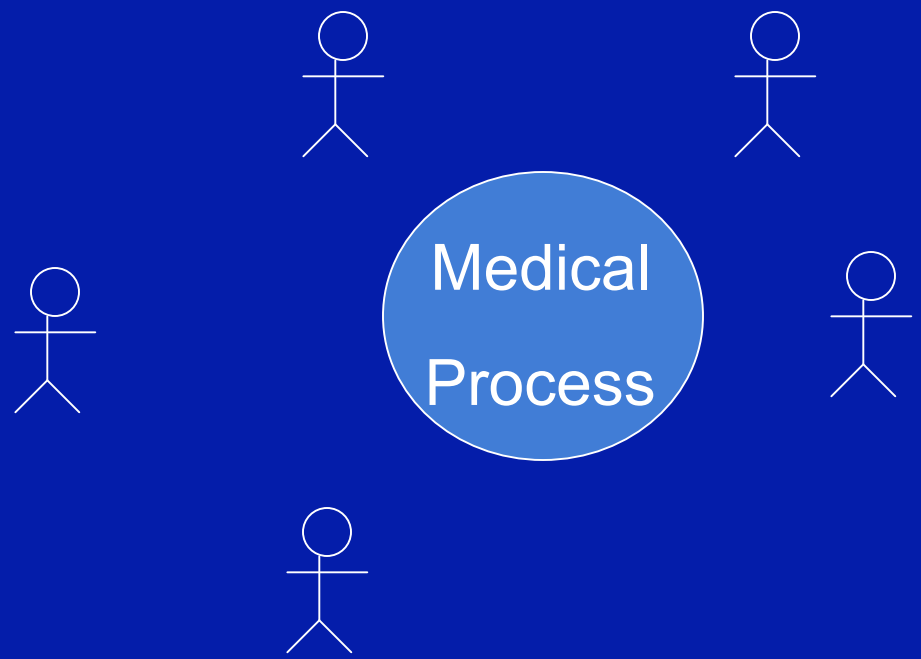


People must communicate and reason clearly and accurately



People must communicate and reason clearly and accurately

Our approach is to do this by means of a precisely defined process



Desiderata for Process Definitions

- Precise
  - Well-defined
- Broad in semantic scope
  - Cover all needed areas
- Sufficiently detailed
  - Capable of being as detailed as necessary
- Clear
  - Easy for domain experts to understand
  - And process developers too

Suggests a Programming Language Idiom

- Artifacts and activities integrated
- Language semantics formally defined
- Resource utilization made explicit
- Agent activities made explicit

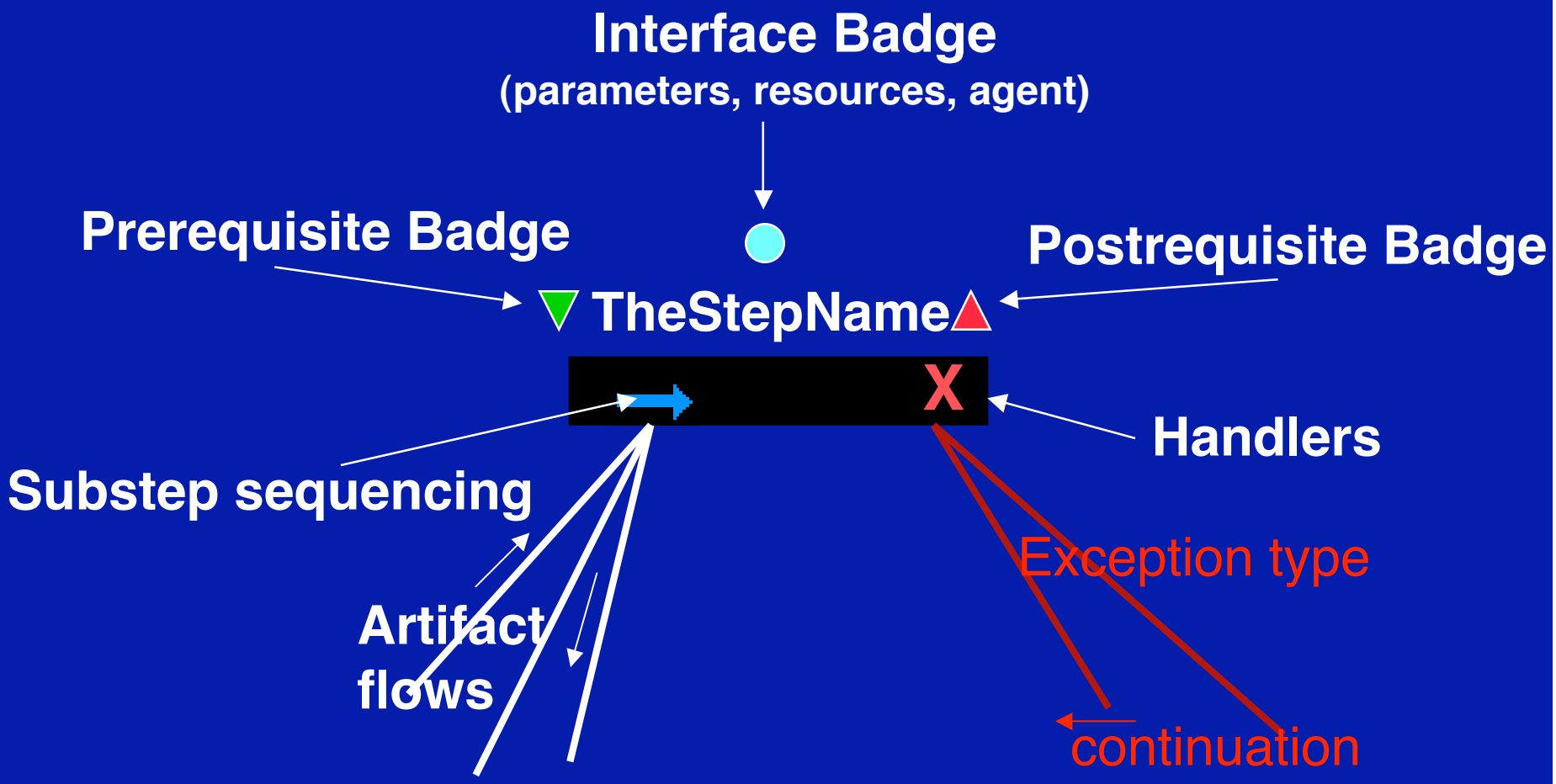
Semantic Capabilities Required

- Proactive control merged with reactive control
- Exception management
- Concurrency control
- Abstraction and modularity
- Integration of human and automated agents
- Timing specification
- Resources
- Transaction management

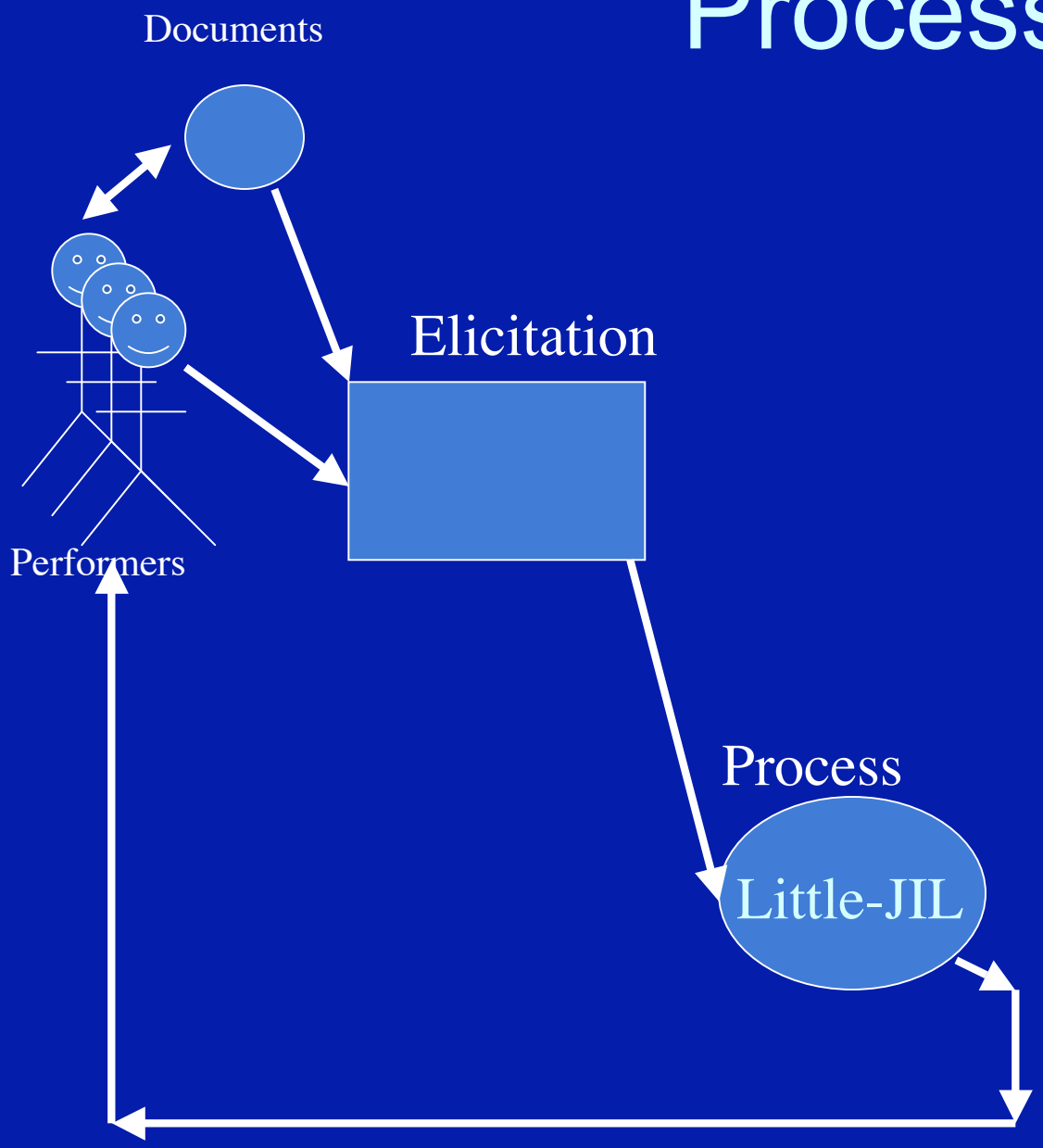
Little-JIL

- Third generation process definition language
- Visual
- Does (most of) the above
- Formally defined
- Visual editor (Visual JIL)
- Interpreter (Juliette)
- Simulator
- Fault Tree generator

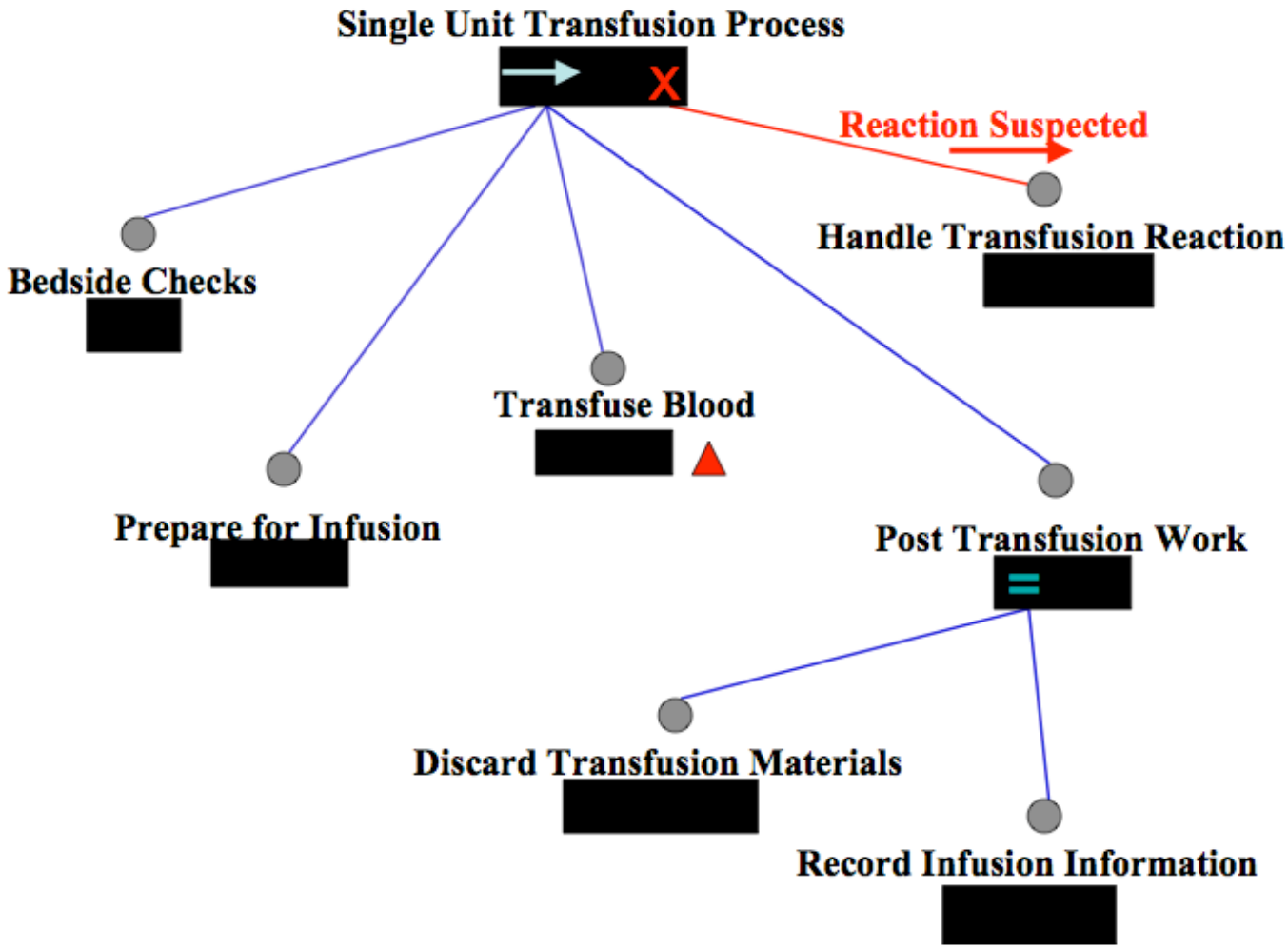
The “Step” is the central Little-JIL abstraction



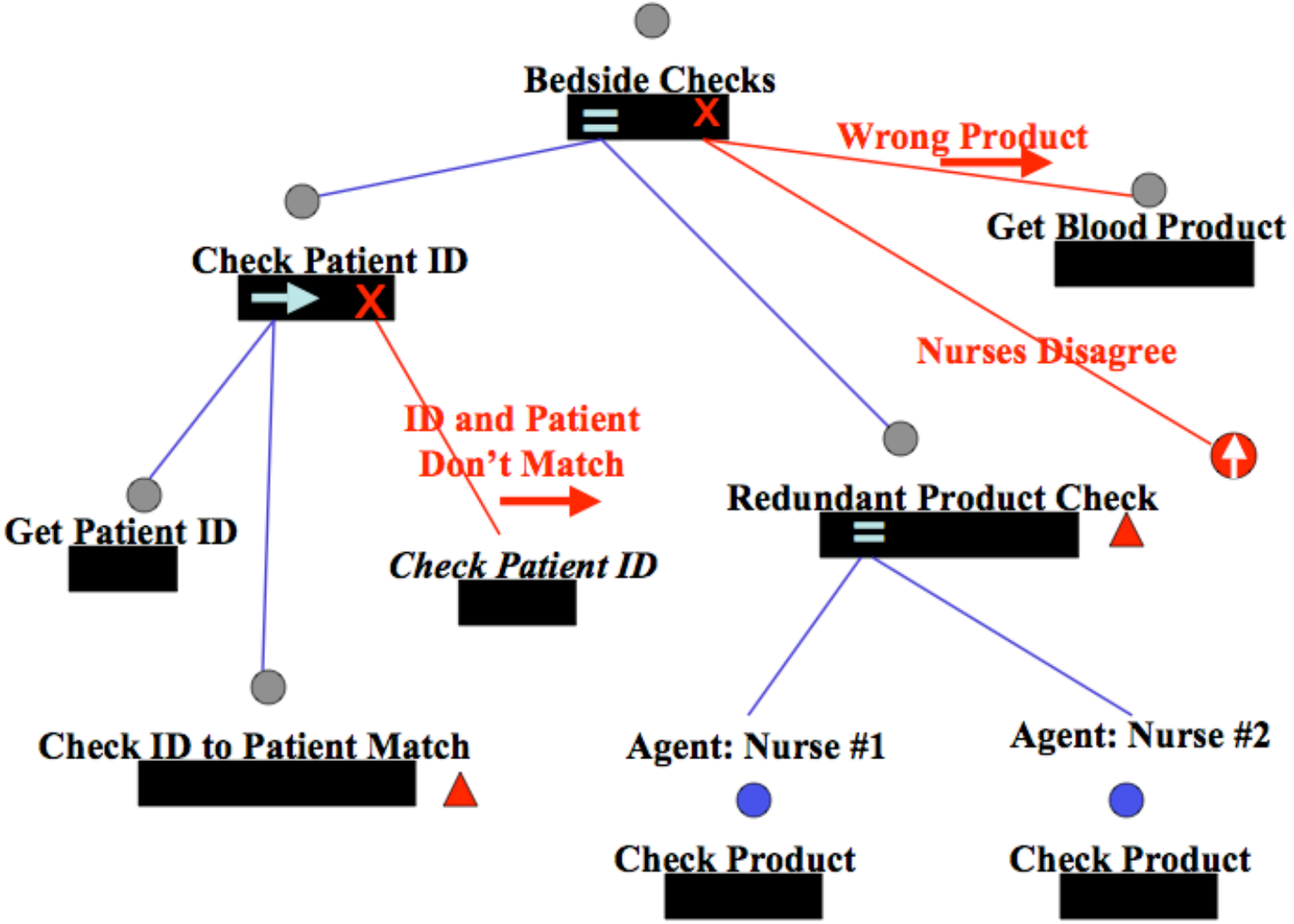
Process Elicitation



Top-level of Transfusion Process



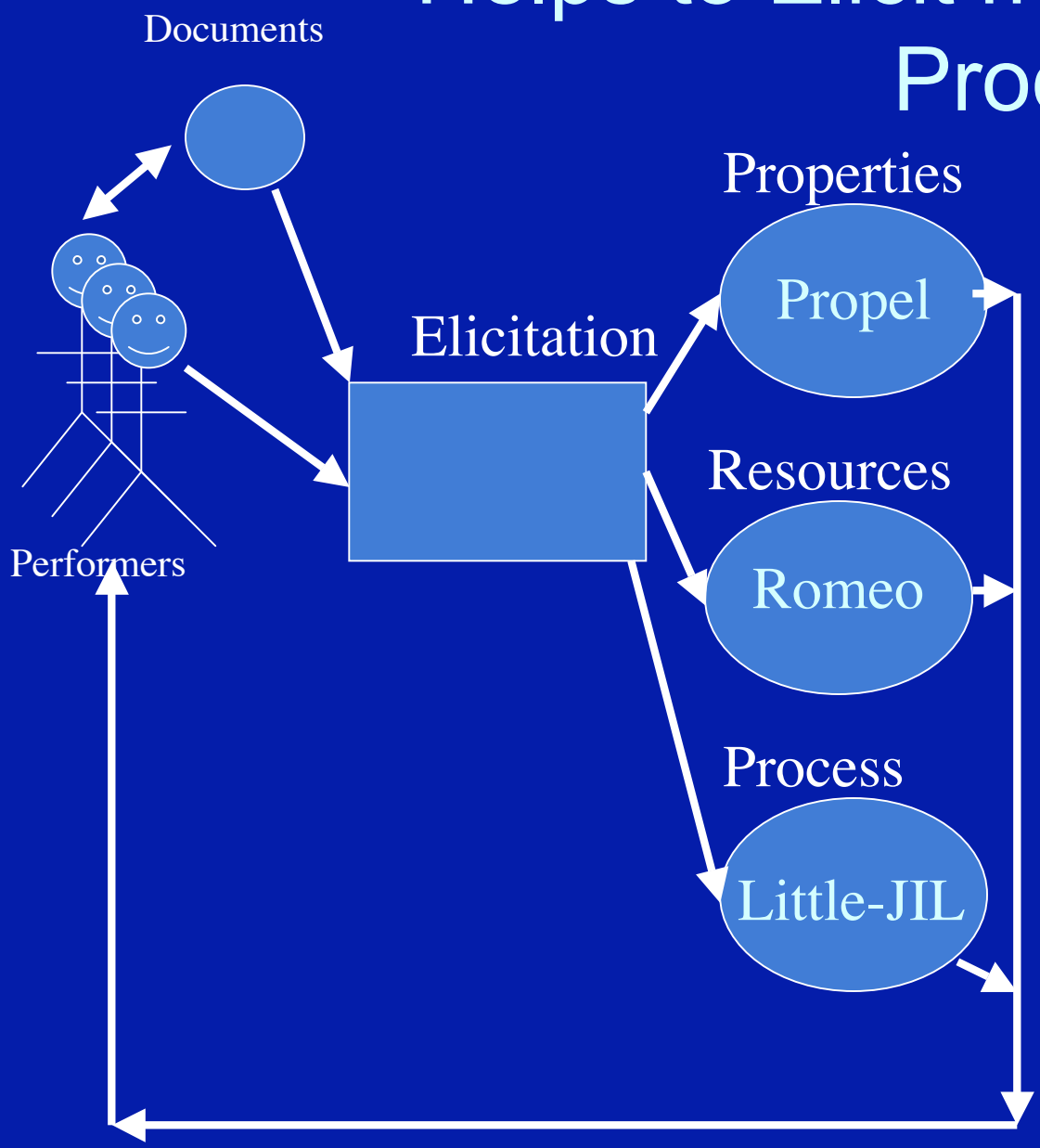
Elaboration of ***Bedside Checks***



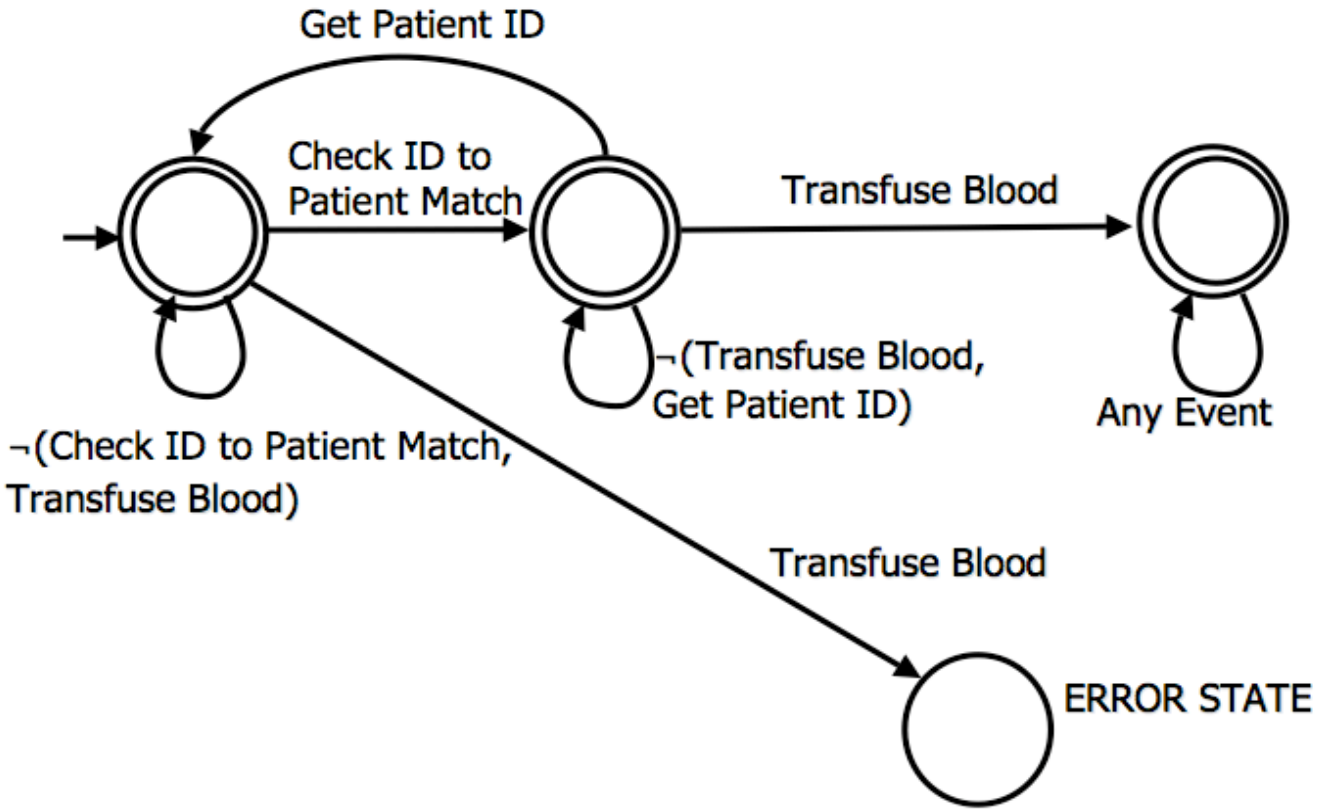
Process Defect Analysis

- Elicit and define required properties
  - As a finite state automaton
  - Using Propel system
- Translate Little-JIL process into concurrency graph
  - Using automated tool
- Apply FLAVERS finite state verification tool
  - Highly efficient model checker
- Verification failure indicates defect(s)
  - FLAVERS produces path where defect occurs

Helps to Elicit more than just the Process

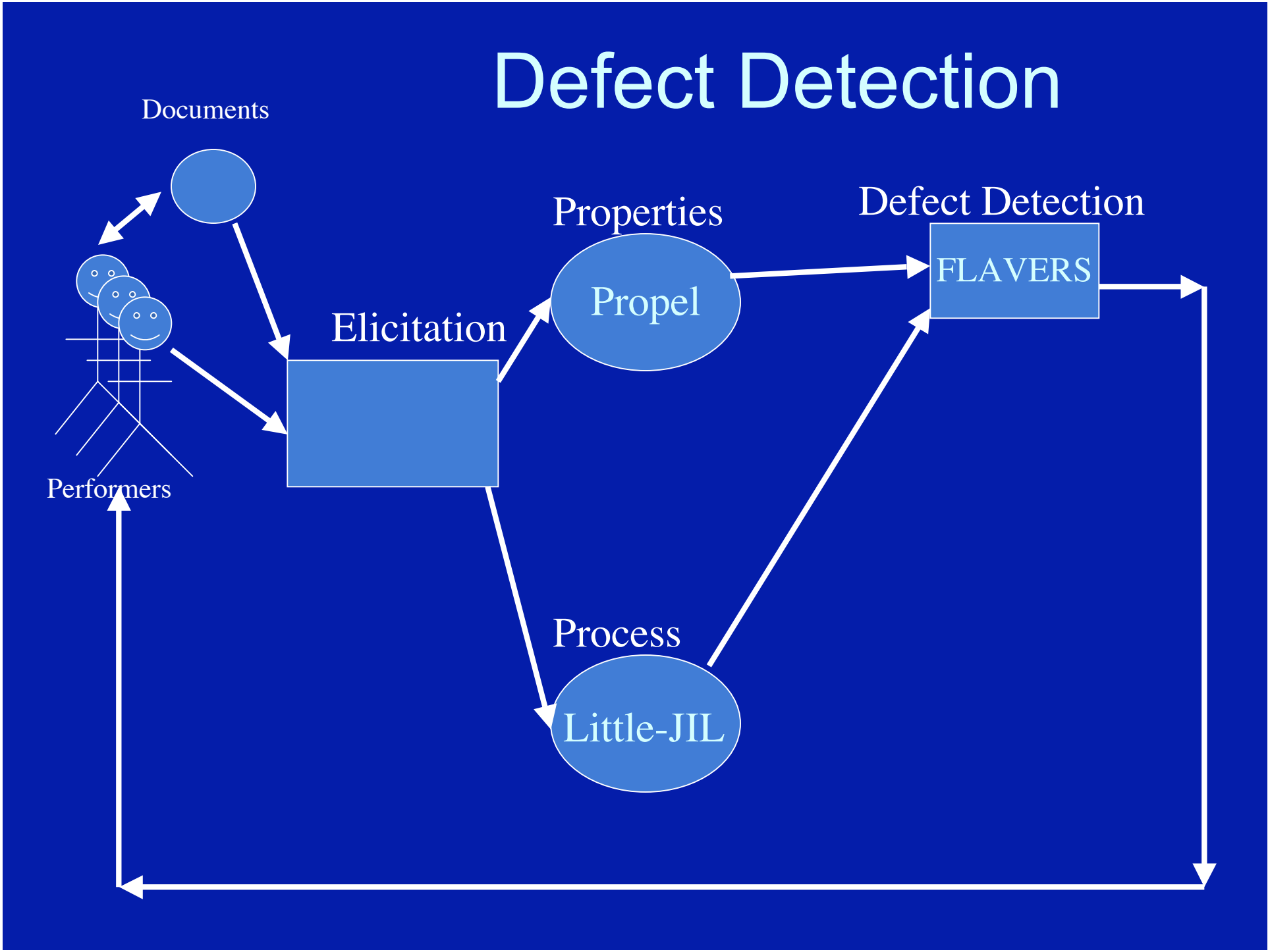


Finite State Automaton Definition of a Key Property

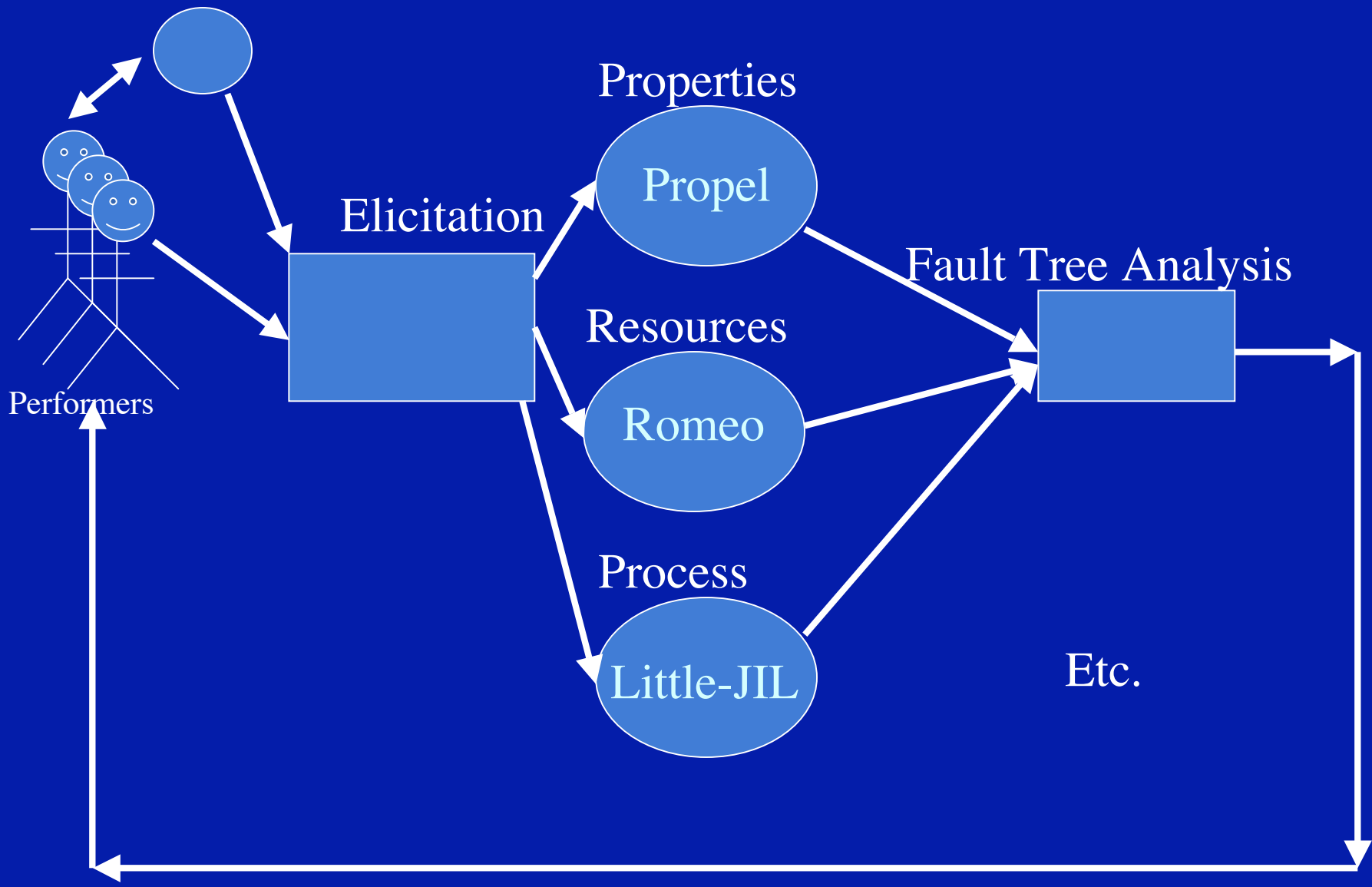


Results

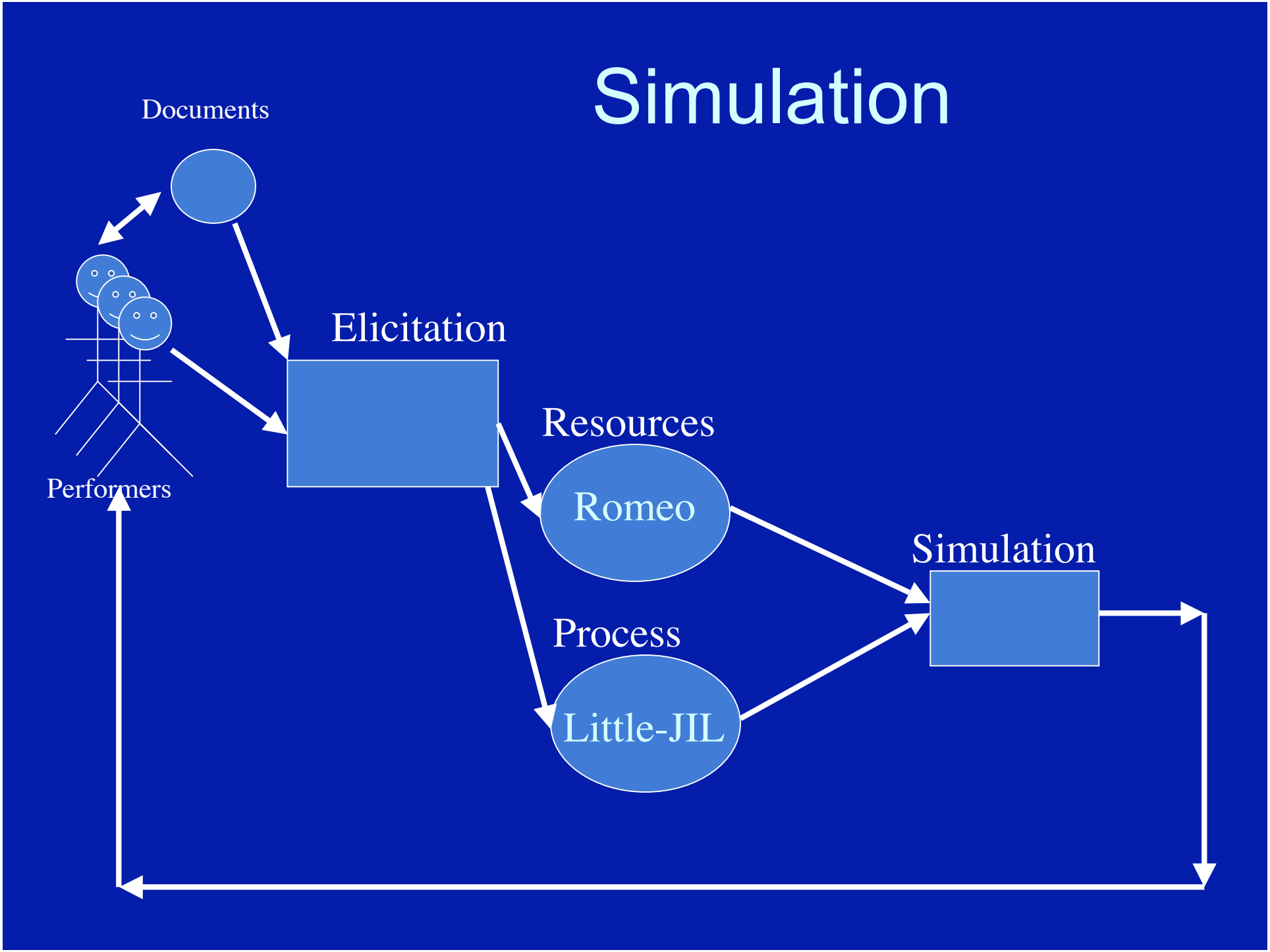
- Defined processes
- Defined properties
- Verification indicated process defects
- Verification took a few seconds of computer time
- Correction of defects is improving processes


Defect Detection
Documents
Properties
Defect Detection
Propel
FLAVERS
Elicitation
Performers
Process
Little-JIL

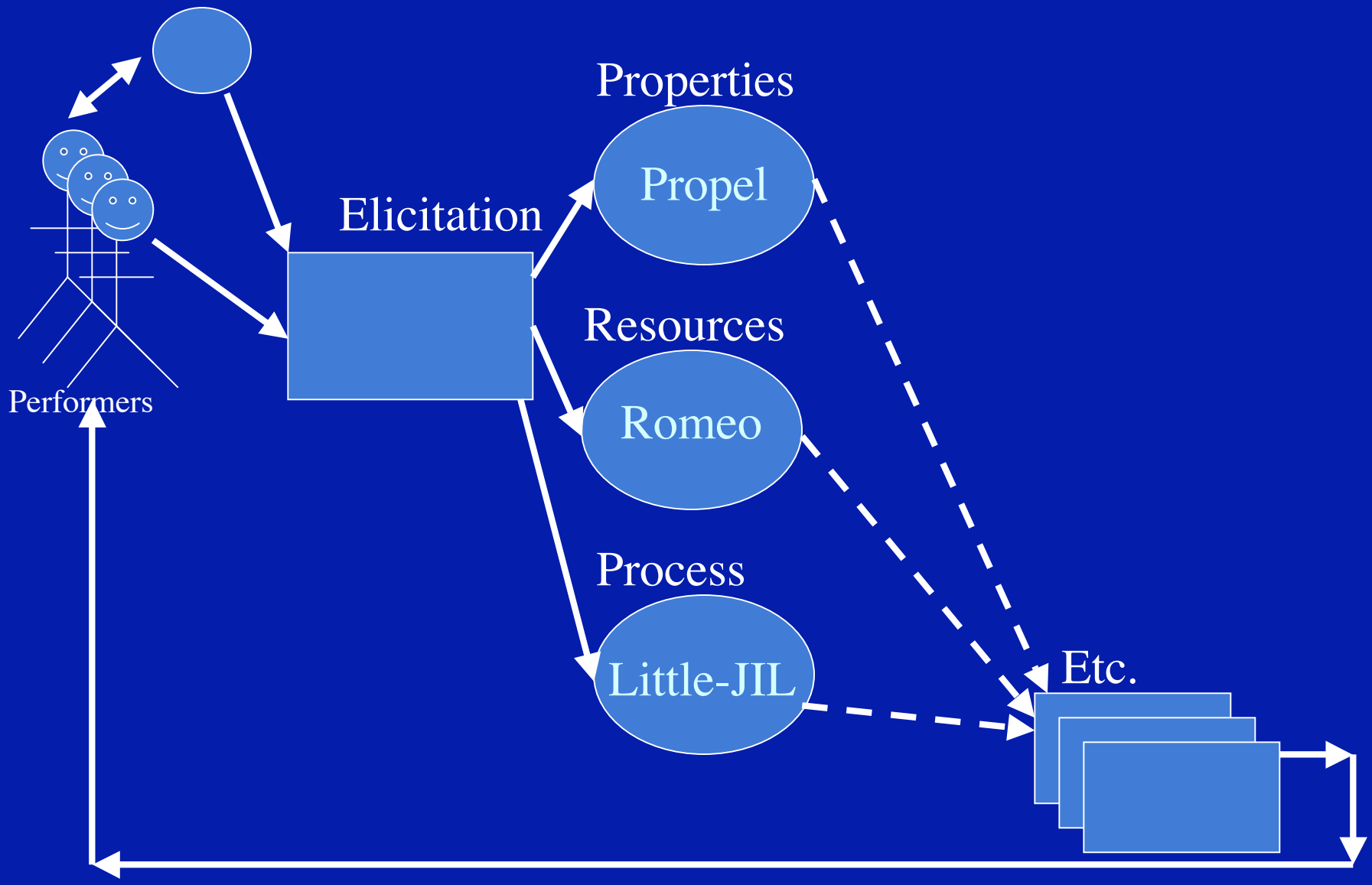
Fault Tree Analysis



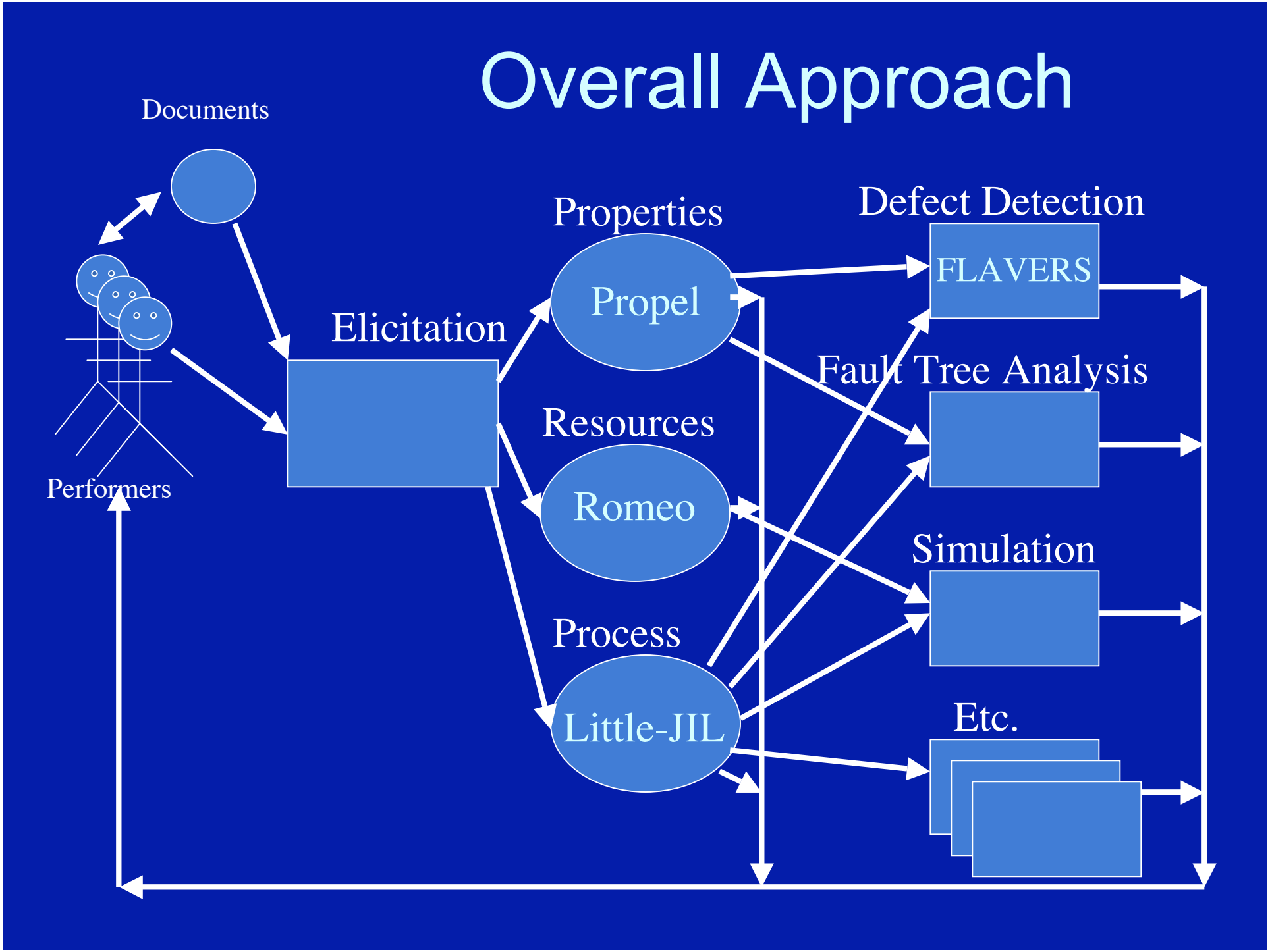
Simulation


Documents
Elicitation
Performers
Resources
Romeo
Process
Little-JIL
Simulation

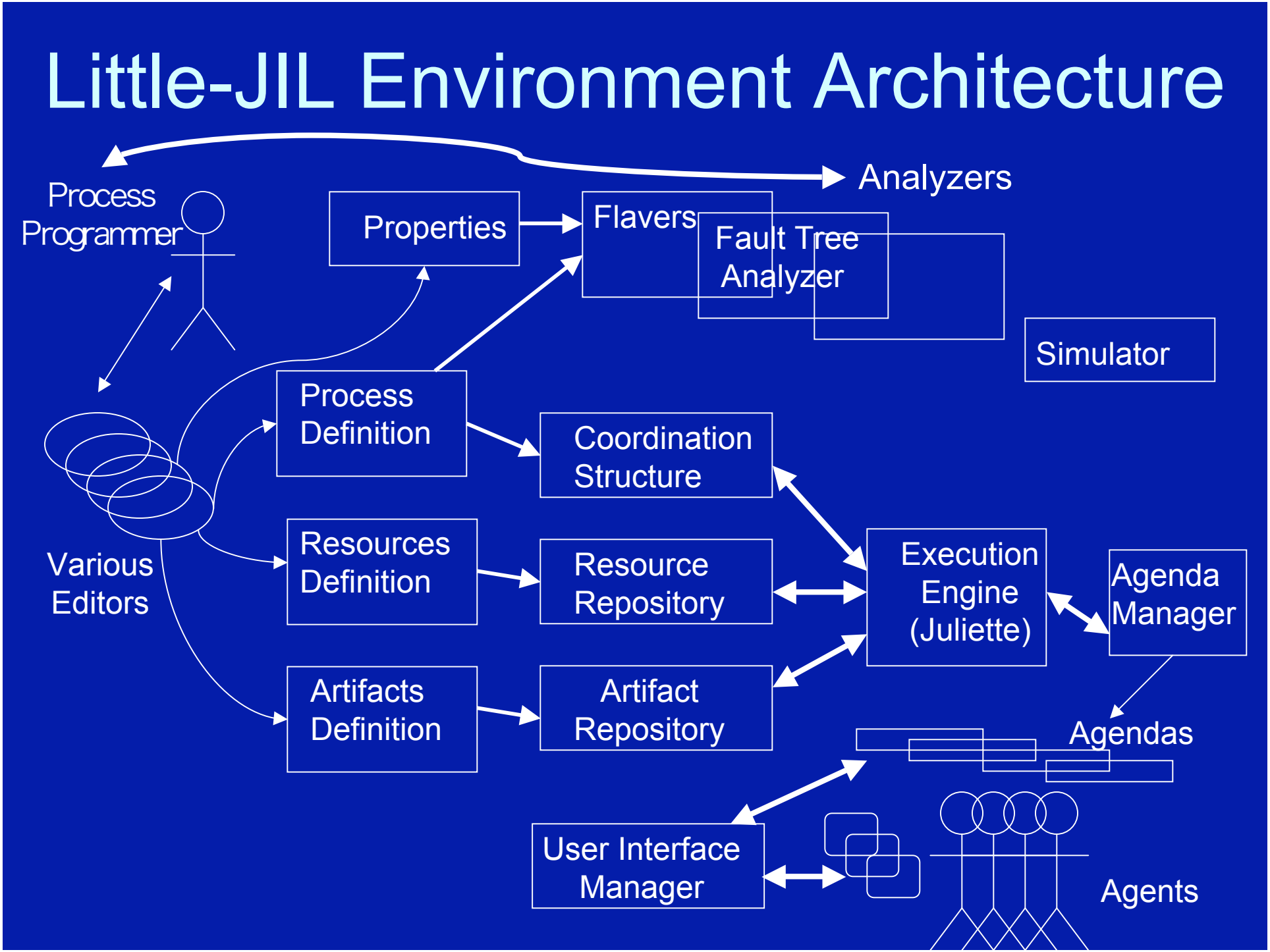
What Other Analyses?



Overall Approach



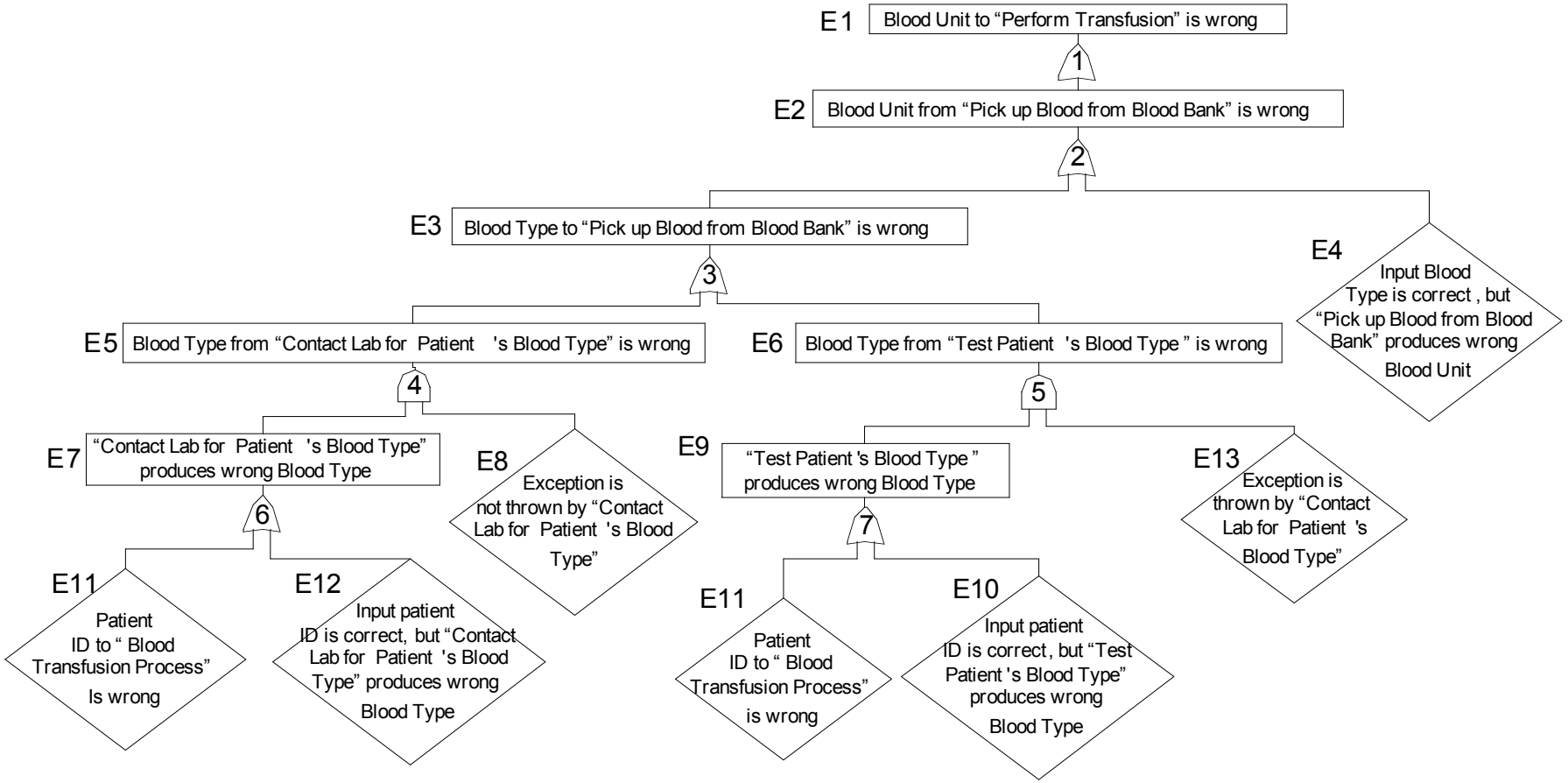
Little-JIL Environment Architecture



Observations

- It is important to elicit both processes and properties, and to do it iteratively
- Real processes have many exceptions
- Real processes are highly concurrent
  - And require coordination/synchronization
- Little-JIL does very well as a definition tool
  - But needs enhancements
- Domain experts can read (but not write) Little-JIL within an hour or so
- Process improvement goes too fast
  - No time to “baseline” original process for evaluation

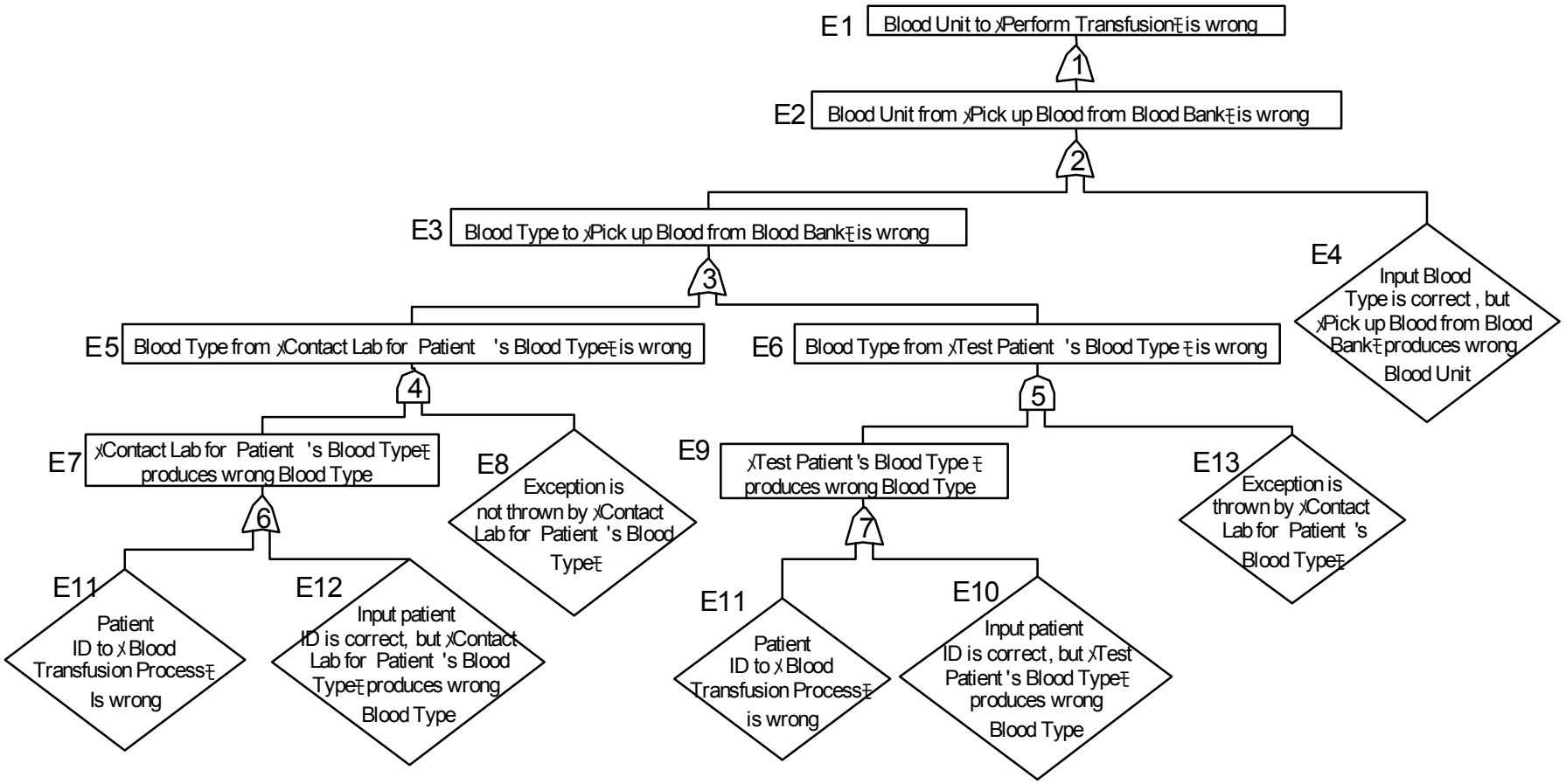
Fault Tree for the Simplified Blood Transfusion Process



Analyze Fault Trees - Calculate Minimal Cut Sets (MCSs)

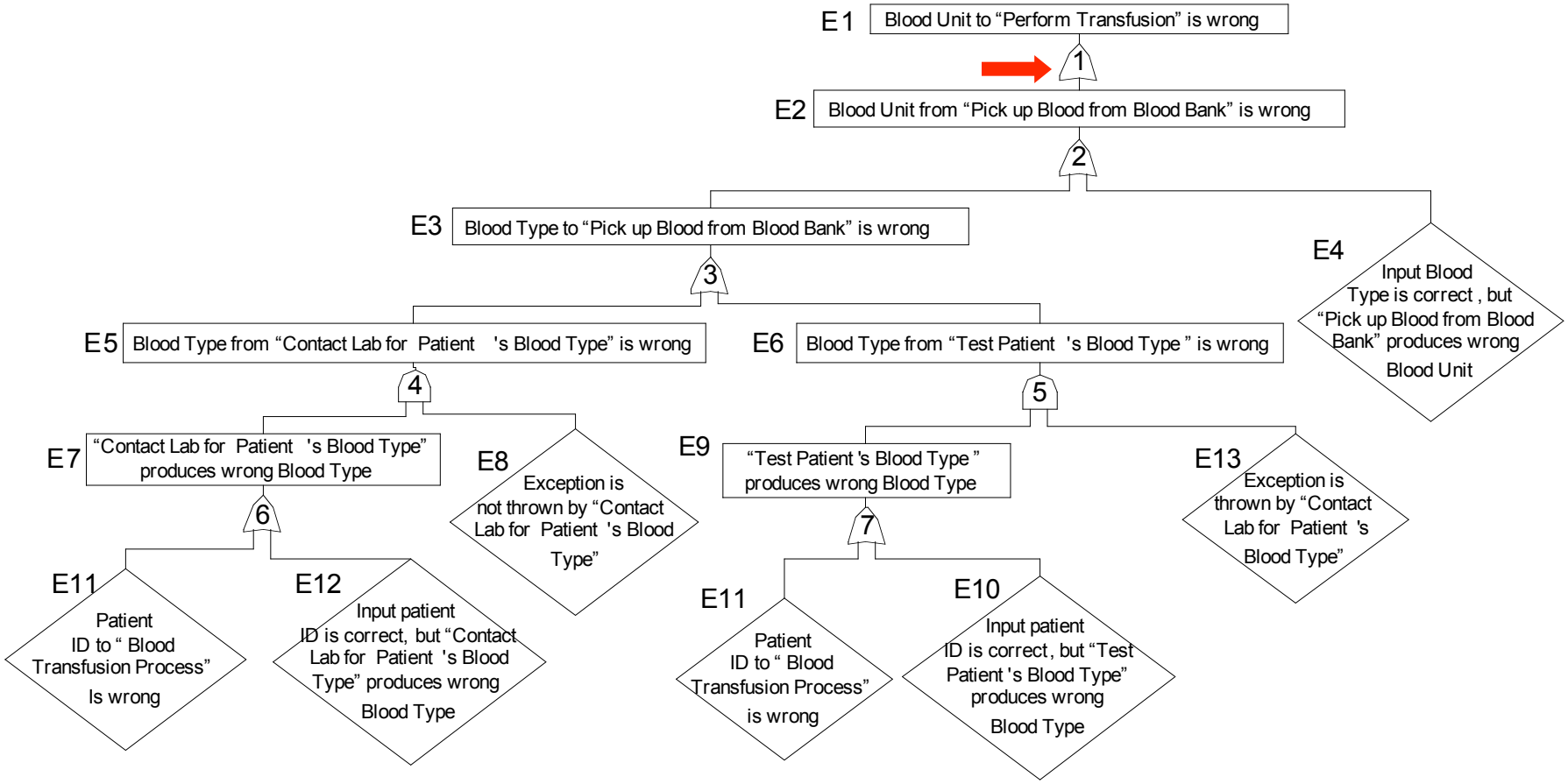
- Cut set - a set of basic events and/or undeveloped events whose occurrence ensures that the TOP event occurs
- MCS - a cut set that cannot be further reduced
- MCSs can be automatically calculated from a fault tree using Boolean algebra

Calculate MCSs



Each gate corresponds to an equation

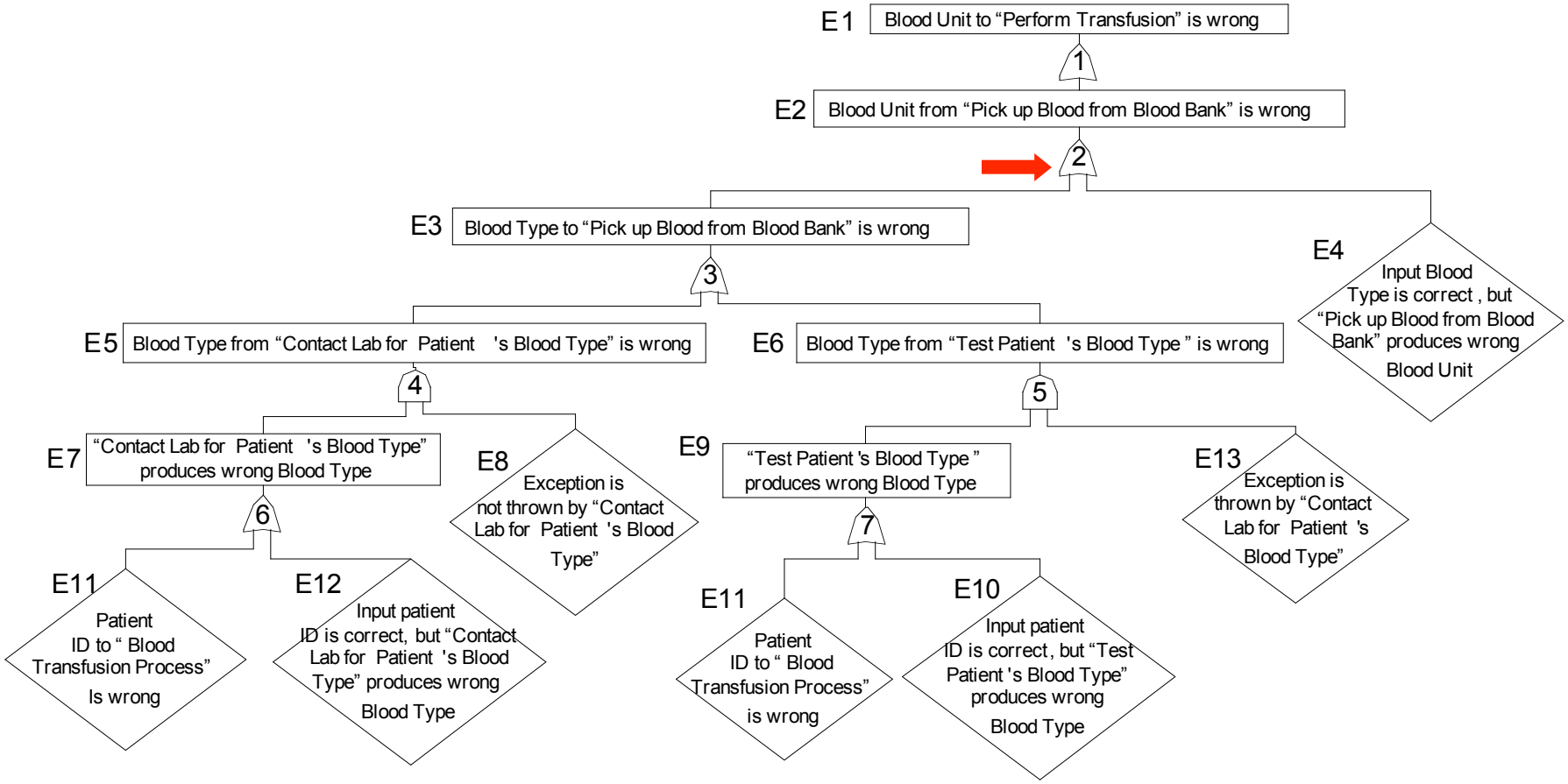
Calculate MCSs



Each gate corresponds to an equation

1: E1 = E2

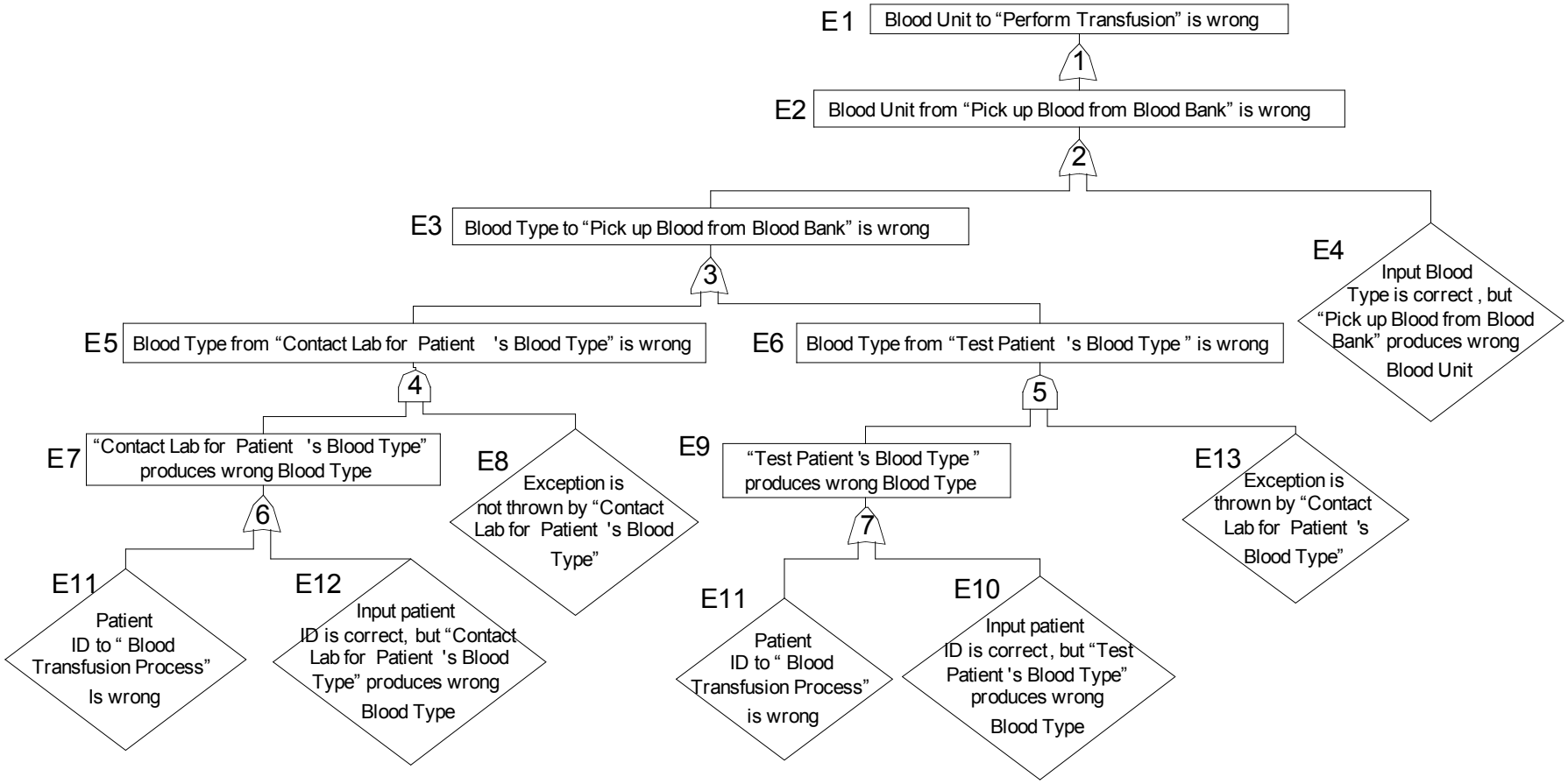
Calculate MCSs



Each gate corresponds to an equation

1: E1 = E2 2: E2 = E3 + E4

Calculate MCSs

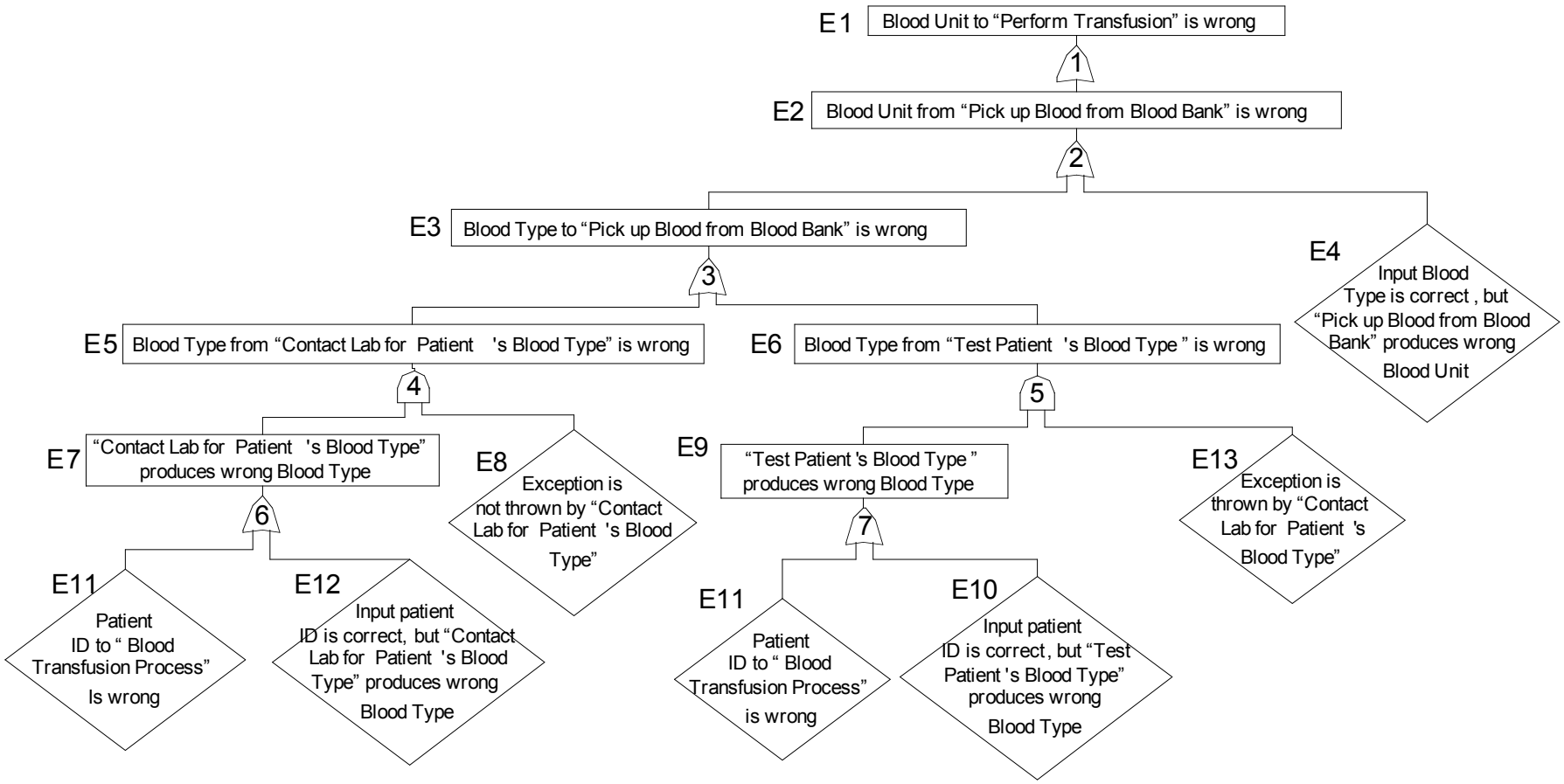


Each gate corresponds to an equation

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

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

Calculate MCSs



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

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

Calculate MCSs (cont'd)

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

MCSs: { E4 } { E11 } { E12, E8 } { E10, E13 }

Analyze FTs – Calculate MCSs (cont'd)

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

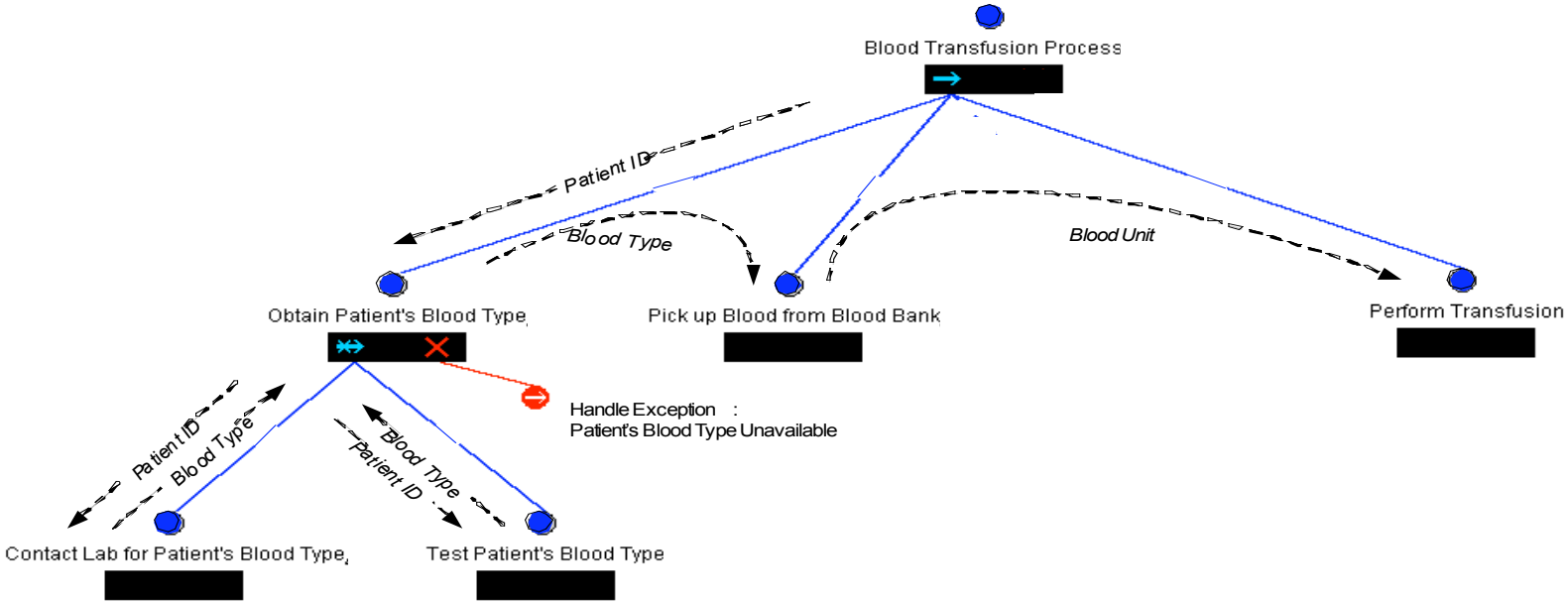
MCSs: { E4 } { E11 } { E12, E8 } { E10, E13 }

Single points of failure ! ! !

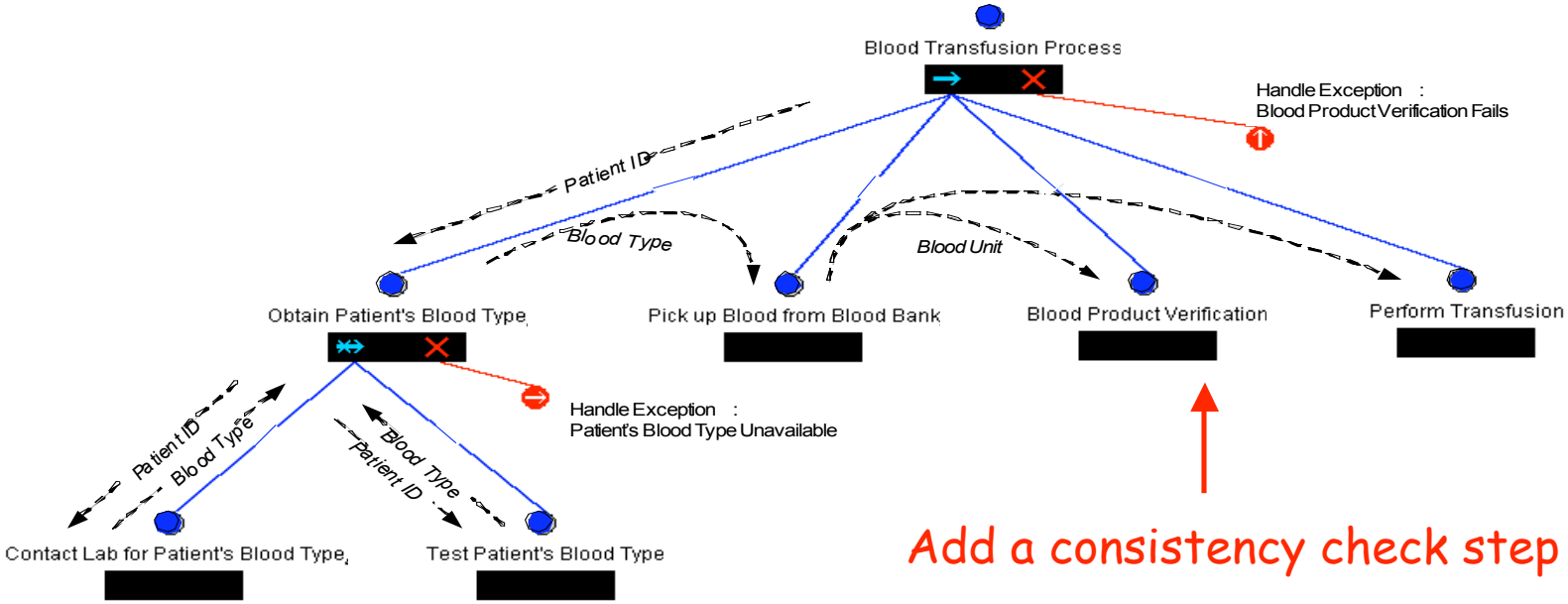
Assess Effectiveness of Process Improvement

- 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

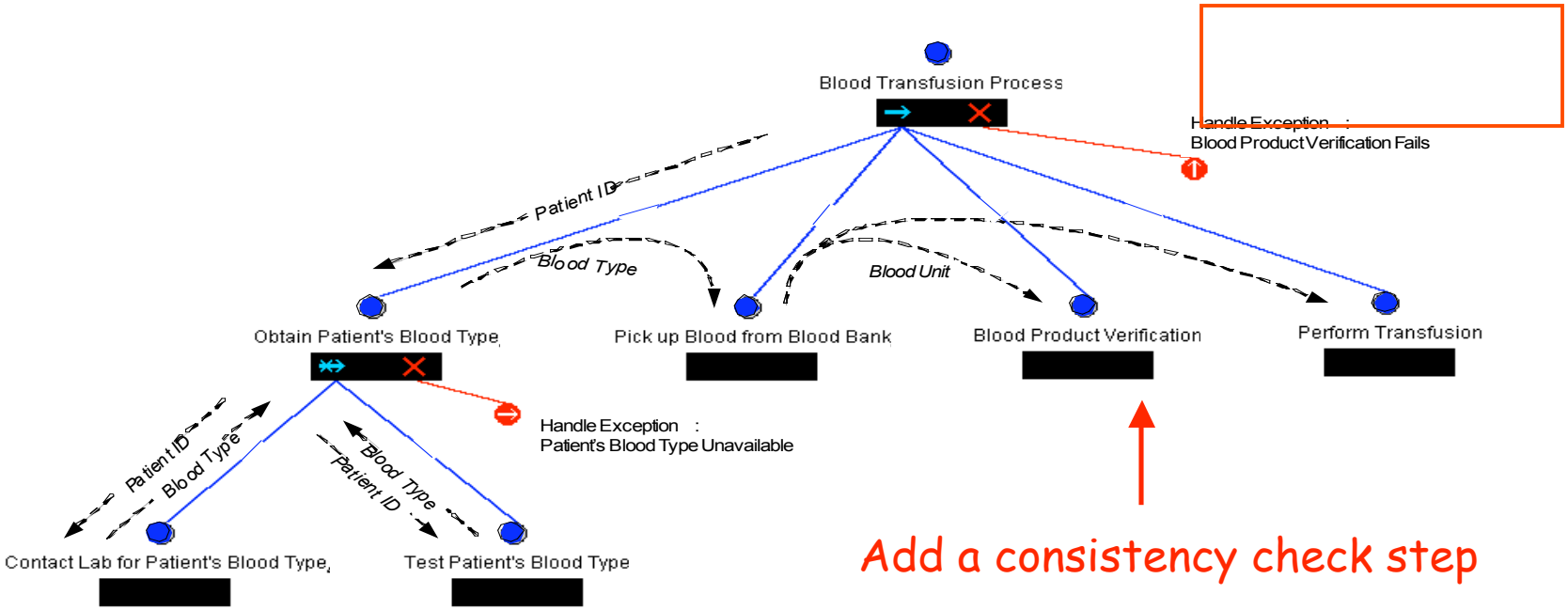
Improve Simplified Blood Transfusion Process



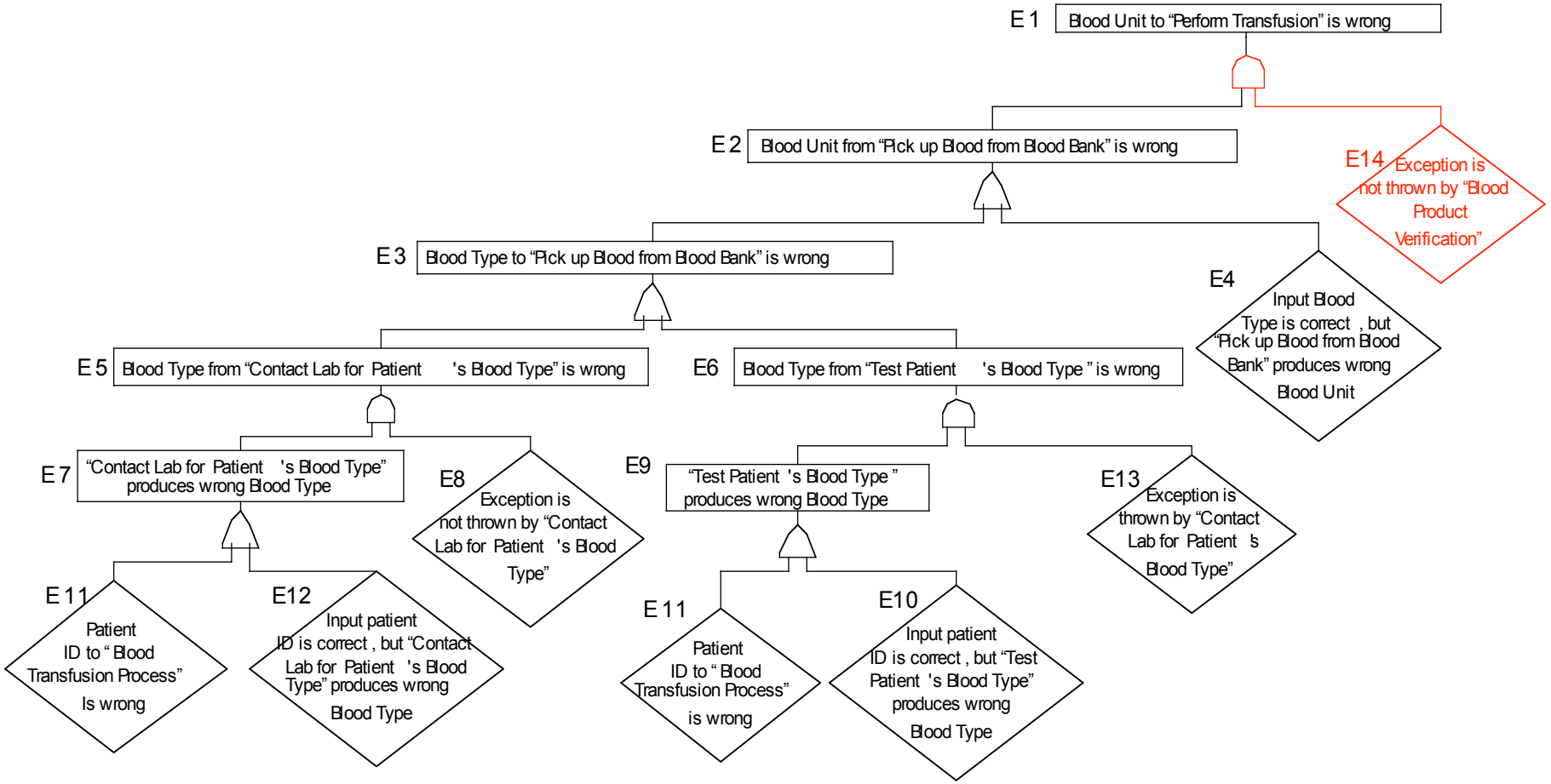
On Alternative to Improve Simplified Blood Transfusion Process



On Alternative to Improve Simplified Blood Transfusion Process



New Fault Tree



New MCSs

{ E14, E4 }
{ E14, E11 }
{ E14, E12, E8 }
{ E14, E10, E13 }

No single point of failure now

New MCSs

{ **E14**, E4 }
{ **E14**, E11 }
{ **E14**, E12, E8 }
{ **E14**, E10, E13 }

No single point of failure now

E14 appears in every MCS

Thus further improvement could be focused on increasing the probability that E14 is correct

Observations

- 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 are derived from a single process definition

Blood Transfusion Example: Generated Fault Tree