CSSE 490 Model-Based Software Engineering: More MBSD

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

Learning Outcomes: MBE Discipline

Relate Model-Based Engineering as an engineering discipline.

- Outline Abstraction and Requirements
- Examine formalisms in representing software
- Discuss KAOS and B Language to show semi-formal approach

Software – It's Big, It's Bad, … and It Gets in Everything

- It's Big / Complex
  - □ Lots of Components Distributed across Net
  - □ Increasingly # and intricacy of interactions
- It's Bad
  - □ Quality and Security …
- And it Gets in Everything…
  - □ Internet Coffee Pot
  - □ Trains, planes, and automobile
  - □ Bank/Mortgage/Finance
  - □ National security systems

What are some of properties of a formal representation form?

How do they support the process of generating software?

- Think for a minute…
- Turn to a neighbor
  and discuss it for a minute

Abstract Representation Form

- A software Need:
  "…We need to be able to share information about ourselves and our activities with our friends."

Share Information

A little Less Abstractly…

- A Software Capability: *"…We accomplish this using a capability that provides accessible, but secure information, about ourselves and our activities…"*

- Of course, there would derived information at this point… access, security, …

- How do we keep track of this information?

At Some Point We Specify the "What"

- Requires must be more specific
  - □ This means that the requirements must be unambiguous, complete, consistent, verifiable/testable, and traceable…

R1: The system must provide an ability to present individual information on a webpage for others to view.

R1.1: The system must allow access to the information for viewing, but protect it from tampering.

R1.1.1: The system must provide requisite security for information about ourselves and our activities.

…R2: The system…

- This can get to be tricky - informality offers flexibility while formality provides requisite specifics

Elaborate and Refine Understanding

- Not a lesson in Requirements, but rather a point about **modeling**…

- Starting with **Abstract** Requirements and through a **process** of **elaboration and refinement**, we successively **transform** them to specifications, models, and ultimately implementation

Elaboration and Refinement…

- We *elaborate* specifications with more and more detail – adding reality to a vision of how things should work
- We *refine* the specifications through activities like refactoring – integrating structure and optimizing for efficiencies
- Systematically, we reduce uncertainties as more concrete information is realized through the engineering process

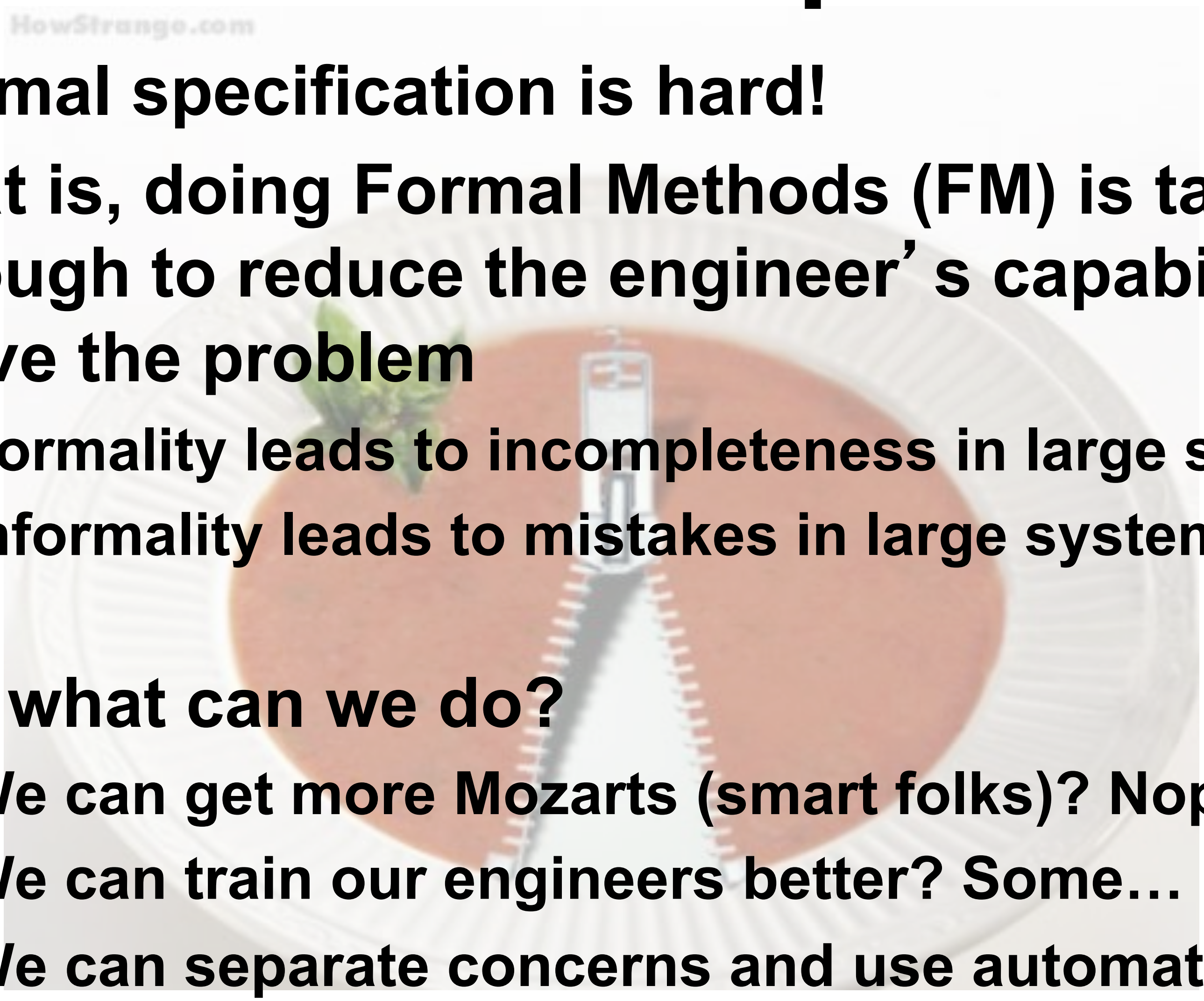
Q2

So, can we Automate some of This?

- Bohner' izm: Objective of requirements engineering is to produce unambiguous, complete, consistent, verifiable, traceable specifications of what the system does from an external perspective.
- Manual methods contain some clues…
  - Above attributes of requirements specifications are the goals of formal specification
- The more formal the representation the more provable and more automateable the process to transform them into implementation!

Q3, 4

BUG in the Formal Soup…



- Formal specification is hard!
- That is, doing Formal Methods (FM) is taxing enough to reduce the engineer' s capability to solve the problem
  - □ Formality leads to incompleteness in large systems
  - □ Informality leads to mistakes in large systems
- So, what can we do?
  - □ We can get more Mozarts (smart folks)? Nope…
  - □ We can train our engineers better? Some…
  - □ We can separate concerns and use automation to help support the load? Maybe more…

Q5

Waiter, there's a Human in my Soup…

Waiter, there is a fly in my soup
And it's having a really good time
Demotivation.us

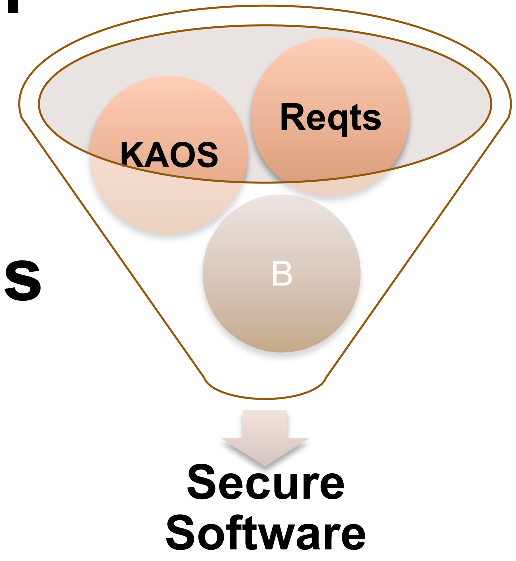
More seriously, …
How do we convert informal requirements into representations that can be used to generate code?

- Think for a minute…
- Turn to a neighbor
  and discuss it for a minute

Going Semi-formal to get to Formal

- Can we do some pre-conditioning of the requirements to get them into a form that we can use formal methods to transform them?

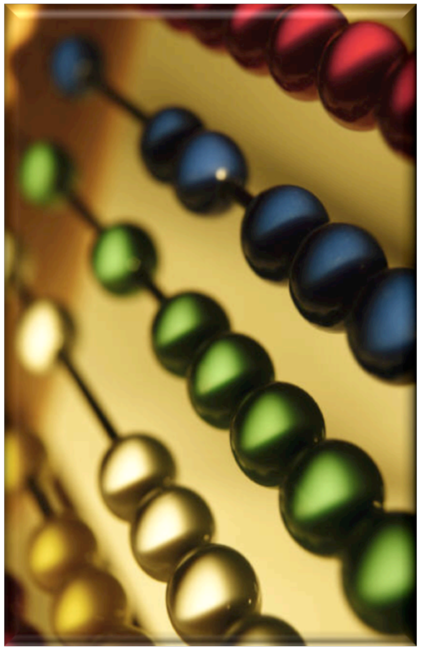
- A goal-directed approach provides some of this scaffolding in this example of formalizing security requirements for generation
  - □ KAOS
  - □ B



Let’s Get Formal (but not too formal, too soon)

Understanding and Formality

- Understanding nascent → Informal
- Understanding forming → Semi-Formal
- Understanding specific → Formal



Q6

KAOS

Knowledge Acquisition in autOmated Specifications

Goal-oriented approach for eliciting, analyzing, & modeling requirements (functional & non-functional)

- Requirements are represented as goals (intuitive)
- The formal underlying framework is based on first-order temporal logic
- Results in a requirements model in the form of a directed acyclic graph (obstacle & impact analysis)
- Assigning agents to goals aids in visualizing responsibility

The B Method

- Popular formal method for developing software systems

- Starts with a very abstract model

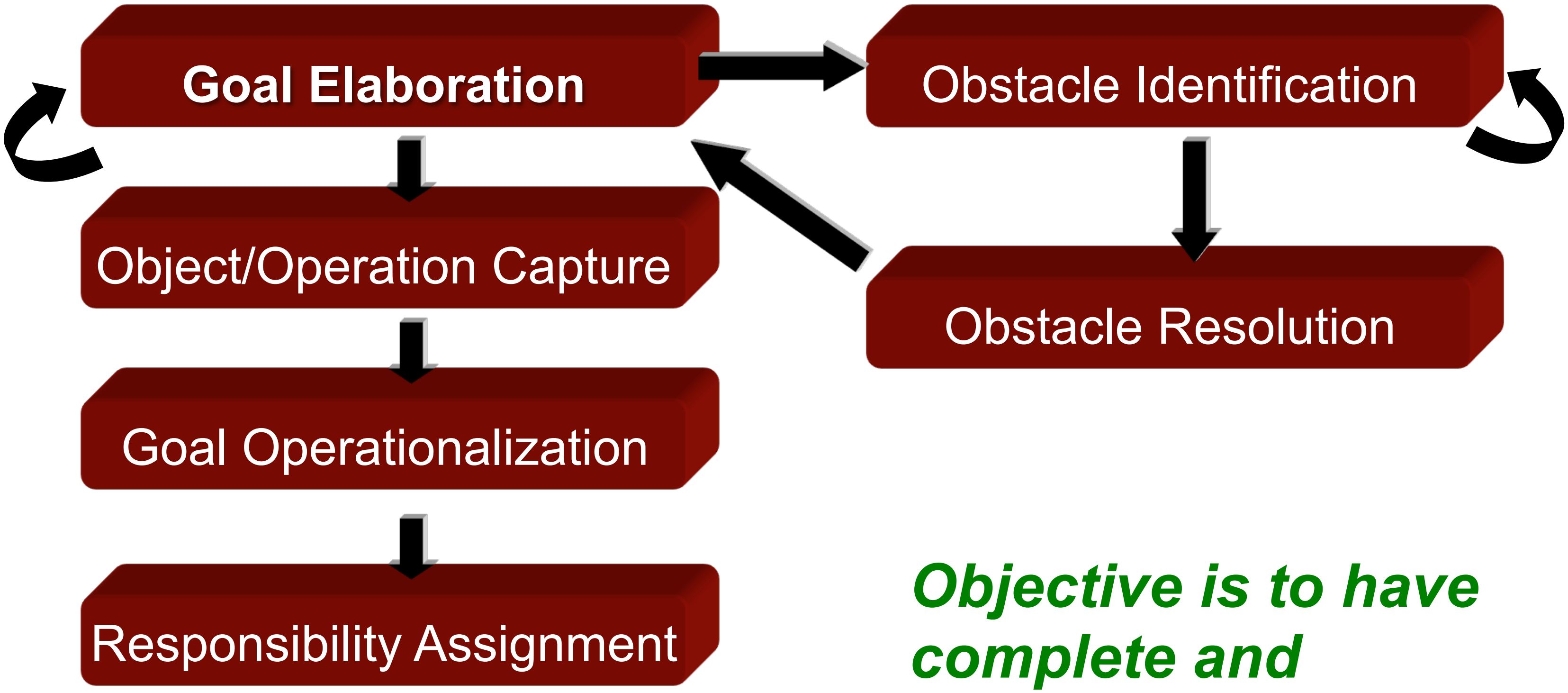
- Preserves proven system properties in refinement

- Provides for correctness by construction
    - Guarantees system correctness…

Q8

KAOS: Elaboration & Obstacle Analysis

Highly Iterative, Goal-Directed



Goal Elaboration

Object/Operation Capture

Goal Operationalization

Responsibility Assignment

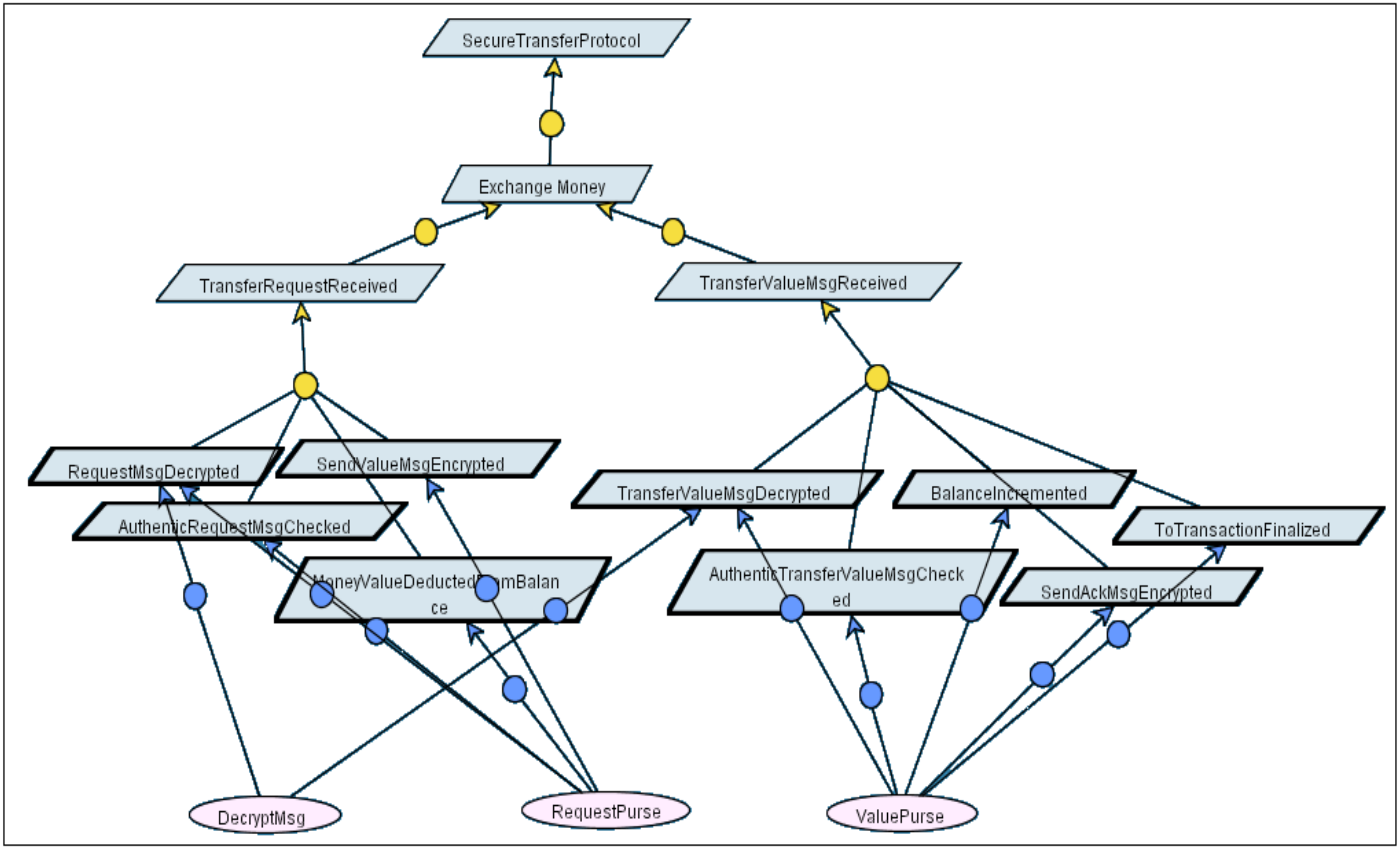
Obstacle Identification

Obstacle Resolution

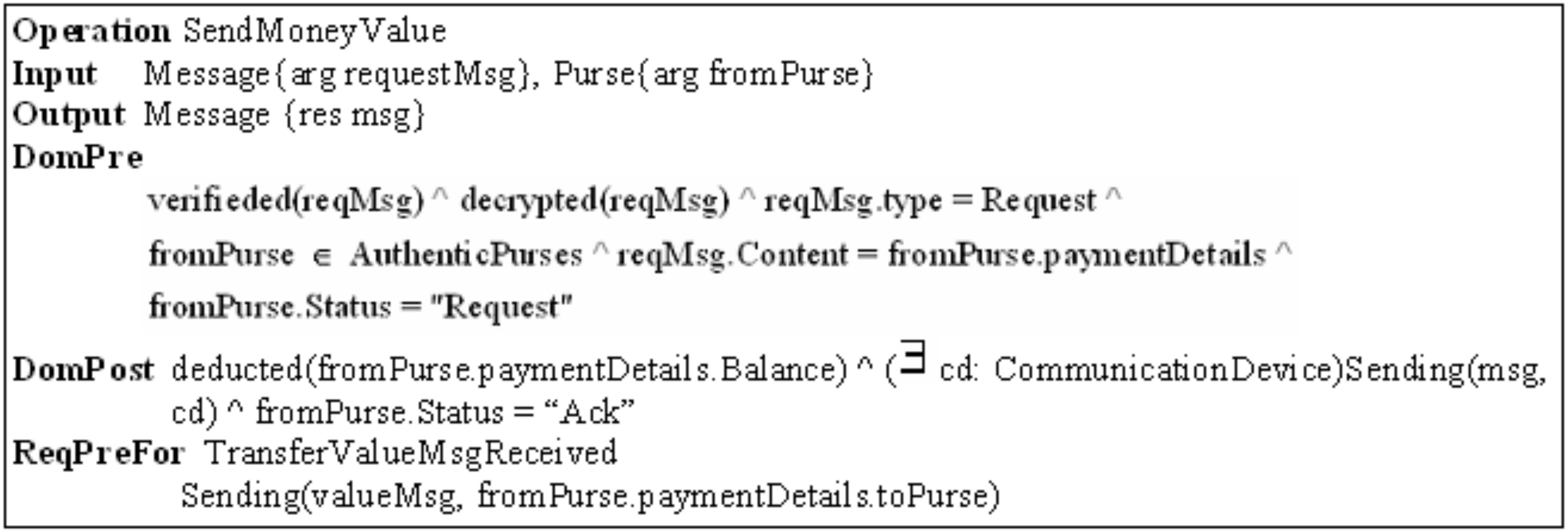
Objective is to have complete and well-organized requirements.

Source: Van Lamsweerde

Example: Electronic Smart Card Goal Graph



Goal Operationalization

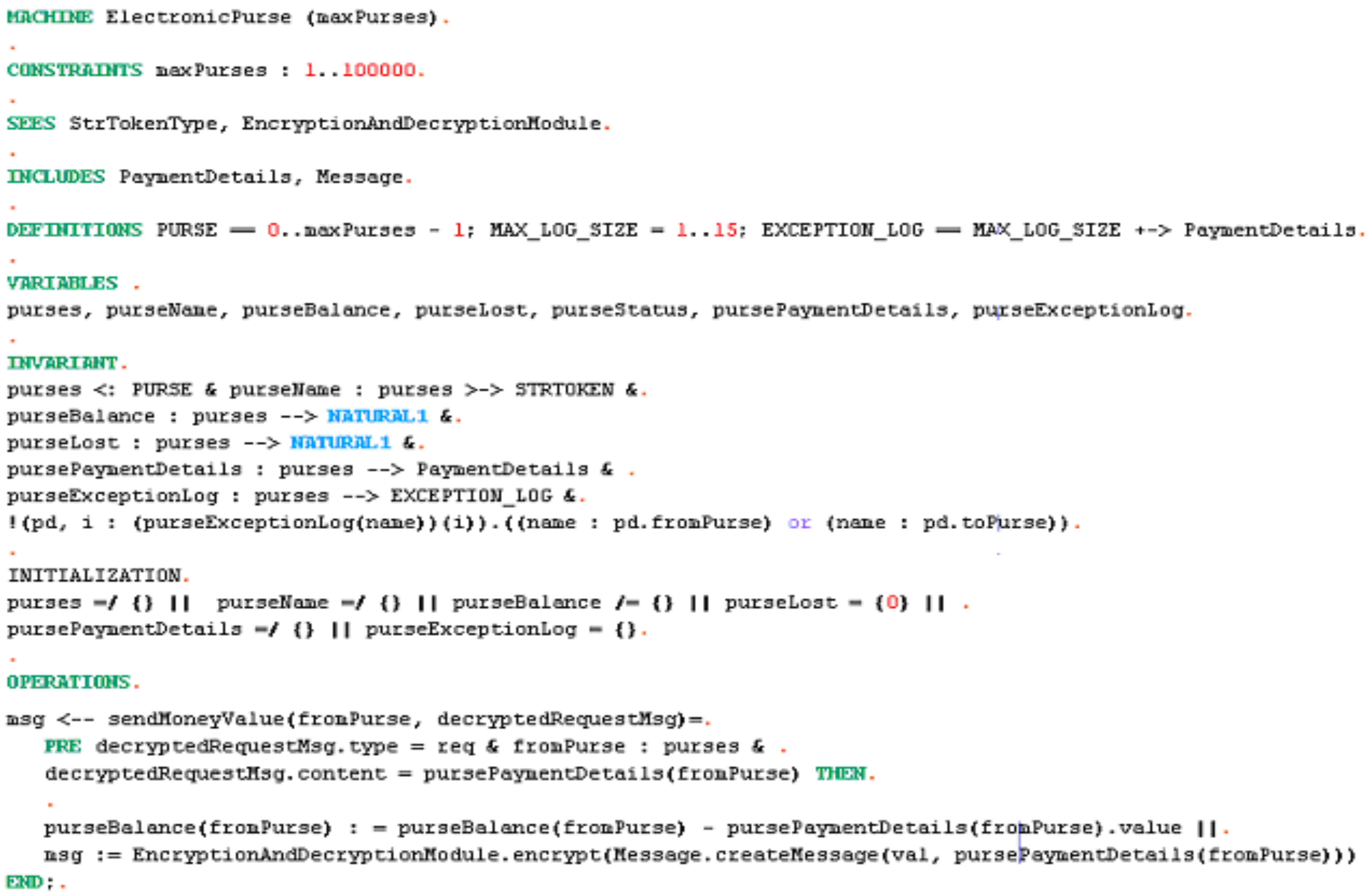


Operation SendMoneyValue
Input Message{arg requestMsg}, Purse{arg fromPurse}
Output Message {res msg}
DomPre

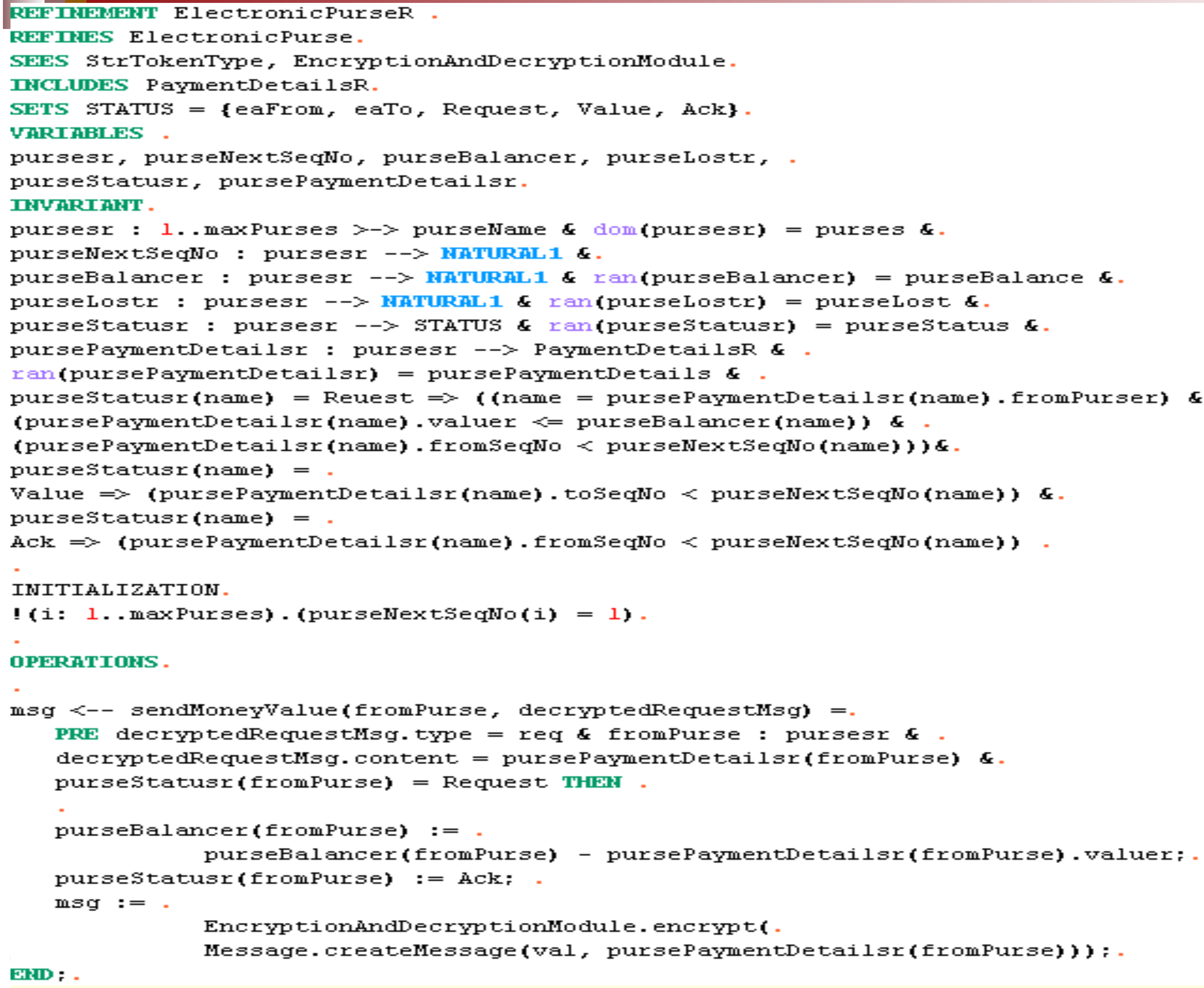
verifieded(reqMsg) ^ decrypted(reqMsg) ^ reqMsg.type = Request ^
fromPurse ∈ AuthenticPurses ^ reqMsg.Content = fromPurse.paymentDetails ^
fromPurse.Status = "Request"

DomPost deducted(fromPurse.paymentDetails.Balance) ^ (∃ cd: CommunicationDevice)Sending(msg, cd) ^ fromPurse.Status = “Ack”
ReqPreFor TransferValueMsgReceived
Sending(valueMsg, fromPurse.paymentDetails.toPurse)

KAOS Transformation to B (Riham Hassan 2008)



```
MACHINE ElectronicPurse (maxPurses)

CONSTRAINTS maxPurses : 1..100000

SEES StrTokenType, EncryptionAndDecryptionModule

INCLUDES PaymentDetails, Message

DEFINITIONS PURSE == 0..maxPurses - 1; MAX_LOG_SIZE = 1..15; EXCEPTION_LOG == MAX_LOG_SIZE +-> PaymentDetails

VARIABLES
purses, purseName, purseBalance, purseLost, purseStatus, pursePaymentDetails, purseExceptionLog

INVARIANT
purses <: PURSE & purseName : purses >-> STRTOKEN &
purseBalance : purses --> NATURAL1 &
purseLost : purses --> NATURAL1 &
pursePaymentDetails : purses --> PaymentDetails &
purseExceptionLog : purses --> EXCEPTION_LOG &
!(pd, i : (purseExceptionLog(name))(i)).((name : pd.fromPurse) or (name : pd.toPurse))

INITIALIZATION
purses =/ {} ||  purseName =/ {} || purseBalance /= {} || purseLost = {0} ||
pursePaymentDetails =/ {} || purseExceptionLog = {}

OPERATIONS

msg <-- sendMoneyValue(fromPurse, decryptedRequestMsg)=
   PRE decryptedRequestMsg.type = req & fromPurse : purses &
   decryptedRequestMsg.content = pursePaymentDetails(fromPurse) THEN

   purseBalance(fromPurse) : = purseBalance(fromPurse) - pursePaymentDetails(fromPurse).value ||
   msg := EncryptionAndDecryptionModule.encrypt(Message.createMessage(val, pursePaymentDetails(fromPurse)))
END;
```



```
REFINEMENT ElectronicPurseR .
REFINES ElectronicPurse.
SEES StrTokenType, EncryptionAndDecryptionModule.
INCLUDES PaymentDetailsR.
SETS STATUS = {eaFrom, eaTo, Request, Value, Ack}.
VARIABLES .
pursesr, purseNextSeqNo, purseBalancer, purseLostr, .
purseStatusr, pursePaymentDetailsr.
INVARIANT.
pursesr : 1..maxPurses >-> purseName & dom(pursesr) = purses &.
purseNextSeqNo : pursesr --> NATURAL1 &.
purseBalancer : pursesr --> NATURAL1 & ran(purseBalancer) = purseBalance &.
purseLostr : pursesr --> NATURAL1 & ran(purseLostr) = purseLost &.
purseStatusr : pursesr --> STATUS & ran(purseStatusr) = purseStatus &.
pursePaymentDetailsr : pursesr --> PaymentDetailsR & .
ran(pursePaymentDetailsr) = pursePaymentDetails & .
purseStatusr(name) = Reuest => ((name = pursePaymentDetailsr(name).fromPurser) &
(pursePaymentDetailsr(name).valuer <= purseBalancer(name)) & .
(pursePaymentDetailsr(name).fromSeqNo < purseNextSeqNo(name)))&.
purseStatusr(name) = .
Value => (pursePaymentDetailsr(name).toSeqNo < purseNextSeqNo(name)) &.
purseStatusr(name) = .
Ack => (pursePaymentDetailsr(name).fromSeqNo < purseNextSeqNo(name)) .
.
INITIALIZATION.
!(i: 1..maxPurses).(purseNextSeqNo(i) = 1).
.
OPERATIONS.
.
msg <-- sendMoneyValue(fromPurse, decryptedRequestMsg) =.
   PRE decryptedRequestMsg.type = req & fromPurse : pursesr & .
   decryptedRequestMsg.content = pursePaymentDetailsr(fromPurse) &.
   purseStatusr(fromPurse) = Request THEN .
   .
   purseBalancer(fromPurse) := .
               purseBalancer(fromPurse) - pursePaymentDetailsr(fromPurse).valuer;.
   purseStatusr(fromPurse) := Ack; .
   msg := .
               EncryptionAndDecryptionModule.encrypt(.
               Message.createMessage(val, pursePaymentDetailsr(fromPurse)));.
END;.
```

Homework and Milestone Reminders

- Read paper on Angel: "Capturing and Using Software Architecture Knowledge for Architecture-Based Software Development" by Babar et. al.
  - □ Be prepared to discuss and even lead the discussion
  - □ Write a brief summary (half page) of observations on the paper and turn it in (in class)
    - Title
    - Basic thesis/premise/problem
    - Basic approach to address the problem
    - Summary of results
    - Key things you got from the paper personally
    - Open questions
- Let's talk Thursday about capturing software assets