Massachusetts Institute of Technology

Integrating the new ISO 19450 standard Object-Process Methodology OPM with MATLAB-Simulink for Model-Based Systems Engineering

Dov Dori

Technion, Israel Institute of Technology

Massachusetts Institute of Technology

Nov. 17, 2015

MITsdm

ESM L

Enterprise Systems Modeling Laboratory

What will this talk be about?

- Introduction to Model-Based Systems Engineering and conceptual modeling
- Object-Process Methodology – OPM, the new ISO/PAS 19450
- Integrating MATLAB/SIMULAINK quantitative aspects into OPM:
  - Approaches
  - Performance
  - Evalaution

What is Model-Based Systems Engineering?

The use of a formal modeling language to

- Model
- Architect
- Design
- Communicate & Share
- Test, Validate & Verify
- Deploy & Maintain

Complex multidisciplinary systems

MBSE Methodology

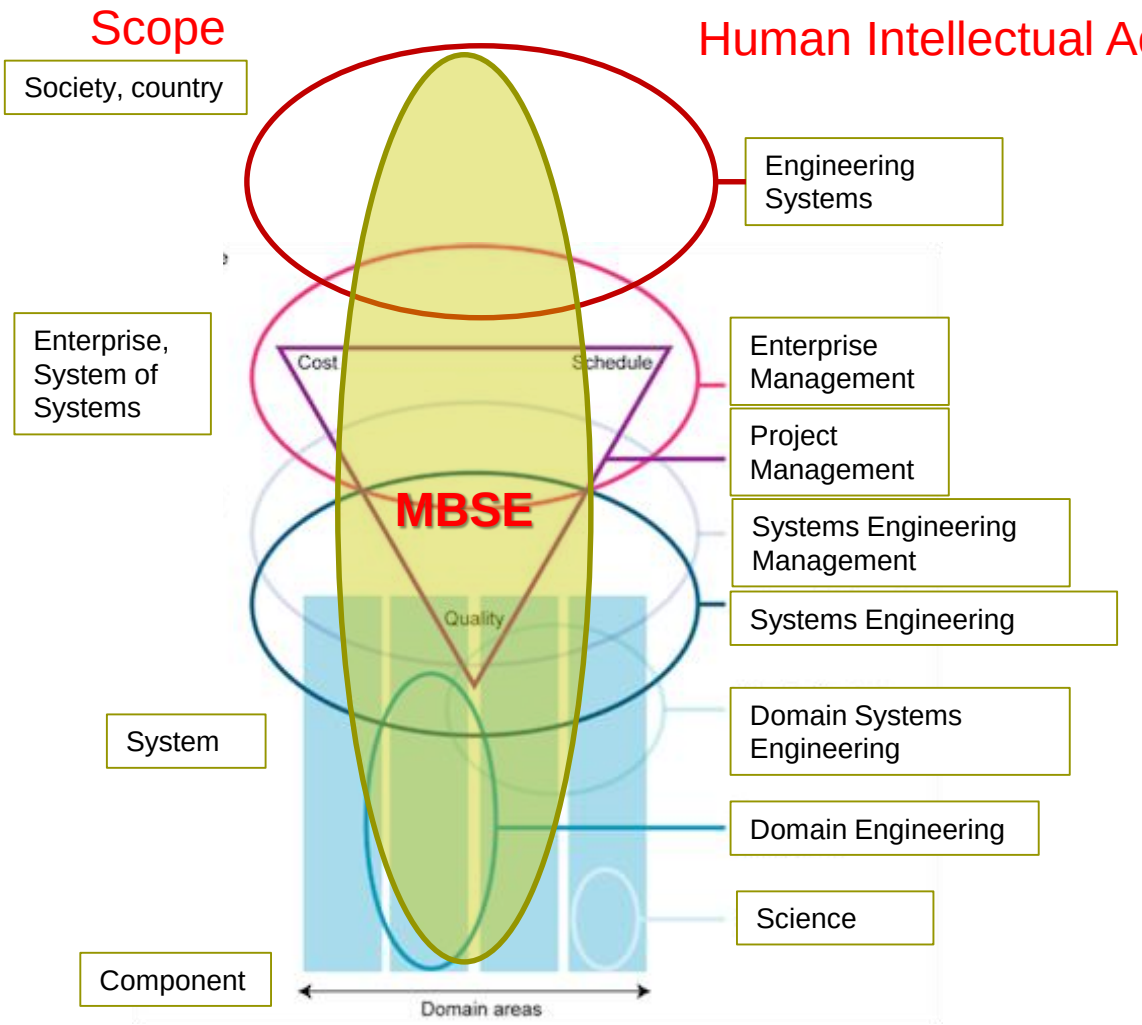
- MBSE calls for the development of a comprehensive **methodology**, capable of tackling the mounting challenges that the evolution of new systems and products poses.
- An MBSE methodology is a collection of related **processes, methods**, and **tools** that support systems engineering.
- **Modeling** is a foundational engineering activity in an MBSE methodology.
- The evolving model resulting from this activity is a central **infrastructural entity**
- **The model supports systems development, evolution, and lifecycle in a "model-based" or "model-driven" context.**

Conceptual Modeling

- Central to the MBSE approach is the activity of **conceptual modeling**:
  - the creation of a model or inter-related models or views in some **formal language**
  - The model specifies at **various** **levels of detail**, and from various viewpoints, how a system is **structured** and how it **behaves** in order for it to deliver its intended **function**.
- Let us examine an OPM model of a generic product lifecycle engineering system.

5

SE and MBSE and Engineering Systems in Context

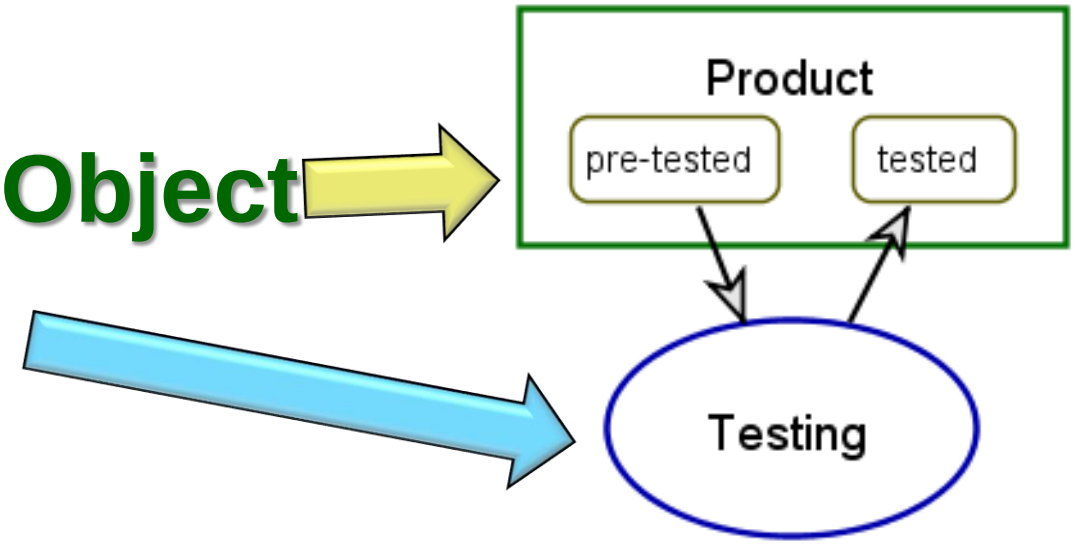


- **Conceptual modeling** and **MBSE** are **orthogonal**
- Not limited to any of the borders on the left.
- OPM-based conceptual modeling can be applied to systems in **any domain** and at **any level of complexity**.

6

OPM's only two building blocks:

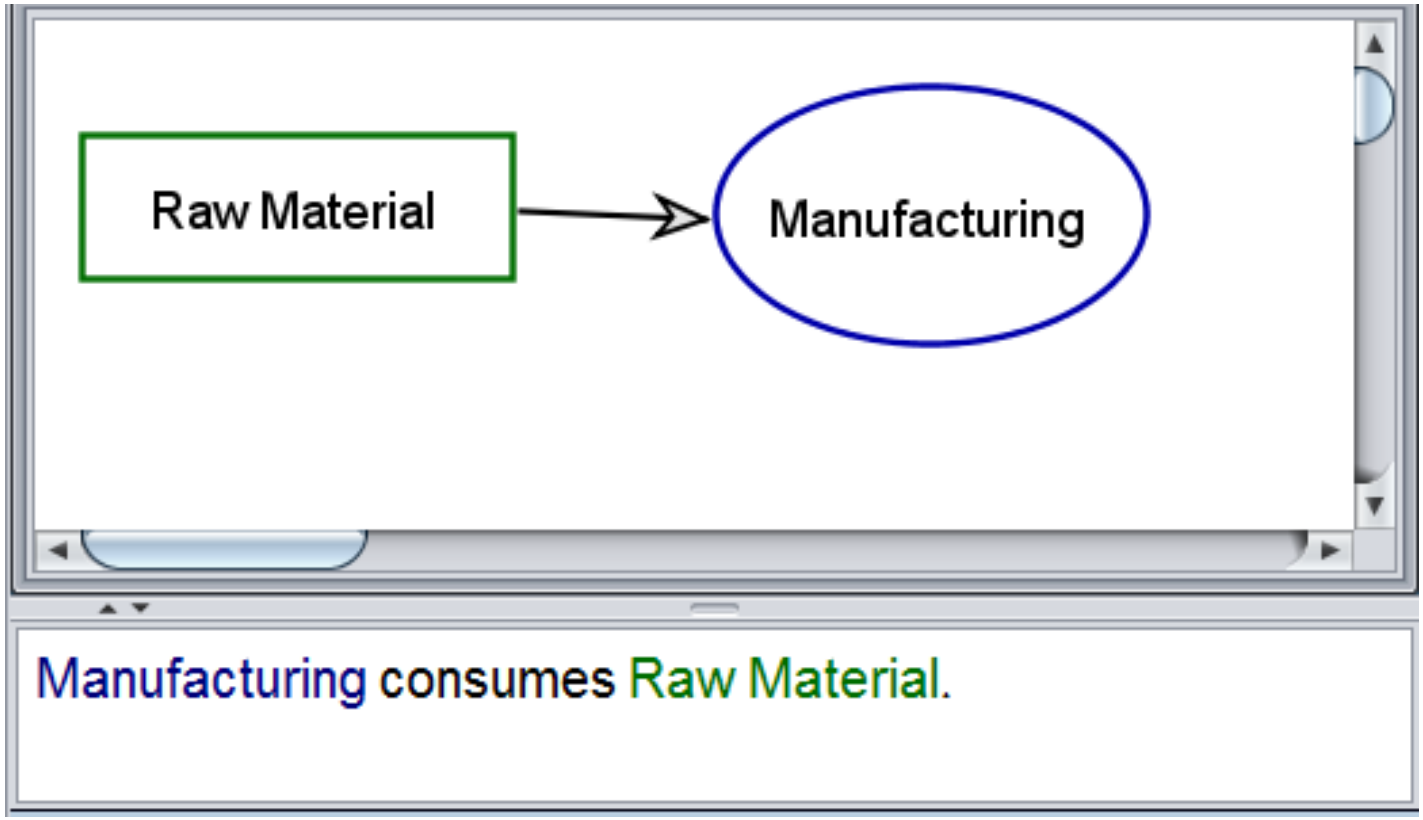
1. **Stateful Object**
2. **Process**



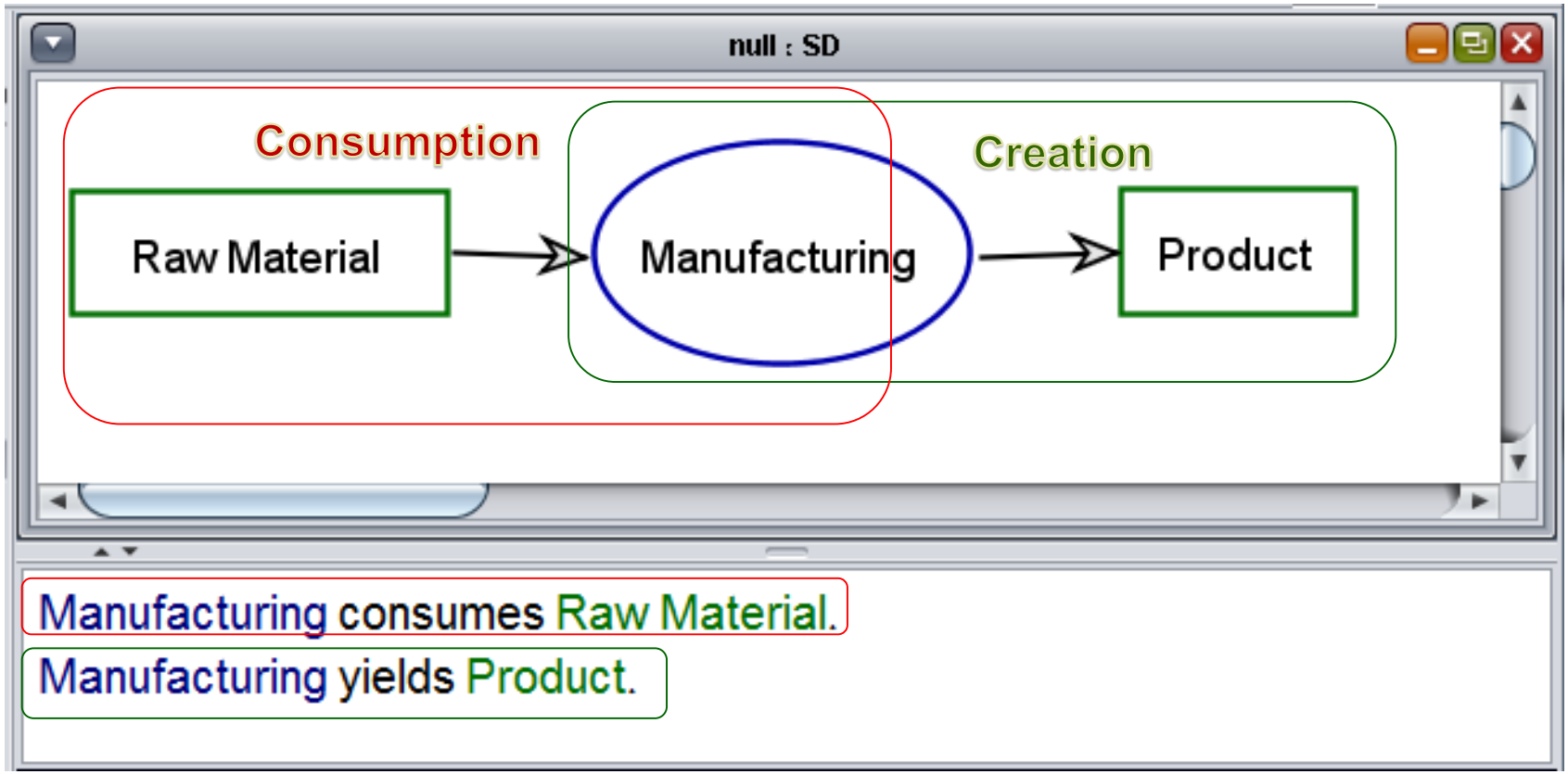
All the other elements are relations between things, expressed graphically as links.

Transforming an object by a process can be done in three ways

(1) Process consumes the object

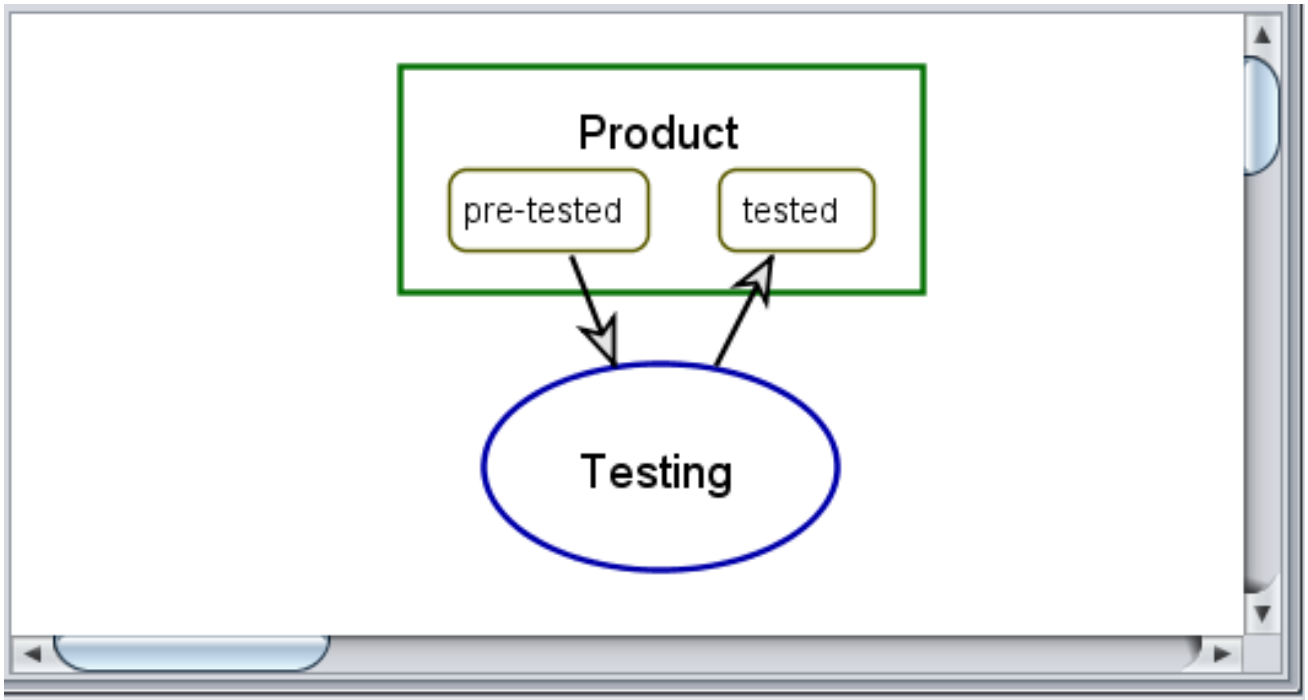


(2) Process creates the object



The third and last kind of object transformation:

(3) Process affects object by changing the object's state:

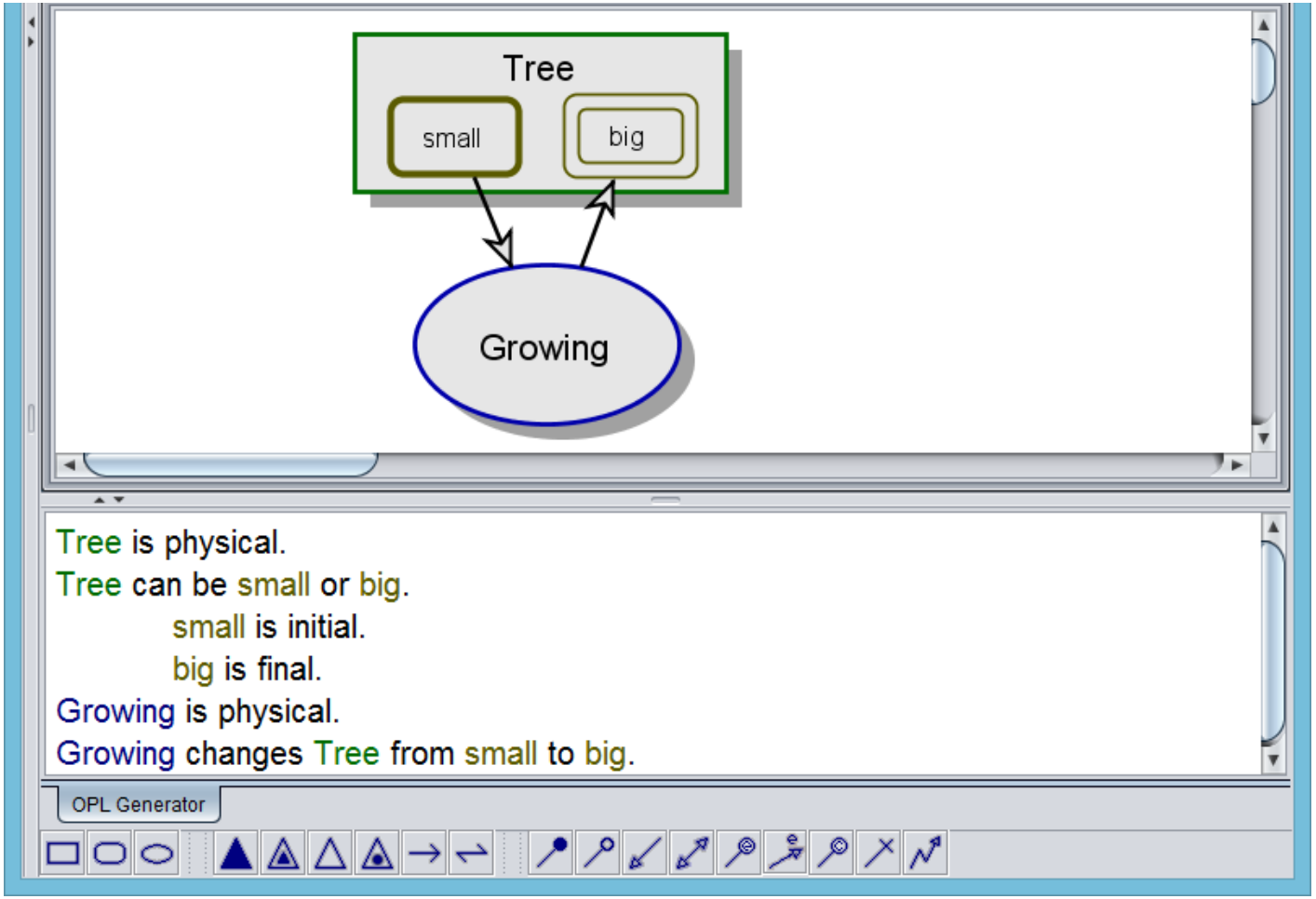


Product can be pre-tested or tested.
Testing changes Product from pre-tested to tested.

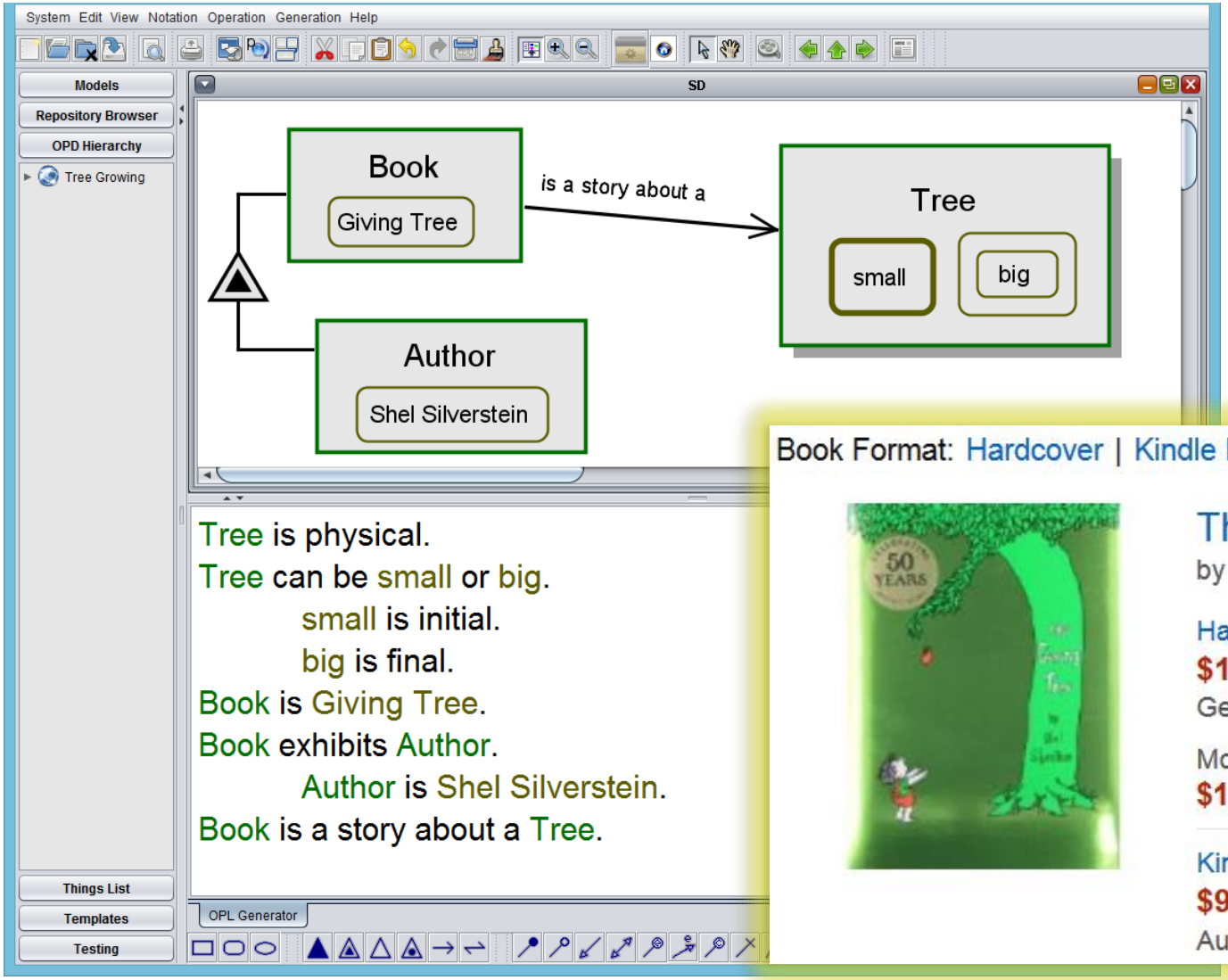
The graphics-text equivalence OPM principle

Any model fact expressed graphically in an OPD is also expressed textually in the corresponding OPL paragraph.

Caters to the dual channel cognitive assumption (Mayer, 2010)



Physical vs. Informatical Things



OPCAT – downloadable free from http://esml.iem.technion.ac.il/

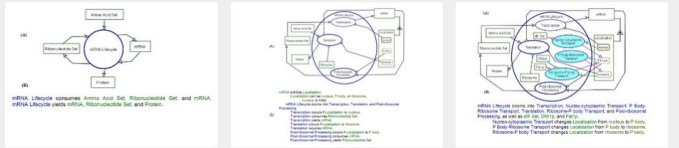
Application in Science: Molecular biology

Conceptual Modeling in Systems Biology Fosters Empirical Findings: The mRNA Lifecycle

Dov Dori, Mordechai Choder

Published: September 12, 2007 • DOI: 10.1371/journal.pone.0000872

Figures



Conceptual Model-Based Systems Biology: Mapping Knowledge and Discovering Gaps in the mRNA Transcription Cycle

Judith Somekh, Mordechai Choder, Dov Dori

Published: December 20, 2012 • DOI: 10.1371/journal.pone.0051430

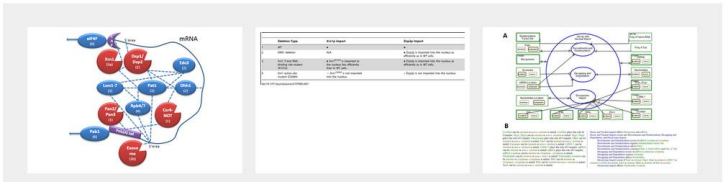
Figures

Conceptual Modeling of mRNA Decay Provokes New Hypotheses

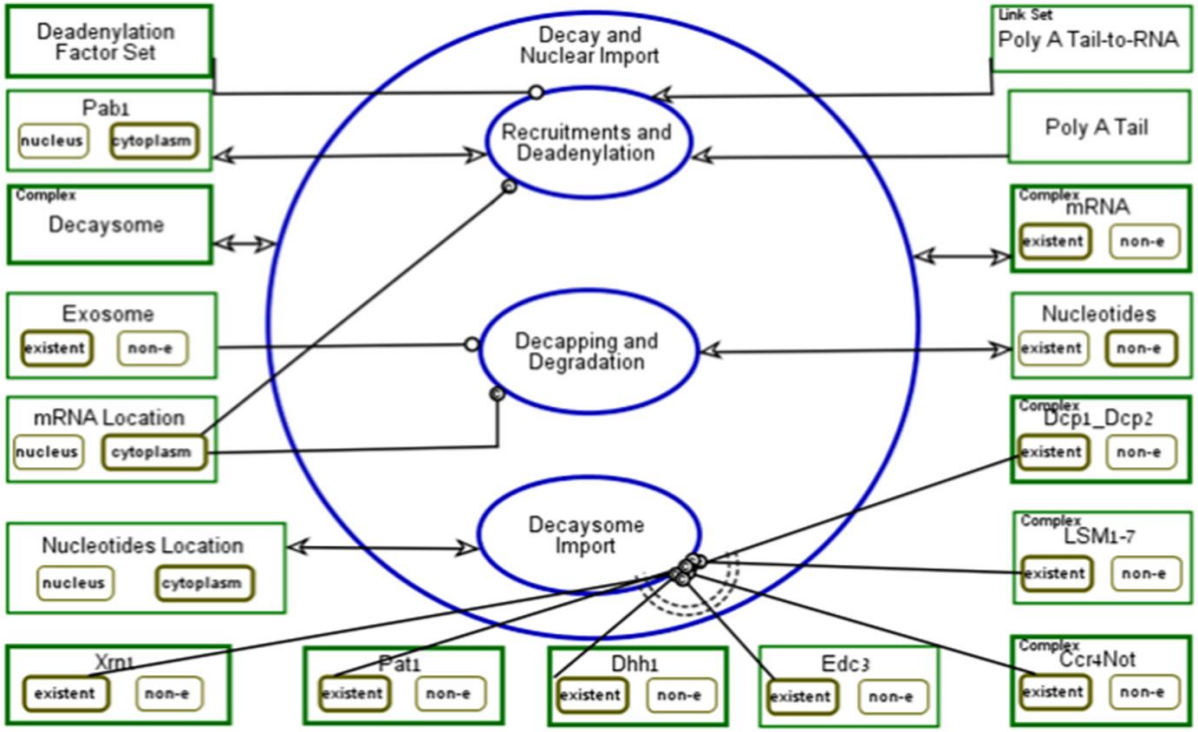
Judith Somekh, Gal Haimovich, Adi Guterman, Dov Dori, Mordechai Choder

Published: September 25, 2014 • DOI: 10.1371/journal.pone.0107085

Figures



A



B

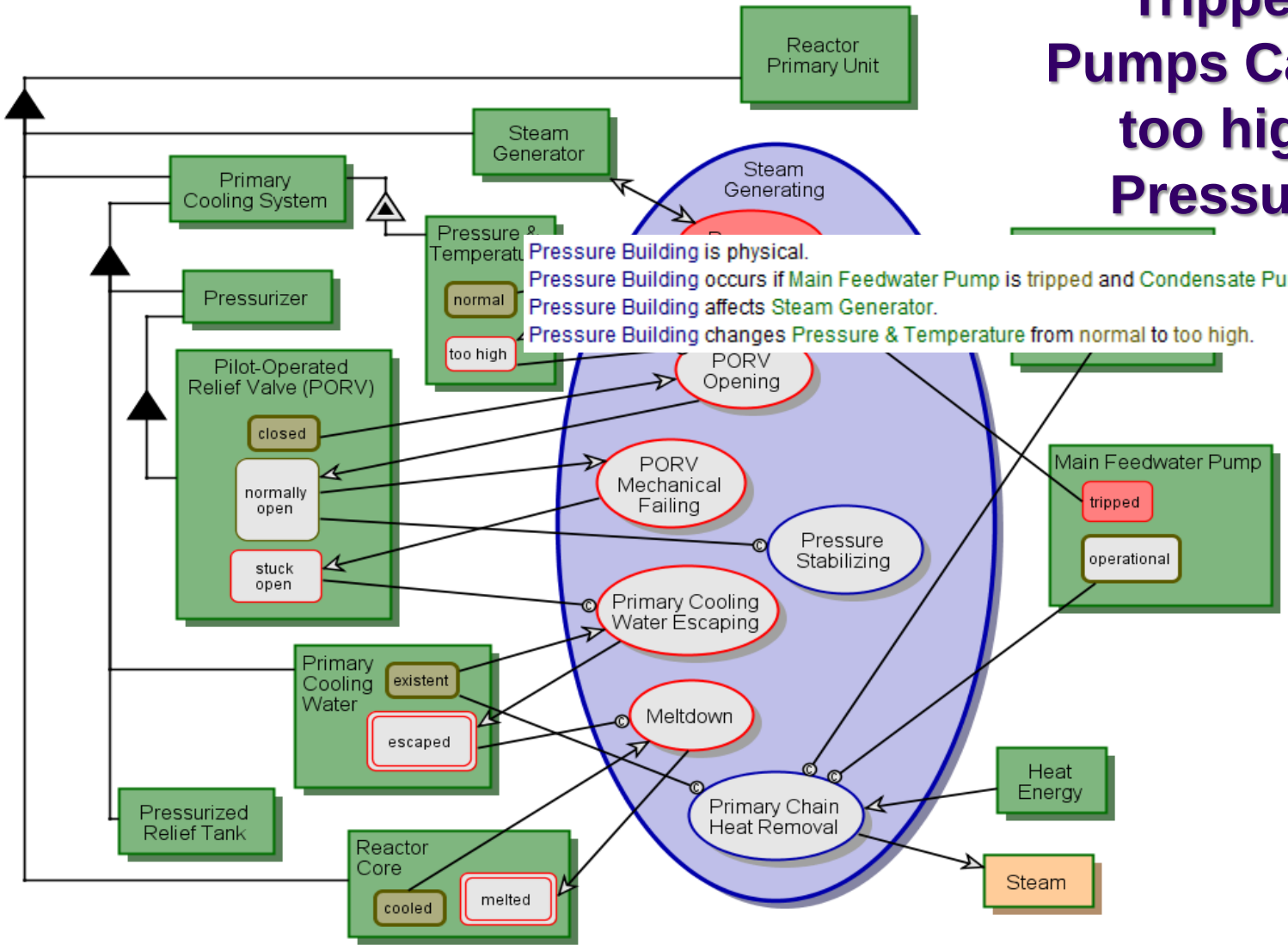
Ccr4Not can be existent or non-e. existent is initial. Ccr4Not plays the role of Complex. Dcp1_Dcp2 can be existent or non-e. existent is initial. Dcp1_Dcp2 plays the role of Complex. Decaysome plays the role of Complex. Dhh1 can be existent or non-e. existent is initial. Edc3 can be existent or non-e. existent is initial. Xrn1 can be existent or non-e. existent is initial. Pat1 can be existent or non-e. existent is initial. Lsm1-7 can be existent or non-e. existent is initial. mRNA can be existent or non-e. existent is initial. Nucleotides can be existent or non-e. existent is initial. mRNA Location can be nucleus or cytoplasm. Nucleotides Location can be nucleus or cytoplasm. Xrn1 can

Decay and Nuclear Import affects Decaysome and mRNA.
Decay and Nuclear Import zooms into Recruitments and Deadenylation, Decapping and Degradation, and Decaysome Import.
Recruitments and Deadenylation occurs if mRNA Location is cytoplasm.

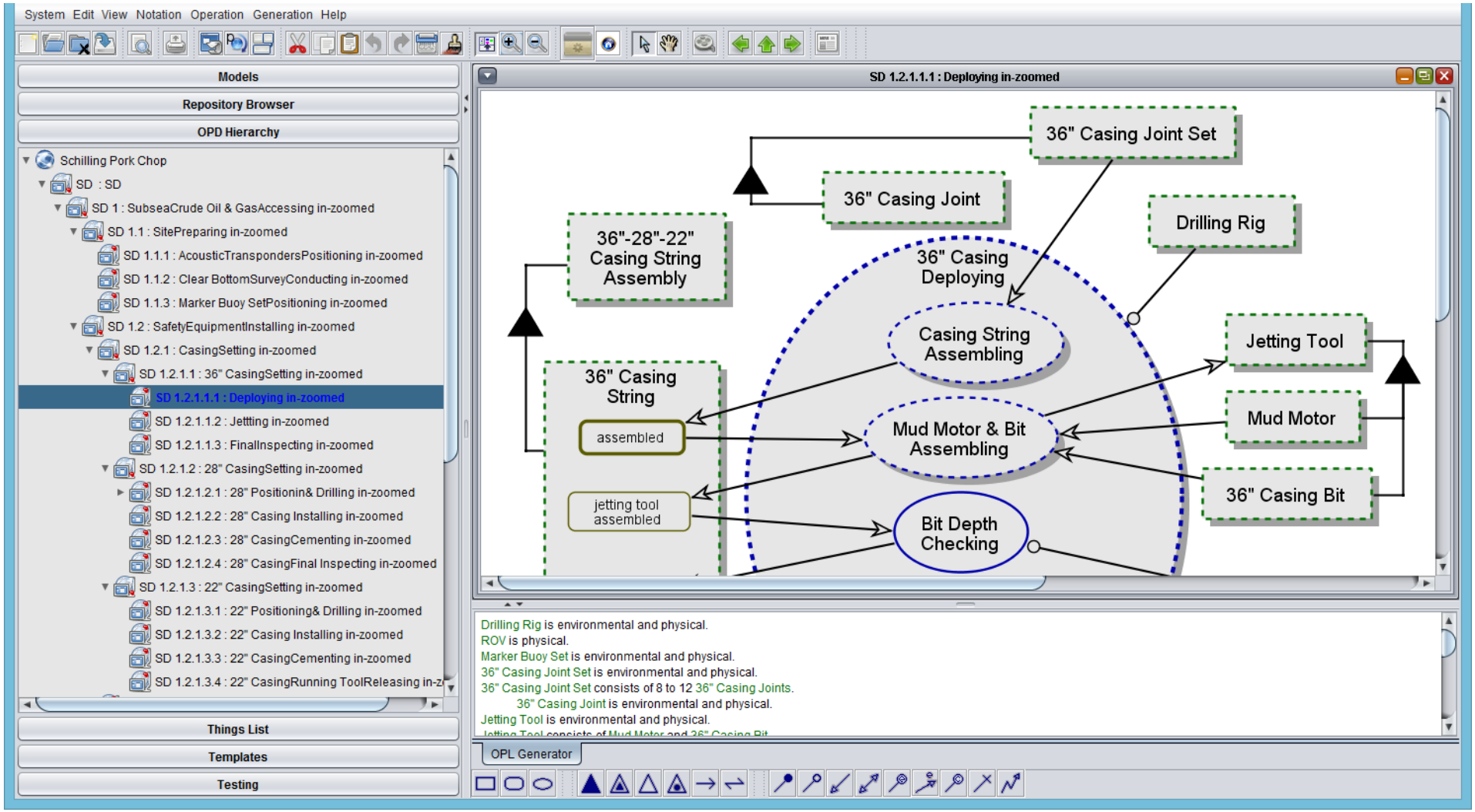
"Beyond the scientific value of these specific findings, this work demonstrates the value of the conceptual model as an in silico vehicle for hypotheses generation and testing, which can reinforce, and often even replace, risky, costlier wet lab experiments."

Nuclear reactor failure: The Three Mile Island Accident

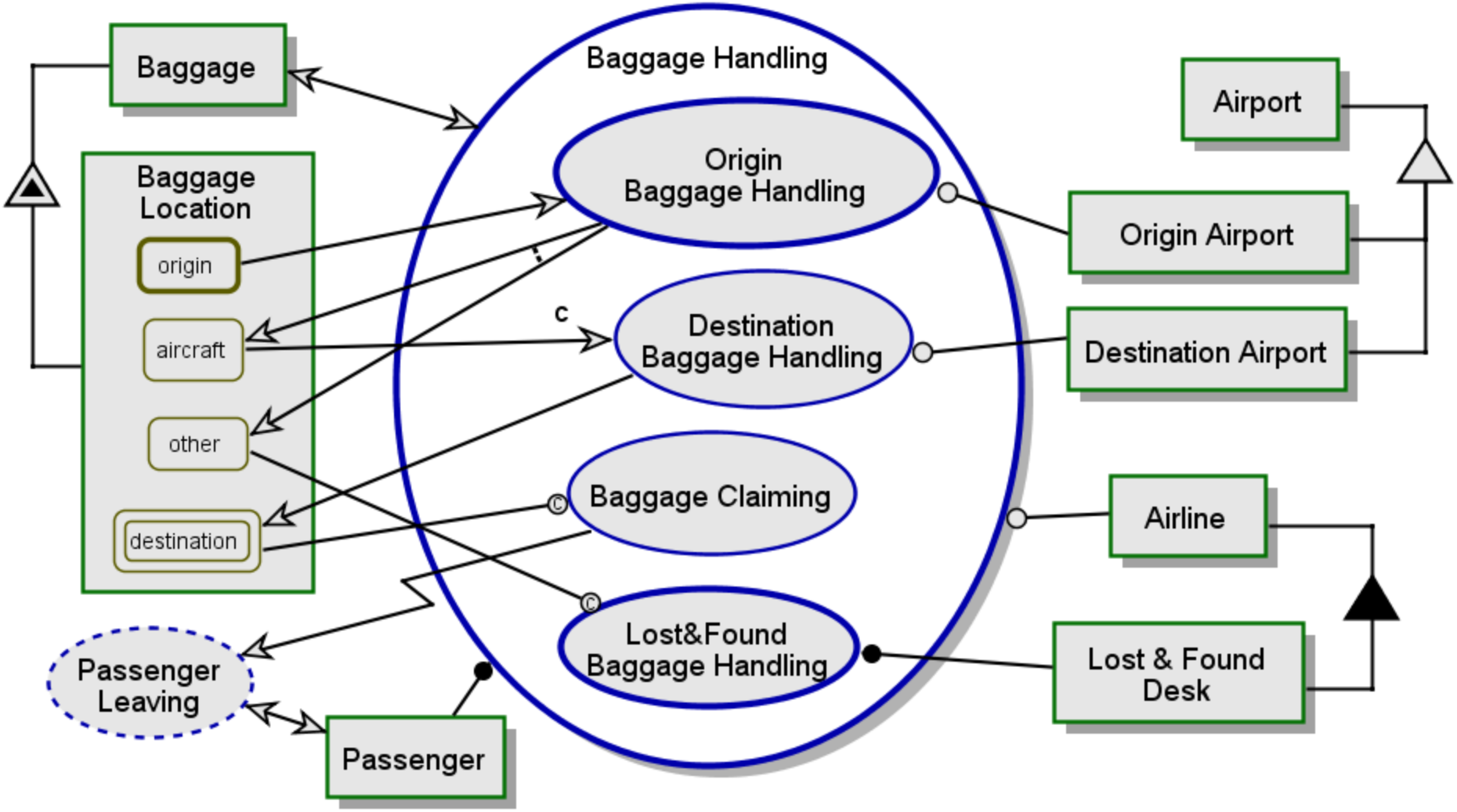
Tripped Pumps Cause too high Pressure



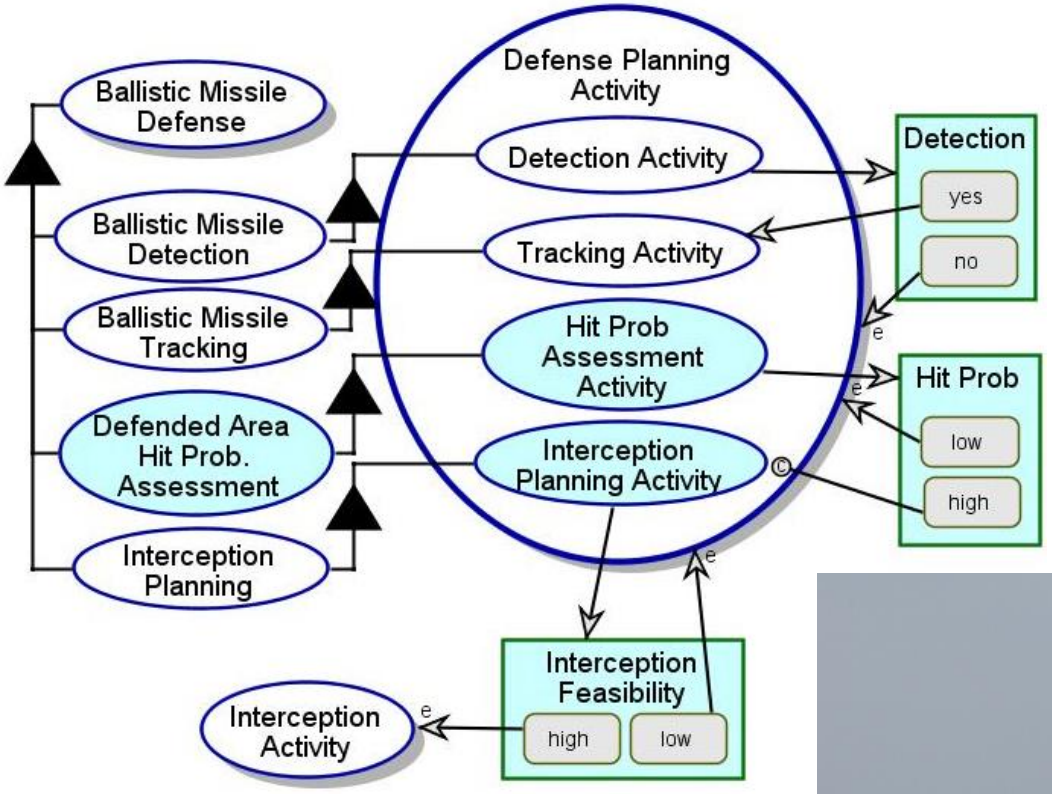
Offshore Oil Well Drilling



Airport Operations: Outgoing Passenger



Iron Dome – an Israeli ballistic missile defense system



Yaniv Mordecai and Dov Dori, Evolving System Modeling: Facilitating Agile System Development with Object-Process Methodology. *SysCon 2015 ,9th Annual IEEE International Systems Conference*, Vancouver, Canada, April 13-16 2015.
To be presented

Sample of engineering domains in which OPM has been used

- **Complex, Interconnected, Large-Scale Socio-Technical Systems.** *Systems Engineering* 14(3), 2011.
- **Networking Mobile Devices and Computers in an Intelligent Home.** *International Journal of Smart Home* 3(4), pp. 15-22, October, 2009.
- **Multi-Agent Systems.** *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 40 (2) pp. 227-241, 2010.
- **Semantic Web Services Matching and Composition.** *Web Semantics: Science, Services and Agents on the World Wide Web*. 9, pp. 16-28, 2011.
- **Project-Product Lifecycle Management.** *Systems Engineering*, 16 (4), pp. 413-426, 2013.
- **Model-Based Risk-Oriented Robust Systems Design.** *International Journal of Strategic Engineering Asset Management*, 1(4), pp. 331-354, 2013.
- **Medical Robotics and Miscommunication Scenarios.** An Object-Process Methodology Conceptual Model. *Artificial Intelligence in Medicine*, 62(3) pp. 153-163, 2014.
- **Modeling Exceptions in Biomedical Informatics**. *Journal of Biomedical Informatics 42*(4), pp. 736-747, 2009.

OPM is currently most fit for early, conceptual design

We want to leverage the strength of MATLAB/Simulink to enjoy the best of two worlds

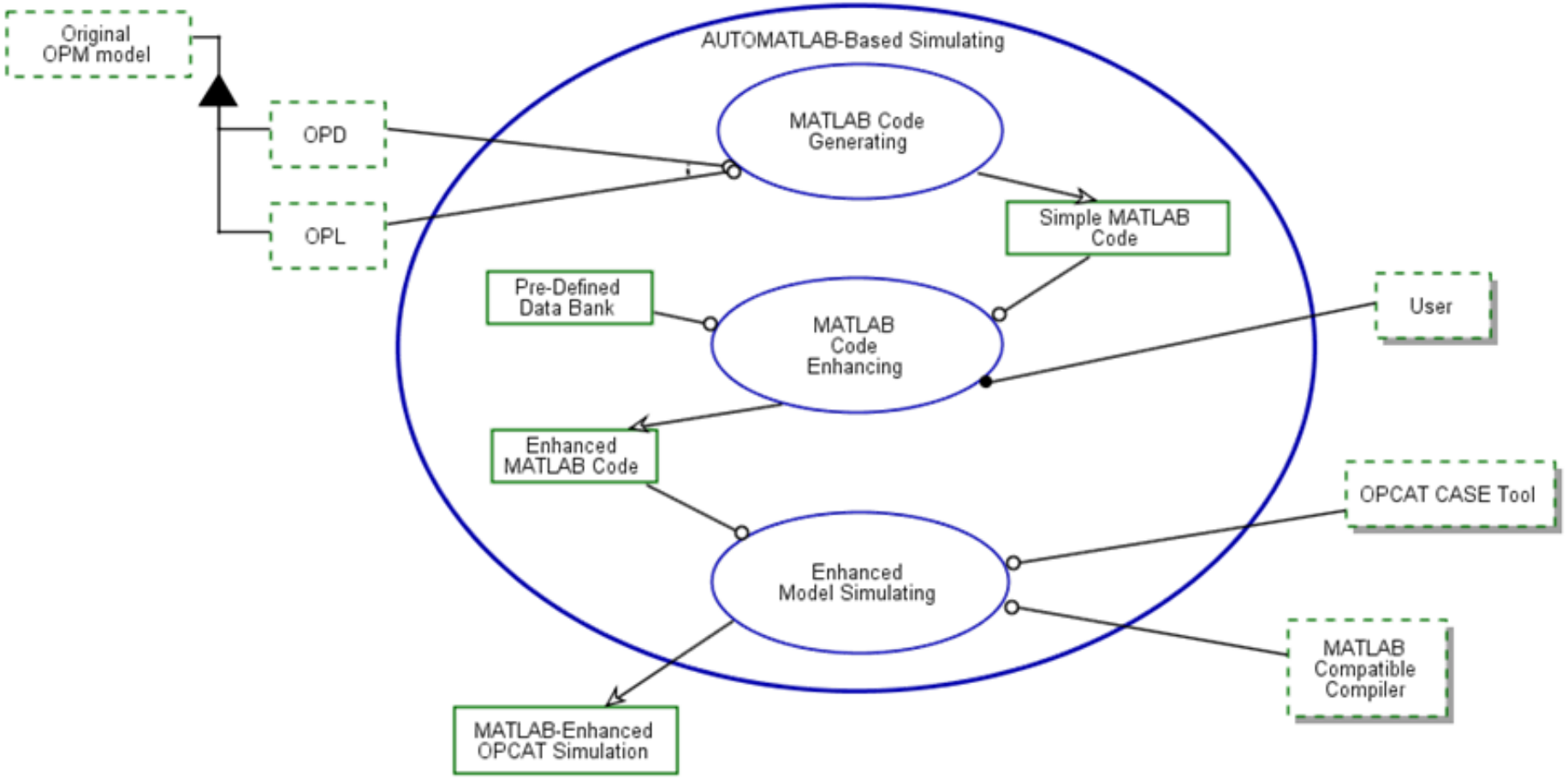
Two approaches (Aharon Renick's Masters Thesis):

1. **AUTOMATLAB**
2. **MMATLAB subcontractor**

AUTOMATLAB

- Adding a numerical computational layer to the conceptual modeling power of OPM.
- Simulating system behavior both qualitatively and quantitatively.
- AUTOMATLAB stages:
  - AUTOMATLAB code generation
  - AUTOMATLAB code enhancement
  - AUTOMATLAB controlled simulation

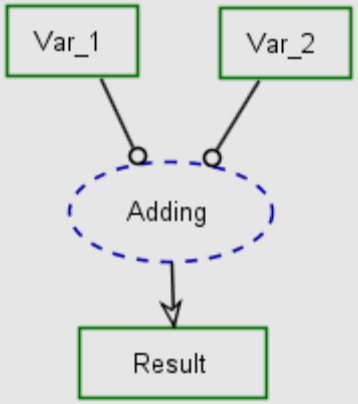
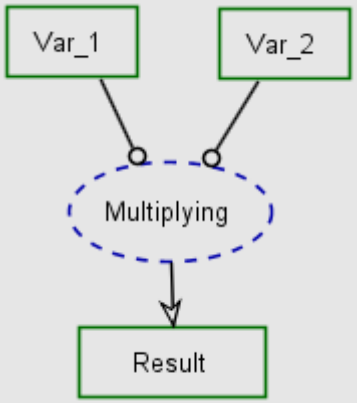
AUTOMATLAB Architecture



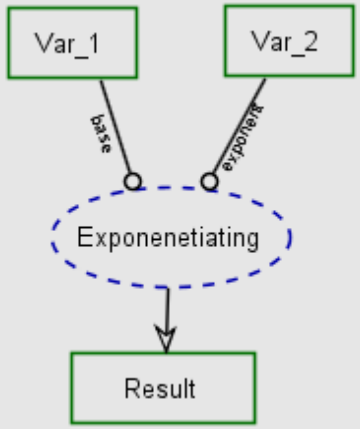
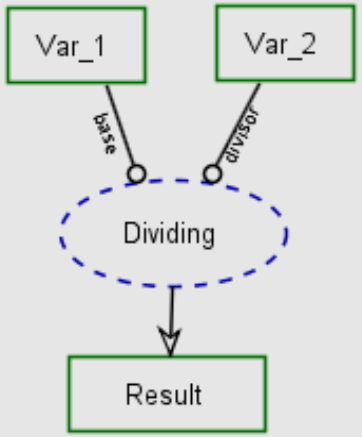
OPM-MATLAB equivalence

- Generating the AUTOMATLAB layer requires to understand the computational meaning of an OPM model.
- As a first step, we have mapped the main basic built-in MATLAB functions to their OPM model equivalents
- Some examples:

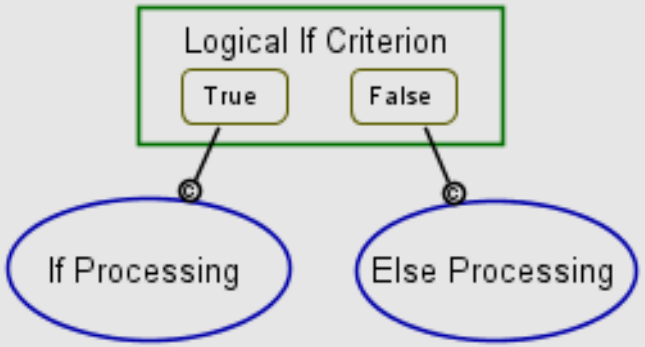
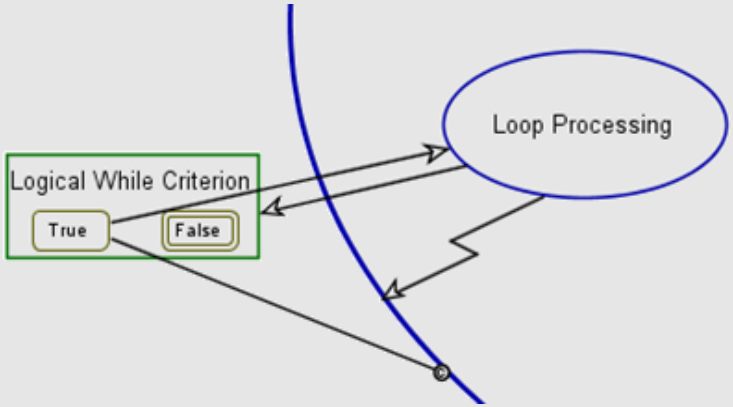
OPM-MATLAB equivalence example (1)

Symbol	Operator/Process Name	OPD
+	Addition / Adding	
*	Multiplication / Multiplying	

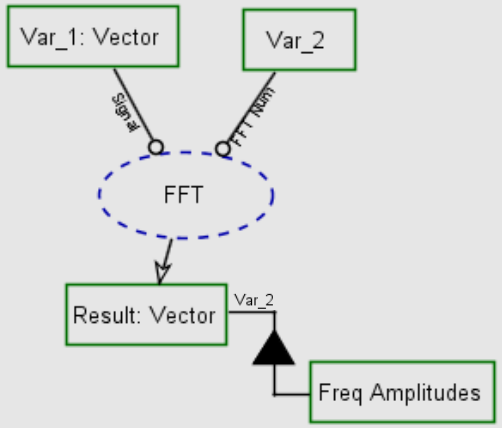
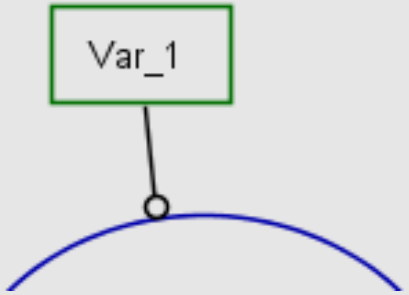
OPM-MATLAB equivalence example (2)

Symbol	Operator/Process Name	OPD
^	Exponentiation / Exponentiating	
\	Division / Dividing	

OPM-MATLAB equivalence example (3)

Operator / Process Name	OPD
if then…else	
while	

OPM-MATLAB equivalence example (4)

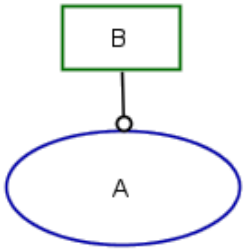
Operator / Process Name	Description	OPD
fft	Returns the discrete Fourier transform (DFT) of vector x, computed with a fast Fourier transform (FFT) algorithm	
isempty	Determine whether array/variable is empty (skips block if code if is empty)	

AUTOMATLAB Code Generator

- MATLAB code that is equivalent to the OPM model is generated.
- Processes, objects, values and relations are identified in the OPL statements and translated to MATLAB code.
- Processes and objects relations are mapped in three matrices: process-to-process relations, object-to-object relations, and process-to-object relations.

AUTOMATLAB Code Generator

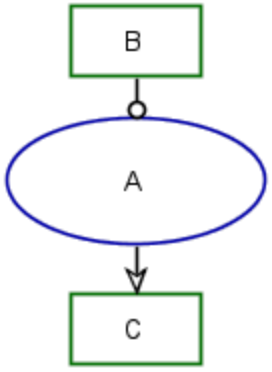
- For example, a ‘requires’ relation between process A and object B means that B is instrument for executing A.
- This will be translated to the following MATLAB code segment in the m file:




```
1      % A requires B.
2 -    if ~isempty(B)
3 -        A();
4 -    end
```

AUTOMATLAB Code Generator

- Adding a 'yields' relation between process A and object C, which means that C results from executing A results in the following code:




```
1     % A requires B.
2 -   if ~isempty(B)
3 -       [C] = A();
4 -   end
```

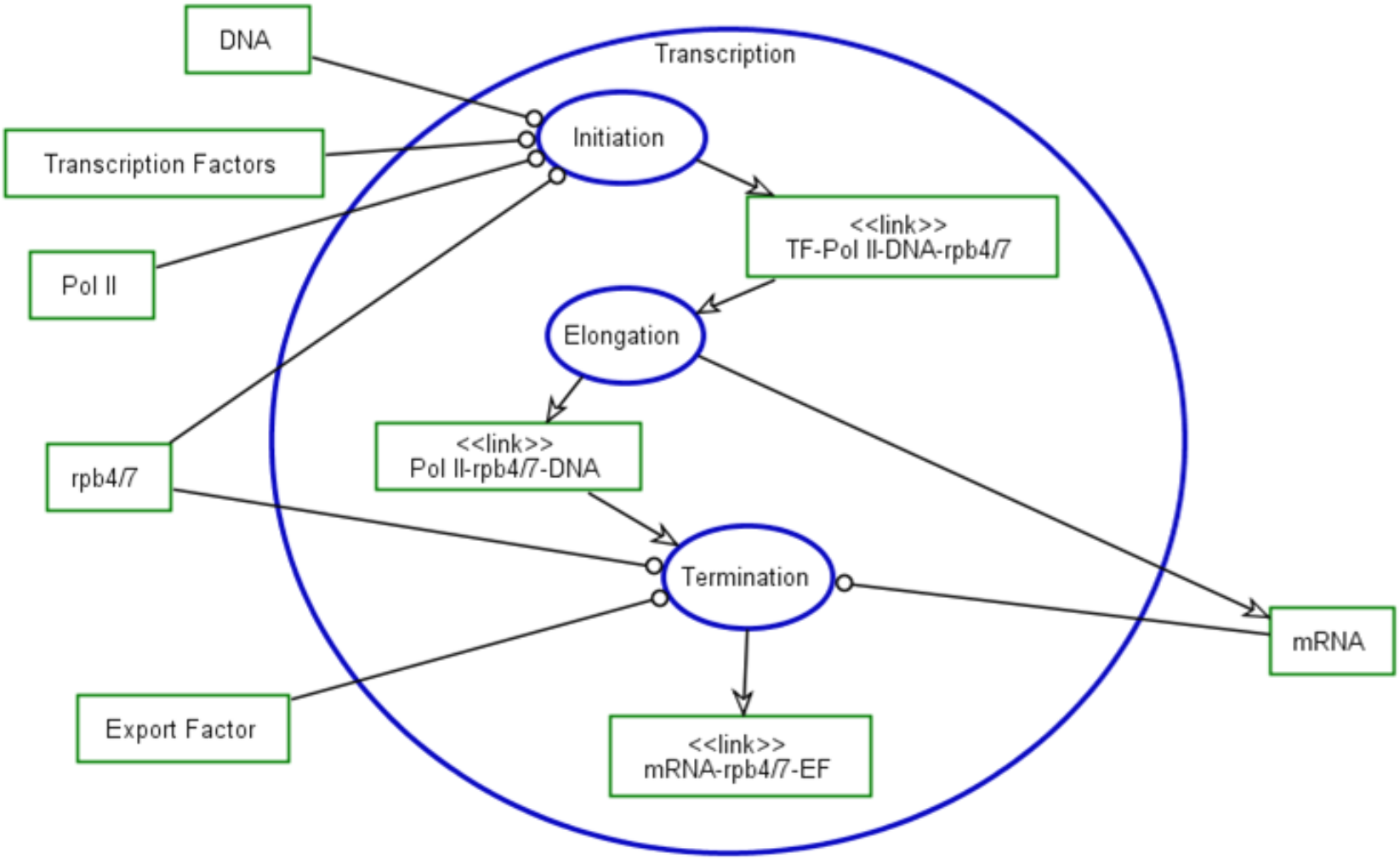
```
1     function [C] = A(B)
2     %A requires B
3     %A yields C
4
5     % B; %A requires B
6
7 -   C = 1; %A yields C
```

- A more complex example will be presented soon…

AUTOMATLAB Case Study

- Demonstration AUTOMATLAB for an OPM model of a molecular biology system.
- OPM model of a biological process called **mRNA Lifecycle** presented in (Somekh et al. 2012).

AUTOMATLAB Case Study



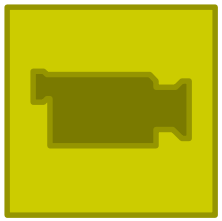
MATLAB code for mRNA Transcription

```
function [mRNA] = Transcription(TranscriptionFactors,rpb4_7,PolII,DNA,ExportFactor)
%Transcription requires TranscriptionFactors, Transcription requires rpb4_7, PolII, DNA and ExportFactor
%Transcription yields mRNA
%Transcription requires mRNA

%Initiation requires TranscriptionFactors, rpb4_7, PolII and DNA
%Initiation yields link__TF_PolII_DNA_rpb4_7
if ~isempty(TranscriptionFactors) && ~isempty(rpb4_7) && ~isempty(PolII) && ~isempty(DNA)
    [link__TF_PolII_DNA_rpb4_7] = Initiation(TranscriptionFactors,rpb4_7,PolII,DNA);
    %Transcription consists of Initiation
end

%Elongation consumes link__TF_PolII_DNA_rpb4_7
%Elongation yields link__PolII_rpb4_7_DNA and mRNA
if ~isempty(link__TF_PolII_DNA_rpb4_7)
    [link__PolII_rpb4_7_DNA,link__TF_PolII_DNA_rpb4_7,mRNA] = Elongation(link__TF_PolII_DNA_rpb4_7);
    %Transcription consists of Elongation
end

%Termination requires rpb4_7, mRNA and ExportFactor
%Termination consumes link__PolII_rpb4_7_DNA
%Termination yields link__mRNA_rpb4_7_EF
if ~isempty(rpb4_7) && ~isempty(link__PolII_rpb4_7_DNA) && ~isempty(mRNA) && ~isempty(ExportFactor)
    [link__PolII_rpb4_7_DNA,link__mRNA_rpb4_7_EF] = Termination(rpb4_7,link__PolII_rpb4_7_DNA,mRNA,ExportFactor);
    %Transcription consists of Termination
end

end
```

MATLAB code for mRNA Transcription

```
function [mRNA,InitiationLength] = Transcription(TranscriptionFactors,rpb4_7,PolII,DNA,ExportFactor)
%Transcription requires TranscriptionFactors, Transcription requires rpb4_7, PolII, DNA and ExportFactor
%Transcription yields mRNA
%Transcription requires mRNA

%Initiation requires TranscriptionFactors, rpb4_7, PolII and DNA
%Initiation yields link__TF_PolII_DNA_rpb4_7
InitiationLength = 0;
InitiationExecuted = 0;
while ~InitiationExecuted
    p = rand;
    if p<0.7
        if ~isempty(TranscriptionFactors) && ~isempty(rpb4_7) && ~isempty(PolII) && ~isempty(DNA)
            [link__TF_PolII_DNA_rpb4_7] = Initiation(TranscriptionFactors,rpb4_7,PolII,DNA);
            %Transcription consists of Initiation
        end
        InitiationExecuted = 1;
        InitiationLength = InitiationLength + 2 + rand*(3-2);
    else
        InitiationLength = InitiationLength + 0.5 + rand*(1-0.5);
    end
end
%Elongation consumes link__TF_PolII_DNA_rpb4_7
%Elongation yields link__PolII_rpb4_7_DNA and mRNA
if ~isempty(link__TF_PolII_DNA_rpb4_7)
    [link__PolII_rpb4_7_DNA,link__TF_PolII_DNA_rpb4_7,mRNA] = Elongation(link__TF_PolII_DNA_rpb4_7);
    %Transcription consists of Elongation
end
%Termination requires rpb4_7, mRNA and ExportFactor
%Termination consumes link__PolII_rpb4_7_DNA
%Termination yields link__mRNA_rpb4_7_EF
if ~isempty(rpb4_7) && ~isempty(link__PolII_rpb4_7_DNA) && ~isempty(mRNA) && ~isempty(ExportFactor)
    [link__PolII_rpb4_7_DNA,link__mRNA_rpb4_7_EF] = Termination(rpb4_7,link__PolII_rpb4_7_DNA,mRNA,ExportFactor);
    %Transcription consists of Termination
end
end
```

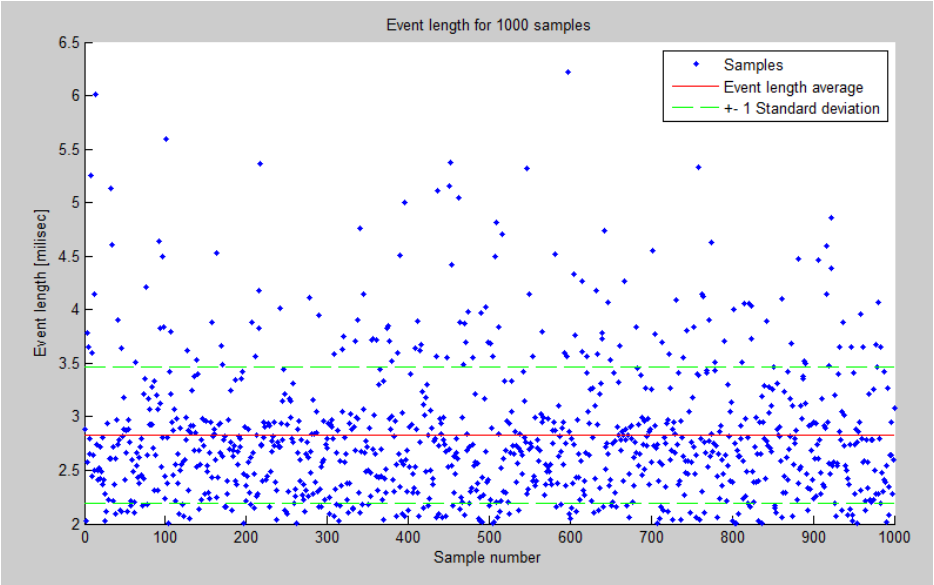
AUTOMATLAB Example



Matlab2OPM_1.wmv

AUTOMATLAB Case Study

- AUTOMATLAB controlled simulation includes additional stochastic capabilities.
- Simulation in MATLAB provides additional information and examining abilities.



OPM Computational Subcontractor

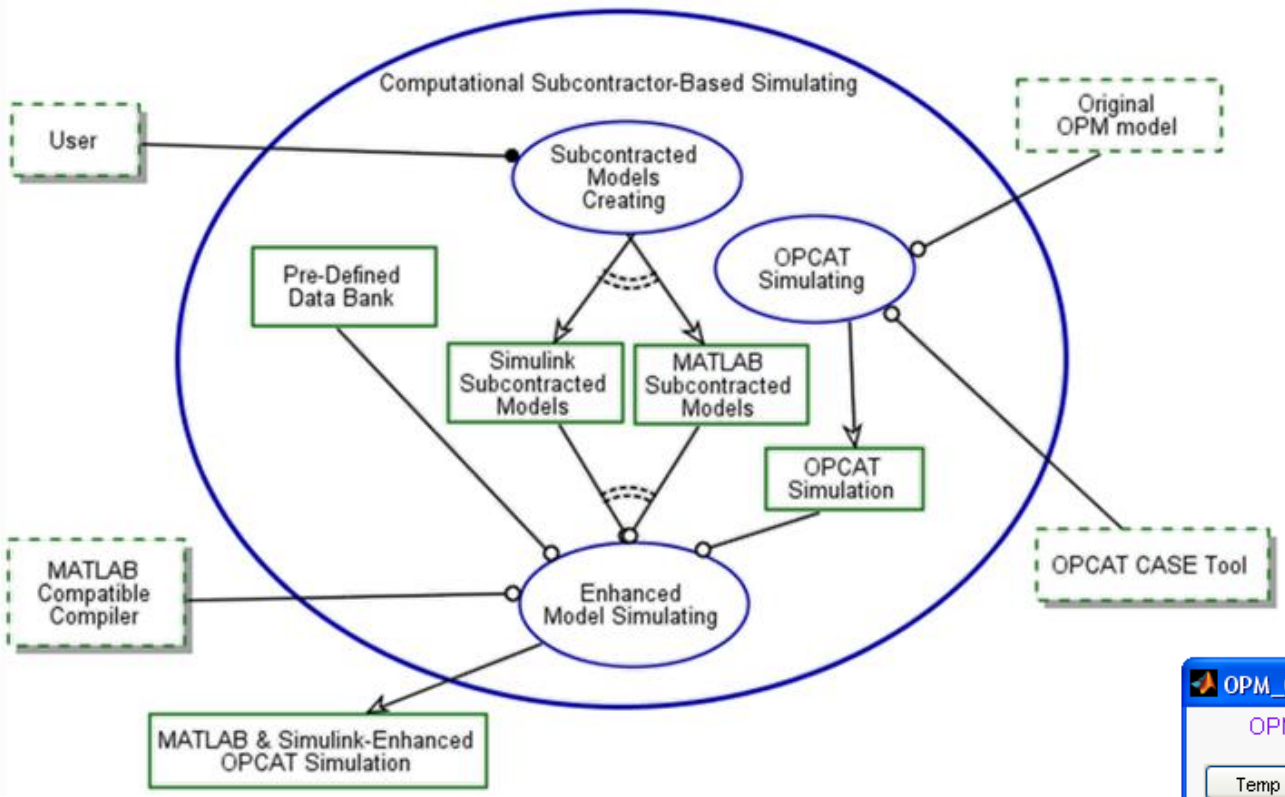
OPM Computational Subcontractor

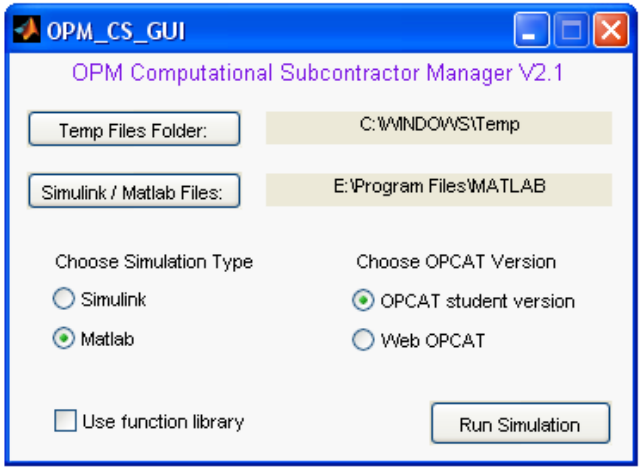
- MATLAB or Simulink is a "computational subcontractor" for the OPM model.
- We augmenting a regular OPM model, such that any process can be in-zoomed by MATLAB code or a Simulink diagram.
- When the OPM simulation is executed, it runs normally according to the OPM semantics.

OPM Computational Subcontractor

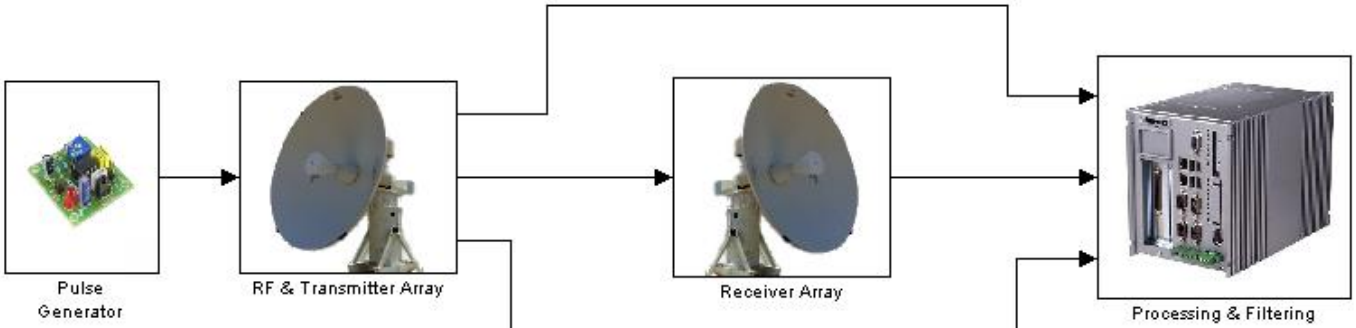
- When reaching a process that was in-zoomed by the computational subcontractor, MATLAB or Simulink are called.
- Relevant information is sent via MATLAB and the sub-simulation function is called.
- The outcome of this sub-simulation defines the outcome OPCAT simulation.

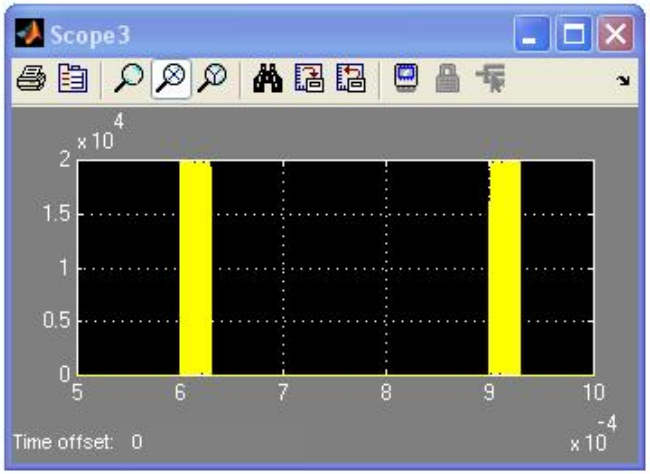
OPM Computational Subcontractor Architecture

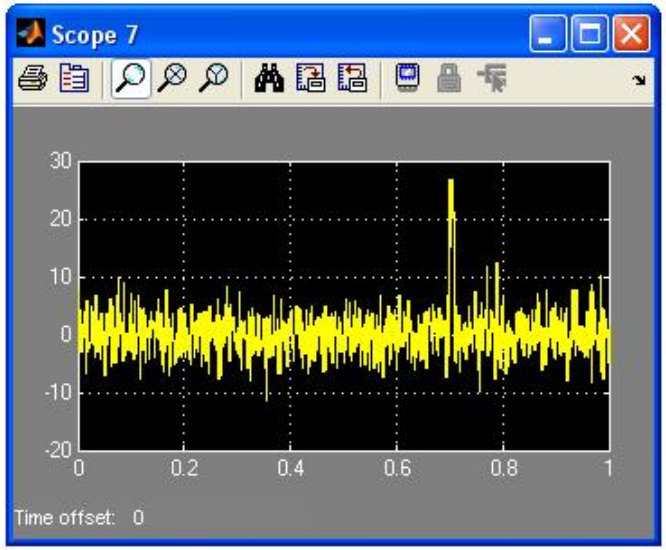




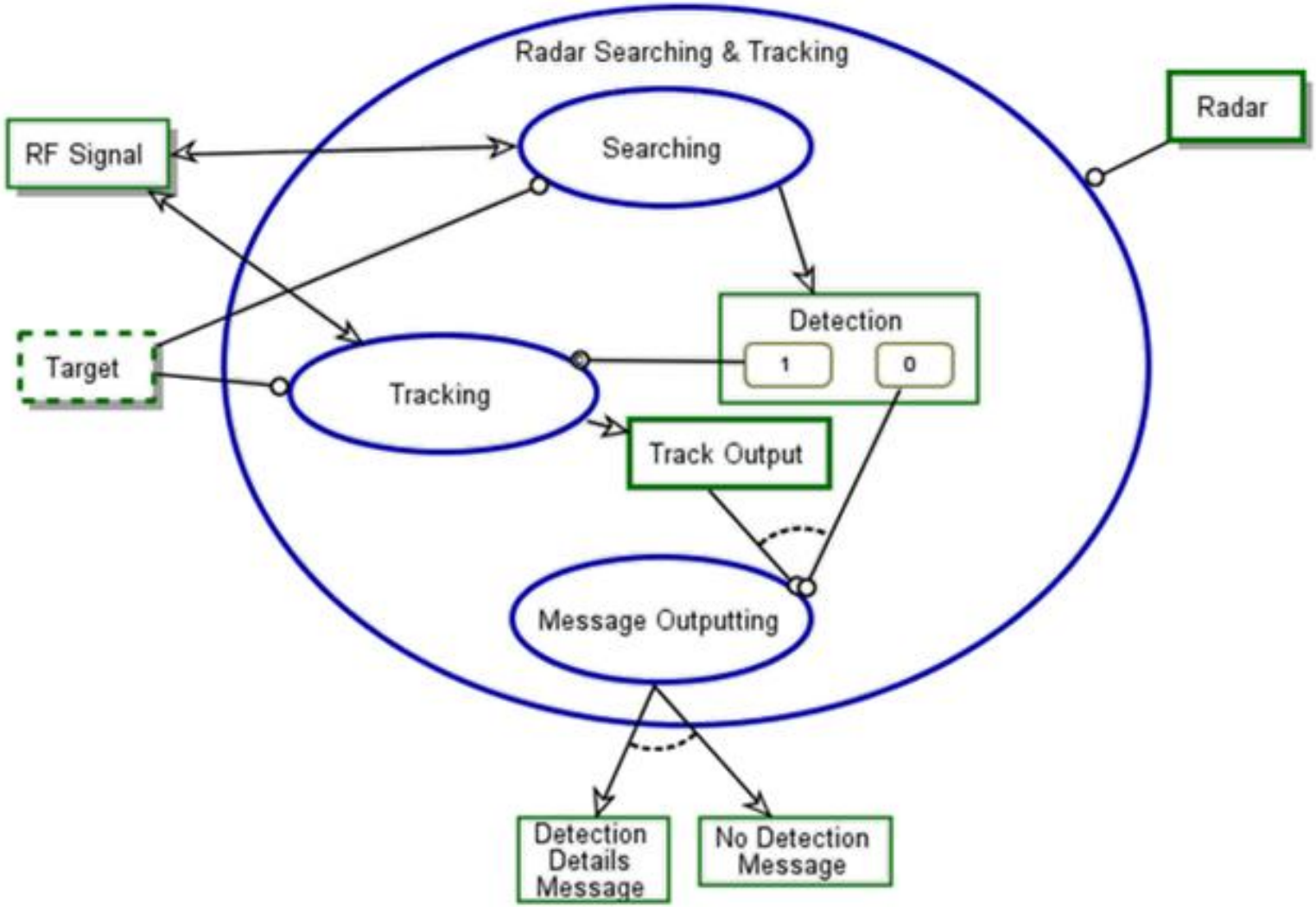
Example: OPM Computational Subcontractor for a search and tracking radar system



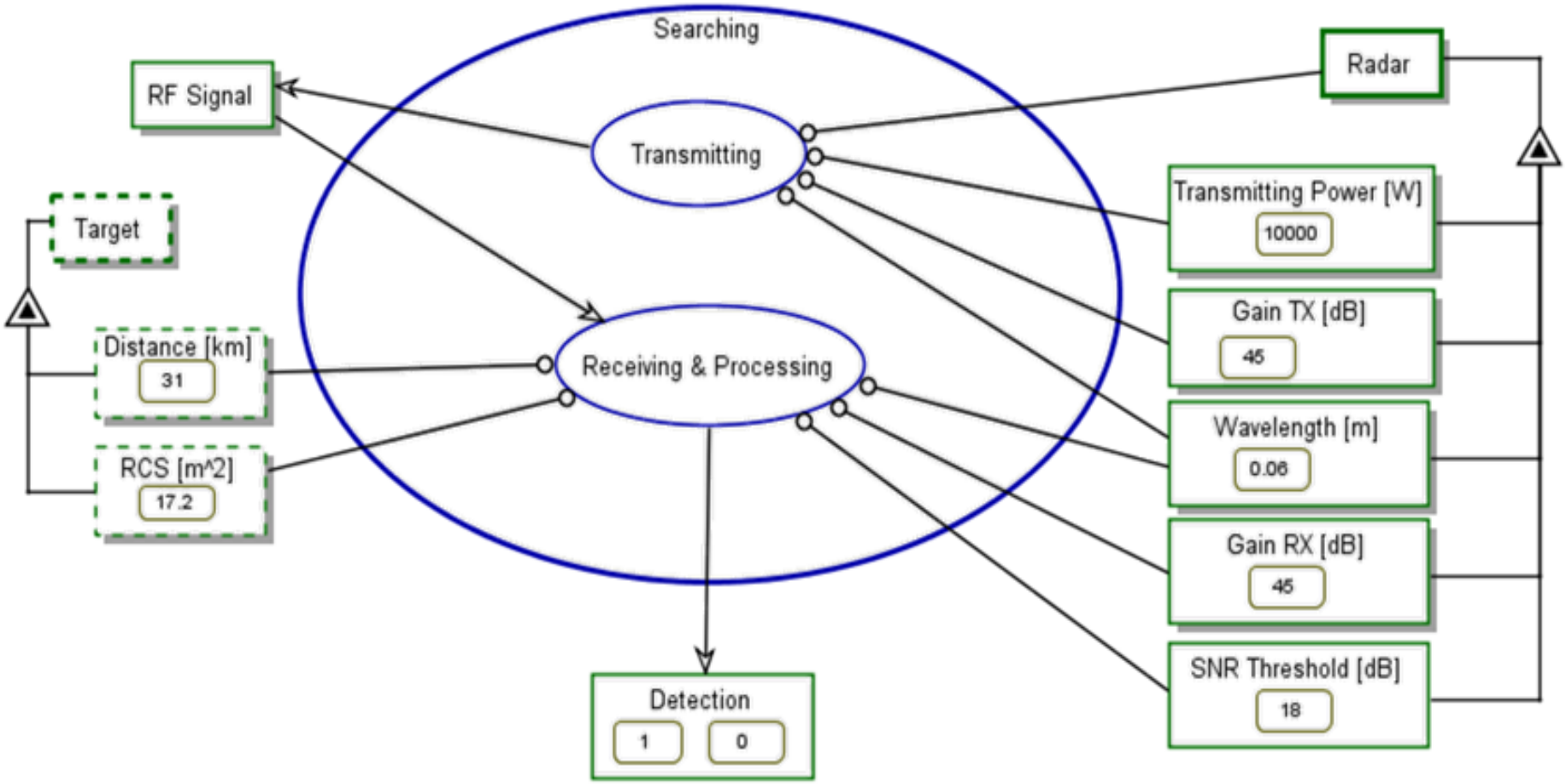




Radar Searching & Tracking in-zoomed



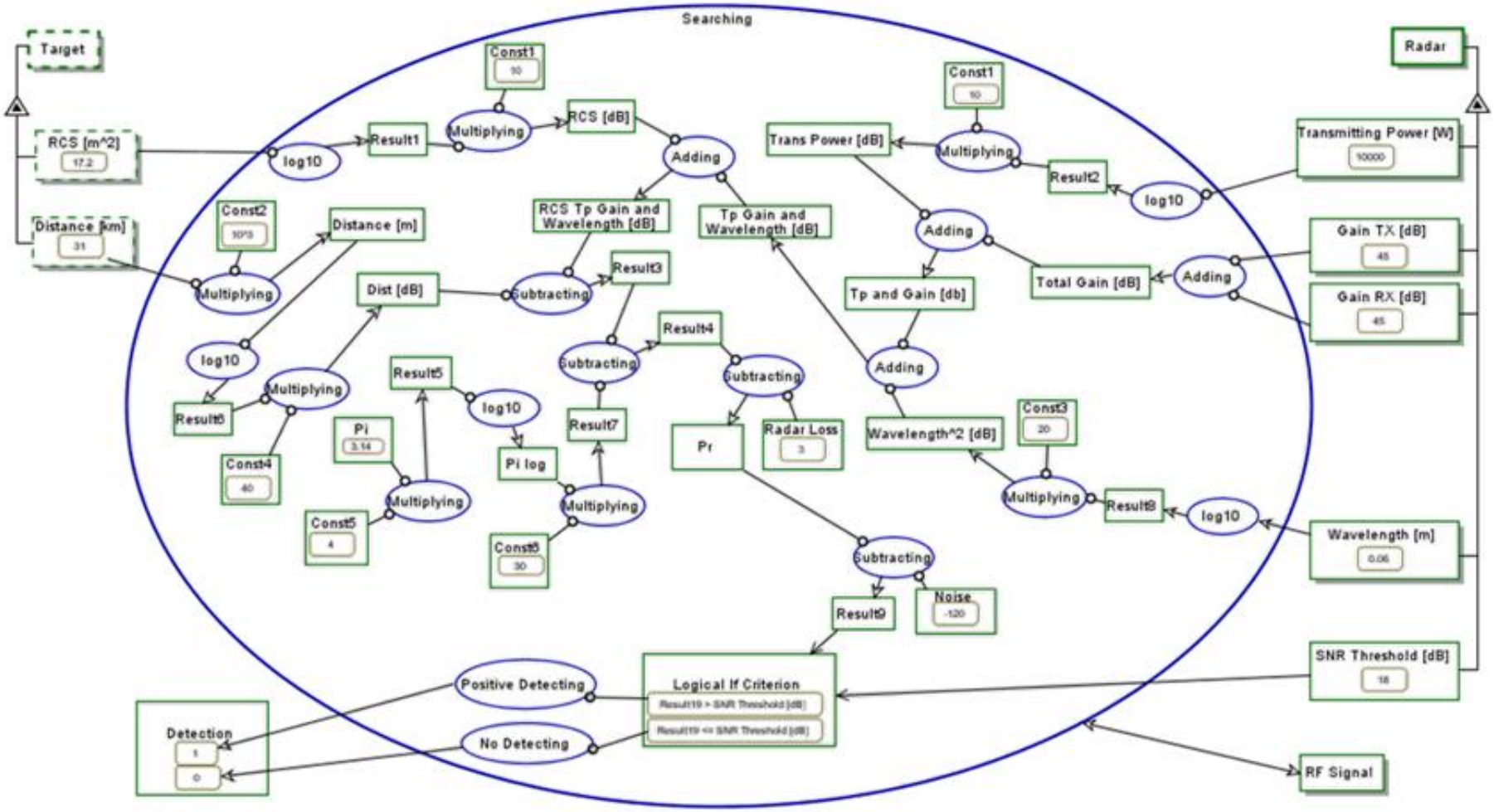
Searching in-zoomed



Math expressions are so much more compact…

$$P_r = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 R^4 L}$$

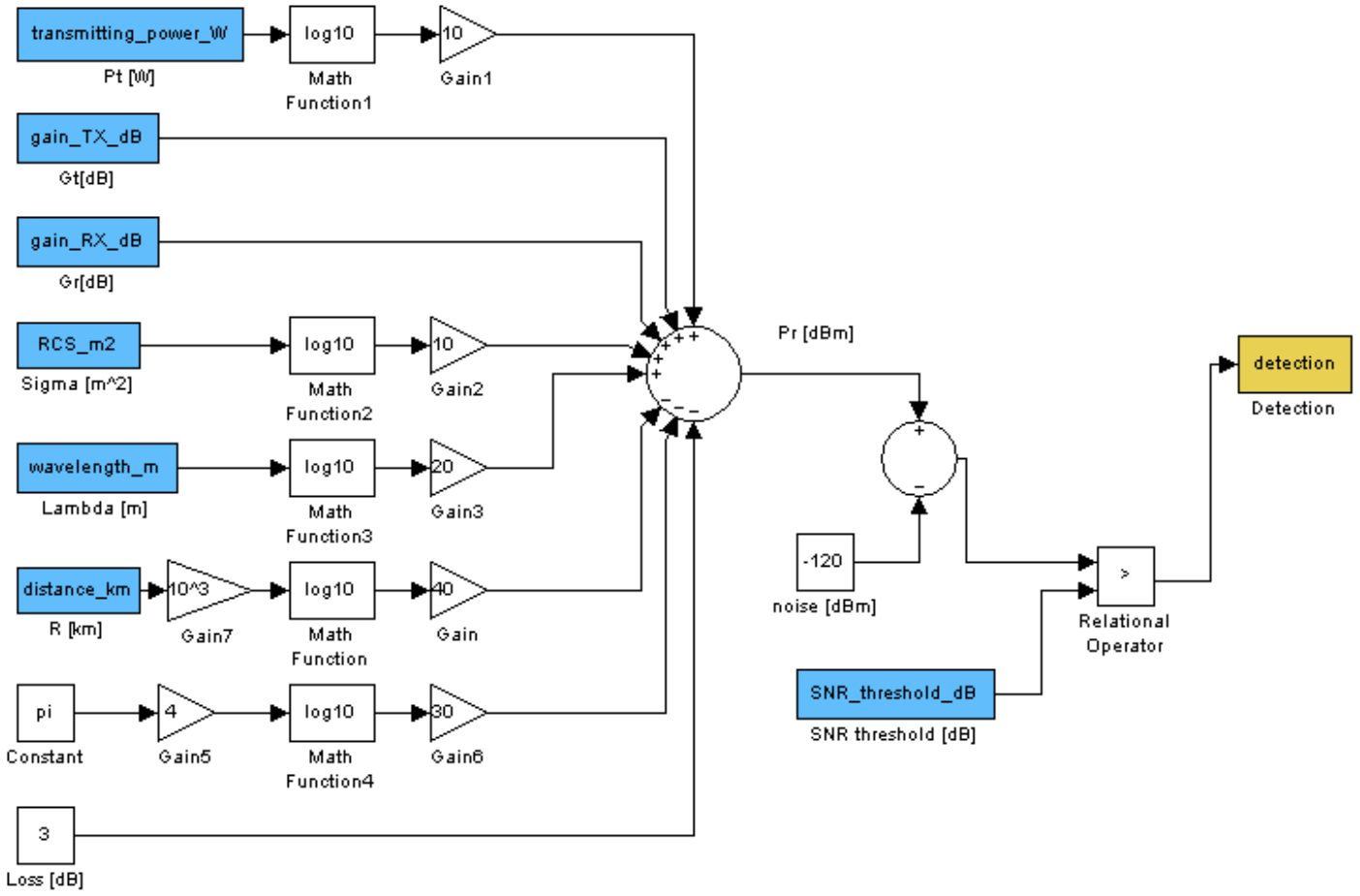
$$P_r = P_t + G_t + G_r + \sigma + 2\lambda - 3 \cdot 4\pi - 4R - L$$



Even Simulink is not as good as math…

$$P_r = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 R^4 L}$$

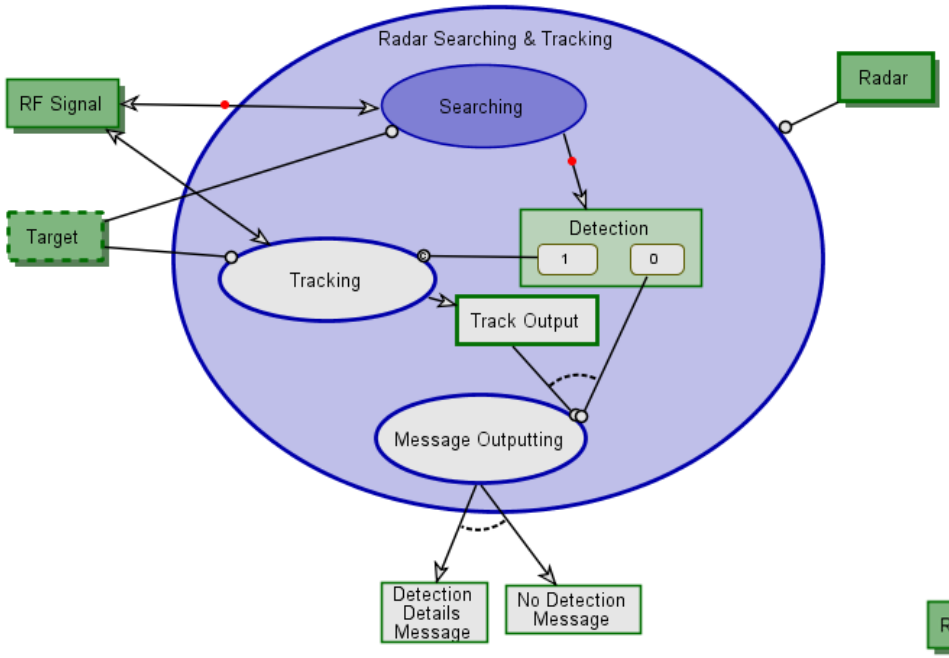
$$P_r = P_t + G_t + G_r + \sigma + 2\lambda - 3 \cdot 4\pi - 4R - L$$



The Simulink code for $P_r = P_t + G_t + G_r + \sigma + 2\lambda - 3 \cdot 4\pi - 4R - L$

```
function detection = Searching(transmitting_power_W,gain_TX_dB,gain_RX_dB,...
                               RCS_m2,wavelength_m,distance_km,SNR_threshold_dB)

Loss = 3; %[dB];
noise = -120; %[dBm]

% Pr [dBm]
Pr = ...
10*log10(transmitting_power_W) + gain_TX_dB + gain_RX_dB...
+ 10*log10(RCS_m2) + 20*log10(wavelength_m)...
- 10^3*40*log10(distance_km) - 4*30*log10(pi) - Loss;

if (Pr - noise) > SNR_threshold_dB
    detection = 1;
else
    detection = 0;
end
```

Executing the OPM model with calls to MATLAB

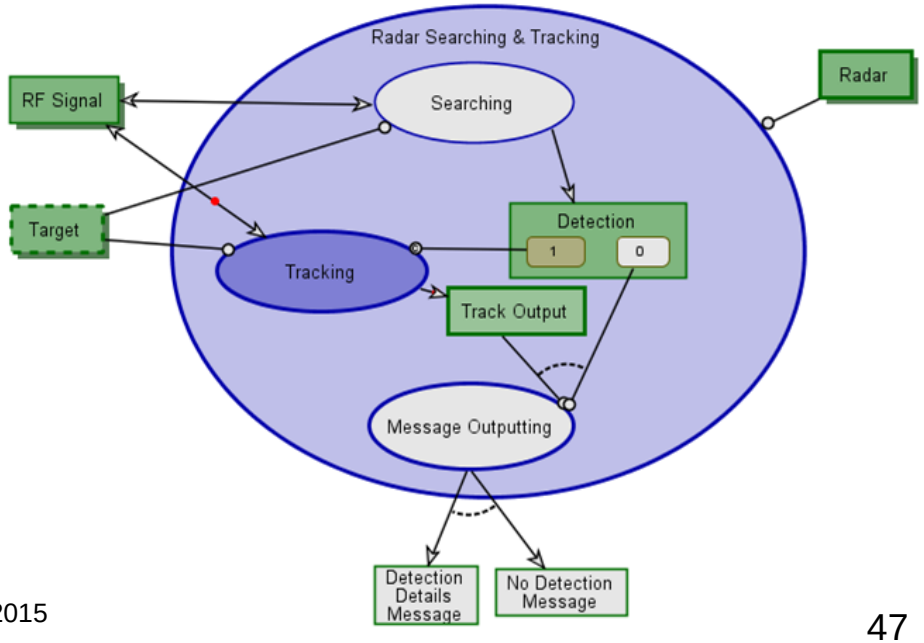


Op2Mat.txt

```
1  1
2  *
3  Searching
4  *
5  transmitting_power_W = "10000"
6  gain_TX_dB = "45"
7  gain_RX_dB = "45"
8  RCS_m2 = "17.2"
9  wavelength_m = "0.06"
10 distance_km = "31"
11 SNR_threshold_dB = "18"
12 *
13 detection = {"0","1"}
```

Mat2Op.txt

```
1  1
2  *
3  Searching
4  *
5  detection = {"1"}
```



OPM Computational Subcontractor Example Summary

- Simple radar equation implementation was demonstrated with both MATLAB and Simulink.
- Changing the level of complexity of the subcontracted model without changing the OPM model itself is easy.

Evaluation

Evaluation

- A thorough evaluation of the AUTOMATLAB approach was conducted as part of the 'Specification and Analysis of Information Systems' course.
- Evaluation was based on an OPM model of a **Web Based Grocery Shopping** system created by the students in the course.

Evaluation

- All students (N=12) had knowledge of OPM
- Some students (N_1=5) had prior knowledge of MATLAB.
- The rest (N_2=7) had none or very little knowledge of MATLAB
- About half the students received the automatically-generated MATLAB code from AUTOMATLAB.

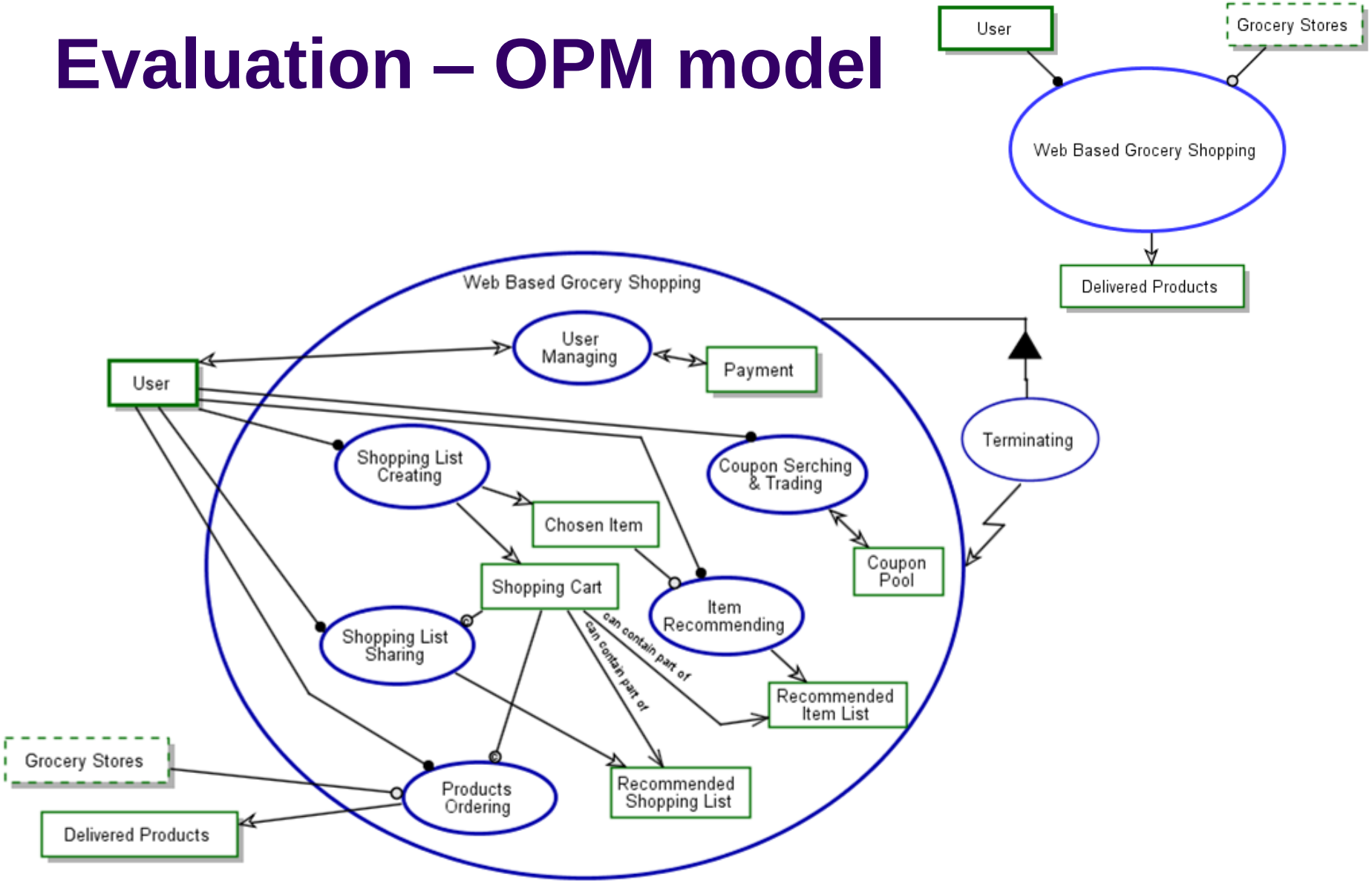
Evaluation

- These student were able to expand the MATLAB code to obtain answers.
- The other half (control group) were asked to answer the questions using any tool they desire, not receiving the code.

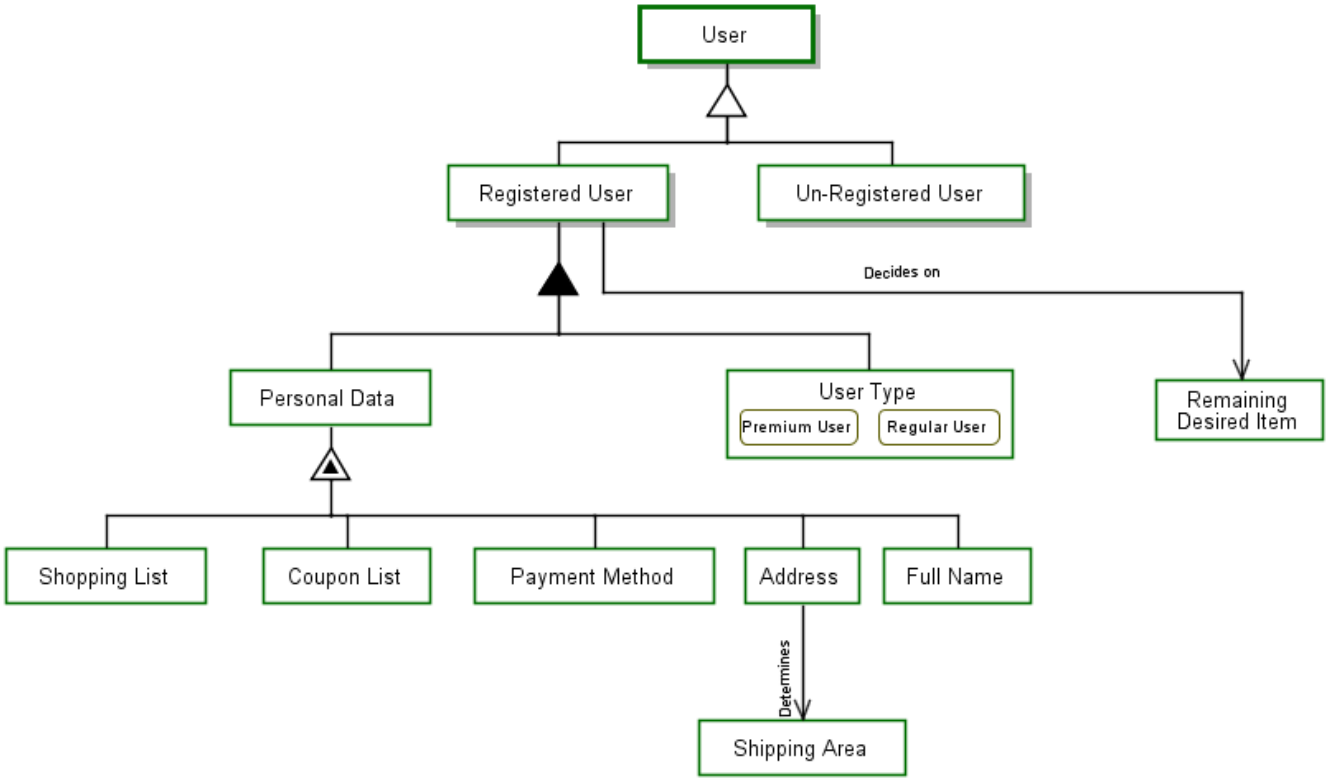
Evaluation

- The students with prior knowledge of MATLAB were the experimental group, while the rest served as the control group.
- In order to extend our sample, each student preformed the evaluation for two different data sets, achieving a total of $\tilde{N}=24$, with $\tilde{N}_1=10$ and $\tilde{N}_2=14$.
- The evaluation was based on an OPM model of a **Web Based Grocery Shopping** system which had been created by students in the course.

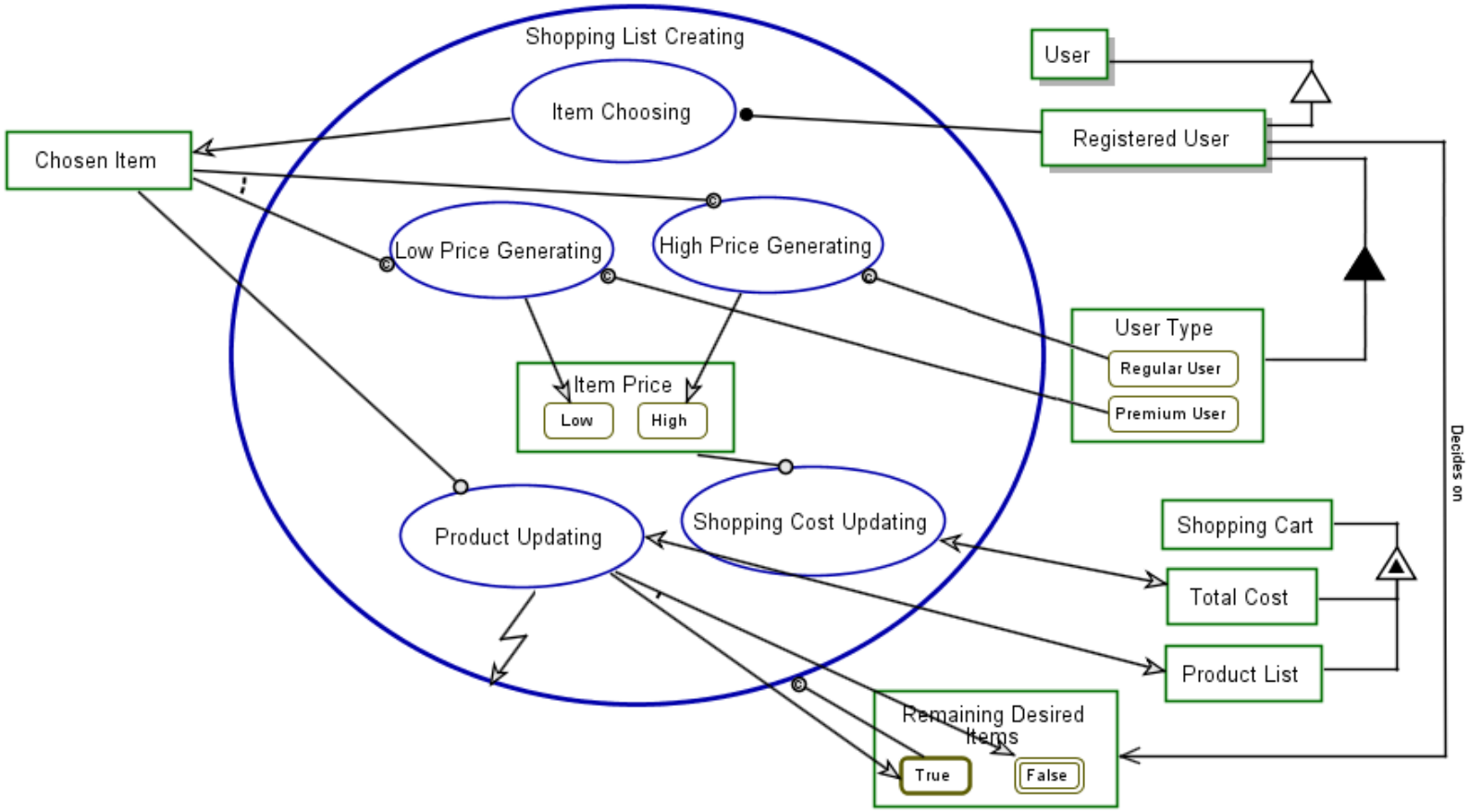
Evaluation – OPM model



Evaluation – OPM model



Evaluation – OPM model



Evaluation - Shopping List Creating MATLAB code

```
function [TotalCost,ProductList,RemainingDesiredItems,ChosenItem] = ShoppingListCreating(RegisteredUser,UserEntity,TotalCost,ProductList,RemainingDesiredItems)
% ShoppingListCreating zooms into ItemChoosing, LowPriceGenerating, HighPriceGenerating, ShoppingCostUpdating, and ProductUpdating, as well as Item Price.
while isequal(RemainingDesiredItems,'True') % ShoppingListCreating occurs if RemainingDesiredItems is True.
    if ~isempty(RegisteredUser)
        % RegisteredUser handles ItemChoosing.
        [ChosenItem] = ItemChoosing(RegisteredUser); % ShoppingListCreating consists of ItemChoosing
        % ItemChoosing yields ChosenItem.
    end
    if ~isempty(ChosenItem) && isequal(UserEntity,'Regular user')
        % HighPriceGenerating occurs if ChosenItem is in existent and UserEntity is Regular user.
        [ItemPrice] = HighPriceGenerating(ChosenItem,UserEntity); % ShoppingListCreating consists of HighPriceGenerating
        % HighPriceGenerating yields High ItemPrice.
    end
    if ~isempty(ChosenItem) && isequal(UserEntity,'Premium user')
        % LowPriceGenerating occurs if ChosenItem is in existent and UserEntity is Premium user.
        [ItemPrice] = LowPriceGenerating(ChosenItem,UserEntity); % ShoppingListCreating consists of LowPriceGenerating
        % LowPriceGenerating yields Low ItemPrice.
    end
    if ~isempty(ItemPrice) && ~isempty(TotalCost)
        % ShoppingCostUpdating requires ItemPrice.
        [TotalCost] = ShoppingCostUpdating(ItemPrice,TotalCost); % ShoppingListCreating consists of ShoppingCostUpdating
        % ShoppingCostUpdating affects TotalCost.
    end
    if ~isempty(ChosenItem) && ~isempty(ProductList)
        % ProductUpdating requires ChosenItem.
        [RemainingDesiredItems,ProductList] = ProductUpdating(ChosenItem,ProductList); % ShoppingListCreating consists of Productupdating
        % ProductUpdating affects ProductList.
        % ProductUpdating yields either True RemainingDesiredItems or False RemainingDesiredItems.
    end
    % ProductUpdating invokes ShoppingListCreating.
end
end
```

Evaluation - Shopping List Creating MATLAB code

```
function [ChosenItem] = ItemChoosing(RegisteredUser)
% RegisteredUser handles ItemChoosing.
% ItemChoosing yields ChosenItem.


% [] = RegisteredUser; % RegisteredUser handles ItemChoosing.

ChosenItem = 1; % ItemChoosing yields ChosenItem.

end
```

```
function [ItemPrice] = HighPriceGenerating(ChosenItem,UserEntity)
% HighPriceGenerating occurs if ChosenItem is in existent and UserEntity is Regular user.
% HighPriceGenerating yields High ItemPrice.



% [] = ChosenItem; % HighPriceGenerating occurs if ChosenItem is in existent

% [] = UserEntity; % HighPriceGenerating occurs if UserEntity is Regular user

ItemPrice = 'High'; % HighPriceGenerating yields High ItemPrice.

end
```

```
function [RemainingDesiredItems,ProductList] = ProductUpdating(ChosenItem,ProductList)
% ProductUpdating requires ChosenItem.
% ProductUpdating affects ProductList.
% ProductUpdating yields either True RemainingDesiredItems or False RemainingDesiredItems.



% [] = ProductUpdating; % ShoppingCostUpdating requires ProductUpdating.

[ProductList] = ProductList; % ProductUpdating affects ProductList.


RemainingDesiredItems = 'True'; % ProductUpdating yields either True RemainingDesiredItems or False RemainingDesiredItems.
% RemainingDesiredItems = 'False'; % ProductUpdating yields either True RemainingDesiredItems or False RemainingDesiredItems.

end
```

Evaluation hypothesis

- Our research hypothesis was that using OPM with the AUTOMATLAB approach would benefit the user in the following ways:
  - Users of AUTOMATLAB will gain deeper, more accurate understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.
  - AUTOMATLAB users will understand the system's computational and quantitative aspects quicker than users who used OPM without AUTOMATLAB.
  - AUTOMATLAB users will be more confident in their understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.
  - AUTOMATLAB users will understand the system's computational and quantitative aspects better, with less difficulty, than who used OPM without AUTOMATLAB.

Evaluation process

The students were asked to answer the following questions:

- What type of customer is more profitable for the iBuy owner: Regular user or Premium user?
- What are the three most profitable products for the iBuy owner?
- What is the premium user monthly fee that will maximize the profit for the iBuy owner?
- What is the premium user monthly fee that will make the amount of items purchased by regular users and premium users equal?

Evaluation data sets

Serial Number	Item Name	Item Group	Item Cost for Retailer	Selling Price for Regular Users	Selling Price for Premium Users
8196113	Apples	Fruits	8.03	11.00	8.48
5572352	Applesauce	Various Groceries	7.20	14.00	12.65
1274121	Asparagus	Vegetables	5.77	12.90	6.27
6680623	Avocados	Fruits	11.81	13.40	12.40
1489543	Bagels	Baked Goods	3.72	9.67	6.07
7501680	Baking powder	Baking Products	1.49	2.49	2.00
7680314	Baking soda	Baking Products	0.19	0.64	0.19
1603686	Bananas	Fruits	4.96	25.02	17.36
7755163	Basil	Spices and Herbs	1.18	2.18	1.18
2287275	Beef	Meats	13.19	44.03	37.64
9379260	Berries	Fruits	10.11	13.01	11.70
9769380	Black pepper	Spices and Herbs	0.38	1.38	0.38
7359278	Bread crumbs	Baking Products	1.08	2.08	1.08
1852531	Broccoli	Vegetables	3.66	3.00	2.10
8069030	Butter	Dairy and Cheese	2.00	3.13	2.03
3588132	Cake	Baked Goods	12.90	23.00	20.00
2043763	Cake icing	Baking Products	0.10	0.97	0.10
9536342	Cake mix	Baking Products	1.05	2.05	1.05
7640968	Canned olives	Various Groceries	8.89	17.14	13.51
2564885	Canned tuna	Various Groceries	3.18	7.96	7.46
7425307	Carrots	Vegetables	1.04	7.00	2.51
5542144	Cauliflower	Vegetables	5.18	7.90	6.12
8677460	Celery	Vegetables	0.24	9.60	0.20
1572087	Cheddar	Dairy and Cheese	5.42	6.88	5.95
6643642	Cherries	Fruits	16.67	19.02	16.63

Customer ID:	218207544	
User Type:	Regular user	
Serial Number	Item Name	Amount (items / Kg)
1603686	Bananas	9
2287275	Beef	5
7425307	Carrots	4
1572087	Cheddar	10
6501778	Cinnamon	7
3020596	Coffee	3
4688139	Crackers	1
2858148	Donuts	2
9967952	Fish sticks	3
7183122	Fresh bread	4
9432786	Frozen steak	2
8780756	Garlic	6
9216096	Grapefruit	10
1611773	Gum	5
4146529	Hamburgers	4
1140283	Honey	5
7165095	Kiwis	3
2445804	Lettuce	6
3480016	Mayonnaise	7
2465482	Melon	10
1614747	Milk	7
8918899	Mushrooms	2
7341510	Nectarines	4

Evaluation data analysis

- A total of 96 answers from 24 questionnaires
- Answers graded according to their accuracy.
- Student explanations regarding difficulty, confidence in the outcome accuracy, and the time required to complete the assignment were analyzed qualitatively.

	Experimental Group		Control Group	
	Jerusalem Data set	Tel-Aviv Data set	Jerusalem Data set	Tel-Aviv Data set
Amount of questionnaires:	5	5	7	7

Evaluation data analysis

- Data analyzed for three variables: *group* (experimental or control), *level* ('Jerusalem' or 'Tel-Aviv') and *question* (Q1, Q2, Q2, or Q4)
- Analysis using multi-way repeated measures tests with two within-subjects independent variables (*level, question*) and between-subjects independent variable (*group*).
- The dependent variable, namely *grade*, *time*, *confidence in answer accuracy*, and *difficulty* was changed in each hypothesis test.

Evaluation data analysis

- Independent t-test and one-way ANOVA with a Bonferroni correction served as our post-hoc tests, where it was needed.

Evaluation hypothesis - Results

- Our research hypothesis was that using OPM with the AUTOMATLAB approach would benefit the user in the following ways:

- Users of AUTOMATLAB will gain deeper, more accurate understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.

- AUTOMATLAB users will understand the system's computational and quantitative aspects quicker than users who used OPM without AUTOMATLAB.

- AUTOMATLAB users will be more confident in their understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.

- AUTOMATLAB users will understand the system's computational and quantitative aspects better, with less difficulty, than who used OPM without AUTOMATLAB.

Summary and Future Work

Summary

- This research tackles the problem of merging computational aspects and capabilities into conceptual models of systems, which are primarily qualitative in nature.
- Due to the level of abstraction of conceptual models, their computational capabilities may be weak or missing.