Incorporating Quantitative Aspects into OPM-based Conceptual Models with MATLAB Computational Capabilities

Aharon Renick

Incorporating Quantitative Aspects into OPM-based Conceptual Models with MATLAB Computational Capabilities

Research Thesis

In Partial Fulfillment of The
Requirements for the Degree of
Master of Science in Information Management Engineering

Aharon Renick

Submitted to the Senate of
the Technion - Israel Institute of Technology

Elul, 5773 Haifa August 2013

The research thesis was done under the supervision of Prof. Dov Dori in the Faculty of Industrial Engineering and Management.

I would like to thank Prof. Dov Dori for his devoted instruction and for his warm and personal treatment.

Thank you to Dr. Niva Wengrowicz, Mr. Sergey Bolshchikov and Mr. Alex Blekhman for all their help and great ideas.

Special thanks to my beloved wife Liat and my lovely daughters Linoy, Shoham and Shaked, for their support and patience throughout my studies.

Table of Contents

List of Figures

List of Tables

Abstract

Modeling is an important part of the lifecycle of systems, starting from the early design stages. Modeling is also very useful in the process of studying an unfamiliar, existing system. Conceptual modeling methodologies disregard certain aspects of the system, making modeling or understanding a model a simpler task as they convey the important aspects of a system in an effective way.

One of the shortcomings of conceptual modeling methodologies is the simplification of the system being modeled at the expense of suppressing computational aspects. This research presents two approaches for solving this computational simplification problem for conceptual models that use Object Process Methodology (OPM), an emerging ISO 19450 standard modeling methodology.

OPM offers a holistic approach for modeling systems that combines the structure and behavior of the system in a single diagram type. We expand the quantitative aspects of an OPM model by representing complex quantitative behavior using alternative approaches that employ MATLAB or Simulink without compromising the holism and simplicity of the OPM conceptual model. The first approach, AUTOMATLAB, expands the OPM model to a full-fledged MATLAB-based simulation. The second, OPM Computational Subcontractor approach, replaces low-level processes of the OPM model with computation-enhanced MATLAB functions or Simulink models.

We demonstrated the two approaches with MATLAB and Simulink enhanced OPM models of a biological system and a radar system, respectively. An evaluation the AUTOMATLAB approach, which compared system modeling and analysis with and without the AUTOMATLAB layer has indicated several benefits of the additional AUTOMATLAB layer compared to a non-enhanced OPM model.

Abbreviations

CASE - Computer-Aided Software Engineering

CS - Computational Subcontractor

GUI - Graphical User Interface

LSC - Live Sequence Chart

MATLAB - Matrix Laboratory

ML - MATLAB Layer

OMG - Object Management Group

OO - Object Oriented

OPD - Object Process Diagram

OPL - Object Process Language

OPM - Object Process Methodology

OPM/CS - Object Process Methodology Computational Subcontractor

SysML - Systems Modeling Language

UML - Unified Modeling Language

1. Introduction and Background

1.1. Conceptual modeling and conceptual model

An important stage of designing a complex system is modeling it. It is often necessary to create a conceptual model of a system as one of the first stages of its design. A conceptual model is a powerful tool in the process of understanding a system under design. Similarly, when aiming to research and fully understand existing systems, it is common to create a conceptual model of the system under study.

Different modeling methodologies, such as OPM (Dori, 2002) and SysML (Weilkiens, 2007), enable one to conceptually model a system and simulate its behavior. This approach is achieved by simplifying some level of reality, such as the level of detail of different aspects of the system (Zeigler, 1976). One aspect that is often simplified in OPM and SysML is the computational aspect of a system—the mathematical entities that may govern the actions and reactions of a system, the exact output of some actions, or even an accurate representation of random effects that can take place during a process. This simplification of the modeling methodology facilitates holistic understanding of the system. Yet, at times, this simplified qualitative-only view of the system might lack some important information, especially when the model is simulated and the dynamic aspect of the system needs to be explicitly expressed as the system changes over time. Moreover, while making progress in the design or study of a system, it is often necessary to fully simulate its structure and operation. The conceptual model alone cannot always convey the required information about the system. This is especially true for systems that exhibit complex behavior, which might include such elements as non-deterministic, stochastic behavior, advanced numerical calculations which drive different actions, and sophisticated quantitative decision-making processes. Advancing from the conceptual model to an elaborate simulation is therefore often critical for testing and validating the system under design, or confirming theories regarding systems under study.

In some cases, due to the human in the loop, the transition from the modeling stage to the simulation stage can result in errors or inaccuracies. Creating a simulation of the system can be done by studying the model or the original system directly, aiming to understand it, and building the simulation accordingly. As long as the model-to-simulation transition process involves human intervention, it is prone to mistakes and inaccuracies. Moreover, such manual transitions can overlook insights gained during early stage of the conceptual model.

1.2. The computational simplification problem

We define the computational simplification problem as the simplification of a conceptual model in a way that reduces its computational aspects. Some modeling methods, presented in the following sections, may suffer from this problem, while other methods that allow modeling at a lower level without losing the computational aspects, lack high level abstraction conceptual abilities. We will present possible solutions for this problem based on expanding Object Process Methodology (OPM) with the capabilities of MATLAB and Simulink, by integrating MATLAB and Simulink with the OPM enviroment.

1.3. OPM

Object Process Methodology (OPM) is an approach to conceptual modeling that uses a single unifying model capturing both the structural and behavioral aspects of a system[1] (Dori, 2002). An OPM model consists of two entities: objects and processes. Objects are the stateful components the system is made of, while processes are things that transform objects.

An OPM model is presented in two modalities: graphical and textual. Object Process Diagram (OPD) is the graphical representation of the model, while Object Process Language (OPL) is the parallel textual representation of the model. Both representations are completely interchangeable and convey the same information about the model.

OPM offers a CASE tool called OPCAT (Dori et al., 2010) for designing and testing OPM models. A future design platform called WebOpcat (Web OPCAT Project, 2013) is under development. Figure 1 shows the three groups of basic symbols of OPM as they appear in OPCAT's toolkit. Entities consist of objects, states and processes. Structural relations consist of Aggregation, Exhibition, Generalization & Classification relations. Procedural links include links between processes to object, including agent link, instrument link, effect link and more.

Figure 1: OPCAT toolkit

A summery of OPM rules can be found in appendix A. The full OPM ontology is described in 'Object-Process Methodology – A Holistic Systems Paradigm' (Dori, 2002).

1.4. MATLAB & Simulink

1.4.1. MATLAB

MATLAB (Houcque, 2005), short for Matrix Laboratory is a numerical computing environment developed by MathWorks, which is widely used in the fields of engineering and science. Thanks to its large selection of function libraries and many numerical abilities, MATLAB is commonly employed to build system simulations, allowing the user to easily solve differential equations, implement stochastic behavior, conveniently manipulate multi-dimensional arrays, etc.

Compared with conventional programming languages, such as C or FORTRAN, MATLAB has many advantages for solving technical problems (Houcque, 2005). MATLAB is an interactive system whose basic data element is an array that does not require dimensioning – so that the programmer can add and modify elements dynamically as the program proceeds, without defining them in advance. The software package has been commercially available since 1984 and is now considered a standard tool at most universities and industries worldwide. While the commercial version is commonly used, there are some free and open source MATLAB-compatible solutions that are very similar, such as Octave, Scilab, and FreeMat (Sharma & Gobbert, 2010). When referring to MATLAB, these solutions can be considered as well.

MTALAB has powerful built-in library functions that serve a wide variety of uses. Groups of functions for specific applications are collected in packages, referred to as toolboxes. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and many other fields. MATLAB also offers easy graphical command interface, enabling visualization of results immediately and conveniently.

1.4.2. Simulink

Simulink, a platform incorporated within MATLAB (Karris, 2008), is a way to program without using textual code, but by means of a graphical display – dragging and connecting predefined blocks, placing them in different places, masking them, and manipulating them. Simulink has many block libraries, including Math Operations, Model Verification, Ports &

Subsystems, Signal Routing, Sinks, and Sources. It is also possible to define blocks by programming with MATLAB code. Simulink's main addition to MATLAB is its graphical environment that replaces the textual code-based view. Similar to the original MATLAB environment, Simulink provides a set of block libraries, similar to the MATLAB library functions.

Simulink's graphical environment can serve as a means of modeling a system, leveraging on the computational strength of MATLAB that exists inherently in Simulink. As a simple step, one can build a simulation of a system represented in MATLAB by using Simulink, adding a graphical representation to the simulation.

Although Simulink enables a graphical approach to the simulation, it does not provide a complete solution to conceptual modeling. Rather, it provides a graphical view of the MATLAB code. The Simulink environment is limited to a box-diagram approach, representing each section or subsystem as a separate box, which can be drilled into, and can contain lower level subsystems as separate boxes. This approach mainly provides a view of the structure of a system, but it does not capture the dynamics of the system, nor does it represent relations between the system and the environment any more than a code based simulation.

1.5. Other Modeling Methodologies

1.5.1. UML and SysML

The Unified Modeling Language (UML) (Object Management Group, 2011) is an object-oriented modeling methodology that became the Object Management Group (OMG) standard for software systems development in 1997. UML consists of a model with fourteen different views, represented by different graphical diagram types. These fourteen views aim to convey different aspects of the system, its structure, behavior, relations, and change over time, so that a system modeled by UML consists of several related diagrams of different types.

The fourteen UML views are activity diagram, class diagram, communication diagram, component diagram, composite structure diagram, deployment diagram, interaction overview diagram, object diagram, package diagram, profile diagram, sequence diagram, state diagram, timing diagram and use case diagram.

Systems Modeling Language (SysML) (Object Management Group, 2012) is a profile of UML. SysML was created from UML, retaining seven of the UML diagram types, modifying

some of them, and adding two new ones. The language is aimed to be more system-centric, as opposed to the more software-centric UML.

One of the main differences between UML and SysML when compared to OPM is the holism of the model. While UML-based methods require many views to represent the system, OPM seeks to do so with a single view. Following this minimalism principle, while we wish to extend the computational power of OPM, we aim to do so while minimizing additional views of the system.

1.5.2. Modelica

Modelica (Mattsson & Elmqvist, 1997) is an object-oriented equation-based modeling methodology. The Modelica language defines and describes the system, its components, and behavior by a set of mathematical equations. The Modelica methodology includes CASE tools for designing Modelica models. These tools allow the user to draw or import a scheme of the system, connecting the model equations to the appropriate component on the scheme. With this tool, one can decompose the model hierarchically, simplifying the model and making it more understandable. Another feature of the Modelica methodology is a large selection of model libraries, offering predefined subsections of systems, sorted in different fields (electrical, mechanical, aerospace, etc.), which can be easily implemented as part of a model. Other predefined libraries allow the integration of numerical solving and stochastic behavior modules.

Modelica supports quantitative and stochastic aspects due to its equation-based representation, allowing direct modeling of the computational aspects of the system. It is also possible to describe the architecture of the modeled system using Modelica without the equation-based representation, but describing the behavior of the system requires the mathematical representation, demanding a greater level of effort from the user to comprehend the model in comparison to a simpler representation of a system flow, especially when she or he did not design it initially. Another aspect that should be noted is that the set of definitions in the Modelica language is object-oriented, confining the users to the OO paradigm.

1.5.3. Play-in/Play-out (Play engine)

The Play-in/Play-out approach (Harel & Marelly, 2003) is a methodology for modeling systems, specifically the reactions of the system with the environment and its different subsystems. In the Play-in/Play-out approach, scenarios are “played in” by the user modeling

the system using a tool called the “Play-engine”. The user executes the various actions that can affect both the system and the expected reaction of the system. This is done intuitively using a Graphical User Interface (GUI) that represents the system and its parts. As the behavior is played in, the play-engine automatically generates a set of Live Sequence Charts (LSCs), specifying the behavior of the system.

After creating LSCs that cover the system’s permitted actions and their result (and possibly forbidden actions), the model can be “played out”. In the play-out mode, the user can apply an action to the GUI and it will react according to the set of rules defined by the LSCs. Contradictions, undefined actions, and other errors can be found by playing out different scenarios.

The play-engine allows specifying an event value as a function predefined in the GUI code (for example, using Visual Basic). This enables modeling complex systems, where the result of an action is not as simple as a constant reaction, but rather has quantitative aspects. The play-engine has also limited support of non-deterministic actions. It is possible to define more than one possible reaction for each action “played-in”, and allocate a probability to each reaction.

At first glance it seems that the Play-in/Play-out approach might satisfy our needs for a modeling methodology that incorporates quantitative and stochastic aspects. Yet, a more detailed examination suggests that this is not entirely true. First, the Play-in/Play-out method is suitable for scenario-based modeling, emphasizing the behavioral aspects of the system (similar to state charts). This limits its relevance compared with OPM, which is a more holistic approach that is suitable for different types of systems. Second, the ability to model stochastic behavior using the play-engine is limited. As noted, it is possible to easily define a probability for each of multiple reactions possibly caused by a single action. Yet, non-uniform continuously distributed probabilities and other forms of stochastic behavior cannot be defined, at least not in a straightforward manner. Another disadvantage of the Play-in/Play-out method is the lack of ability to easily incorporate quantitative aspects into the model. It is possible to predefine functions as part of writing the GUI code in a visual programming environment, which does not provide for a straightforward implementation of quantitative aspects, as opposed to MATLAB’s direct access to arrays, its many toolboxes, etc.

1.5.4. MLDesigner

MLDesigner is an open environment for UNIX or Linux systems used for designing and testing of system architectures and their functions (Schultz et al., 2010). The MLDesigner model consists of a block diagram, containing a variant of C++ code representing the functionality of each block. The blocks can be modified (by editing its size, position, color, etc.), and many predefined code blocks are available, simplifying the design and execution of a model.

MLDesigner can be partially used as a computational level behind a model, similar to part of the solutions suggested in this work. An attempt to apply this approach has been presented by Schultz et al. (2010), where an OPM-to-MLDesigner translation was suggested in order to add simulation capabilities to an OPM model.

The main advantage of the OPM-MLDesigner approach is the built-in simulation abilities in the MLDesigner CASE tool. This tool can be more convenient than the MATLAB environment, especially for simple simulations. The OPM-MLDesigner approach has some downsides, making it unsuitable for our needs as a combination of modeling capabilities and quantitative simulation abilities. Mainly, MLDesigner is not as widespread as MATLAB and there is no Windows-compatible version. Secondly, the concept presented in the OPM-MLDesigner approach is to generate an MLDesigner model out of the OPM model. This is done by using OPL—the textual modality of the OPM model—to generate an MLDesigner model, referred to as MML. MML is not linked to the OPM model and OPCAT simulation tool, and does not affect the original OPM model. In other words, the OPM-MLDesigner approach does not improve OPM's numeric abilities, but rather generates the OPM model in a different language. Another issue is the incompatibility of the MLDesigner language with OPM. MLDesigner does not have an entity fully comparable with the OPM concept of object, as it focuses on the process, so it is necessary to define "dummy" processes to play the role of these objects. This can substantially complicate the model.

1.5.5. Arena

Arena (Kelton et al., 2000) is a widely used, general purpose simulation tool, which enables the user to build a visual model of the system using a graphic editor, while the functionality of a model is converted into the SIMAN language. The modules of a system are represented on the graphical editor as boxes of different shapes, while connecting lines represent

connections and relative actions between modules. Each model has its flow and timing defined, from its arrival in the system until its departure. Arena supports common functions and mathematical operators, and allows the user to define new models (and behavior) by using external solutions, such as Visual Basic.

An approach to designing simulations using OPM was presented in (Gilat, 2002). The research focused on using OPM to better specify and design a model of a given simulation in order to improve its design and clarity. As a case study, the approach was demonstrated using the Arena simulation tool, where the advantages of connecting Arena and OPM in various scenarios were shown in experiments. In the Arena-OPM approach, the advantages of conceptual modeling are utilized only in the simulation design stage, and are not used in the simulation creation and operation stages. In contrast, we aim to improve the model and simulation by incorporating the quantitative aspects in the model itself, benefiting from both the advantages of the simpler model and the numerical abilities of the simulation tool. Therefore, the Arena-OPM approach does not satisfy the requirement of a combined conceptual and computational modeling methodology.

2. Research Motivation and Goals

One of the most basic requirements of a modeling language and simulation is to be sufficiently expressive to model the phenomena encountered in the design of a system, such as non-linear, multi-disciplinary, continuous or discrete flows. The models must also be easy to create and reuse (Sinha et al., 2001). Simplification of the computational aspect of a system—the accurate representation of the system's behavior—might conceal some important information about the system. Object Process Methodology might suffer from this computational simplification problem.

Modeling methodology simulation software should integrate with common design tools used by experts in different domains in order to effectively collaborate on the design of complex artifacts using the different tools (Sinha et al., 2001). OPM presents clear, intuitive and easily reusable models and simulation capability, but may lack the ability to allow the modeling of the complex quantitative and computational aspects of a system. While OPM enables exporting a model as an XML file, JAVA code and more, it does not fully integrate with common development tools such as MATLAB. As we show in the sequel, exporting a MATLAB-based representation of the OPM model allows one to express the model so that a domain expert who uses MATLAB can understand the system without knowledge in OPM.

A conceptual model is required to avoid being overly complex. In other words, "The conceptual modeling mantra is one of developing the simplest model possible to meet the objectives of the simulation study" (Robinson, 2010). Figure 2 illustrates how excessive levels of detail hamper model accuracy. At a certain point, the increasing level of detail reduces the model accuracy due to lack of knowledge on how to model such details. It is thus clear that we would want a model to be as simple as possible while still allowing sufficient accuracy and detail required for the model objective.

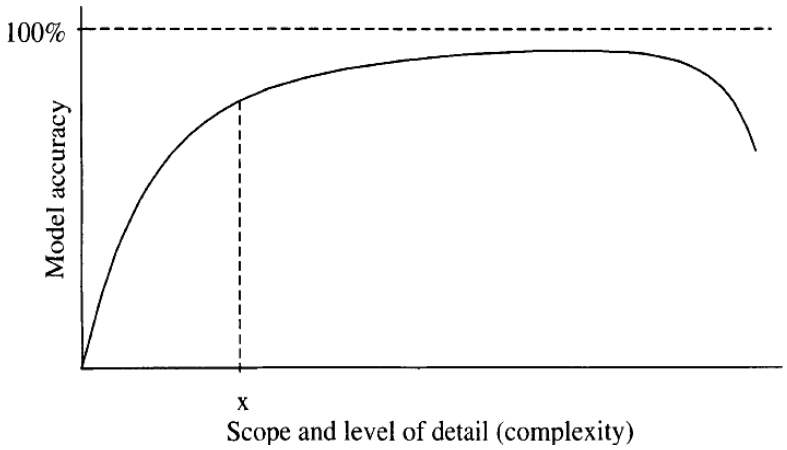


Figure 2: Model and simulation accuracy vs. complexity (Robinson, 2008)

Although simple models are desirable, a simpler model requires larger assumptions regarding the system, risking the possibility of missing important aspects of the system (Davies et al., 2003). It is thus clear why modeling a system with a solution such as MATLAB or Simulink alone, without OPM, is undesirable. The combination of OPM and MATLAB or Simulink enables model simplification while avoiding the risk of over-simplification by incorporating MATLAB and Simulink representations where required.

In our effort to present a solution to the computational simplification problem, we are faced with the following challenge: On the one hand, mathematical aspects of a system, such as arithmetic or differential equations, are often needed as part of the description of the system and its behavior. On the other hand, the simplified modeling tool may lack the ability to describe complex structures or behaviors that can easily be specified by code or a programmable block diagram. Such extensions should part of the conceptual model. A designed, a model creator or a domain expert should be able to incorporate them into the conceptual model in order to make it more complete.

In view of this need, our research aims to bridge the gap between the qualitative and the quantitative model aspects by allowing the use of OPM as the holistic, basically qualitative methodology, while providing a MATLAB or Simulink-based computational layer to enhance OPM and minimize the computational simplification problem. This enhancement must be done with minimal constraints on the original methodology, OPM. We present and compare two approaches to solving the qualitative-quantitative gap: OPM MATLAB Layer and OPM Computational Subcontractor.

As the design or study of a system progresses, an elaborate simulation is often created for examining the system in operation. While in general a human in the loop may introduce errors during the transition from the conceptual model to a simulation, our approach provides for a MATLAB-based simulation, which is created directly from the evolving OPM model, avoiding possible introduction of new errors.

3. OPM MATLAB Layer – AUTOMATLAB

3.1. AUTOMATLAB Description

The modeling methods presented above provide conceptual models of the system under development or study and enable some kind of simulation. However, in most cases, either the level of the quantitative aspect of the simulation is insufficient, or it is achieved at the expense of sacrificing simplicity and generality of the method and the resulting model. MATLAB is a convenient tool for simulating complex systems, but it does not have advanced abstract conceptual modeling capabilities. Adding a numerical computational layer to the conceptual modeling power of OPM provides for simulating its behavior both qualitatively and quantitatively.

The architecture of OPM AUTOMATLAB Layer, shown in Figure 3, consists of three main processes:

- AUTOMATLAB code generating, which uses the graphical (OPD) or textual (OPL) representation of the OPM model. This initial code is identical to the OPM model in the sense that it describes the same system structure and behavior. We will refer to this stage as the **MATLAB Code Generating** stage.
- AUTOMATLAB code enhancing, where the user can utilize predefined standard OPM models with computational and numerical representations or add computational aspects of interest as code in the MATLAB file. We will refer to this stage as the **MATLAB Code Enhancing** stage.
- AUTOMATLAB enhanced simulating of the OPM model in OPCAT. We will refer to this stage as the **Enhanced Model Simulating** stage.

OPD describing the Architecture of AUTOMATLAB:

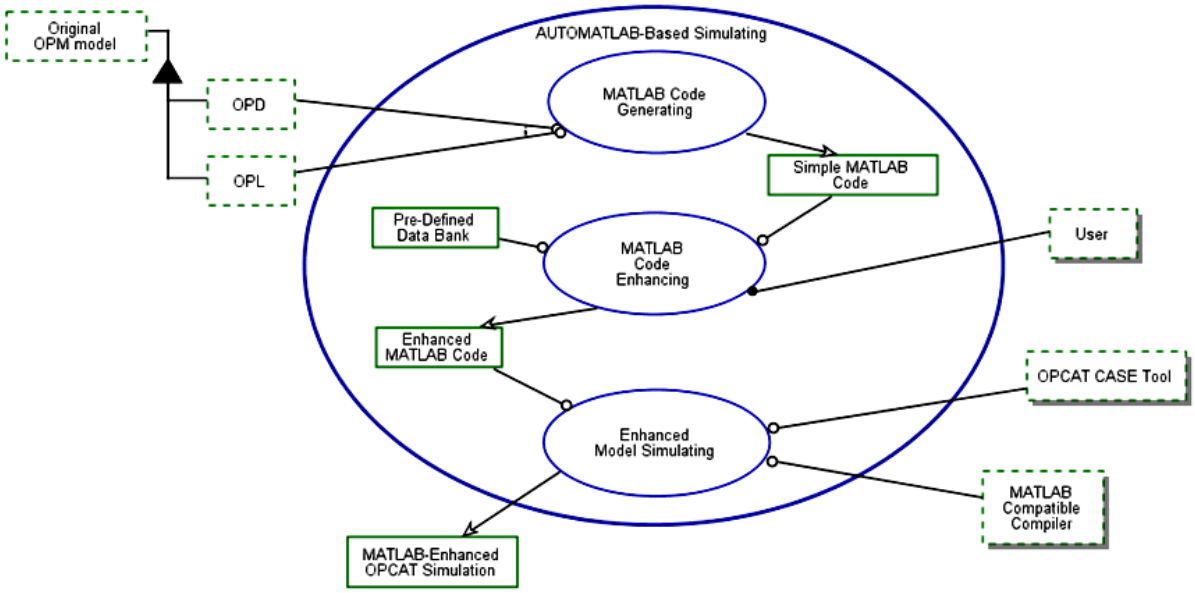


Figure 3: Architecture of AUTOMATLAB

The OPL that corresponds to Figure 3 follows.

AUTOMATLAB-Based Simulating consists of MATLAB Code Generating, MATLAB Code Enhancing, and Enhanced OPCAT Simulating.
AUTOMATLAB-Based Simulating exhibits Simple MATLAB Code, Enhanced MATLAB Code, and Pre-Defined Data Bank.
AUTOMATLAB-Based Simulating zooms into MATLAB Code Generating, MATLAB Code Enhancing, and Enhanced OPCAT Simulating, as well as Pre-Defined Data Bank, Enhanced MATLAB Code, and Simple MATLAB Code.
MATLAB Code Generating requires either OPD or OPL.
MATLAB Code Generating yields Simple MATLAB Code.
MATLAB Code Enhancing requires Simple MATLAB Code and Pre-Defined Data Bank.
MATLAB Code Enhancing yields Enhanced MATLAB Code.
Enhanced OPCAT Simulating requires Enhanced MATLAB Code, MATLAB Compatible Compiler, and OPCAT CASE Tool.
Enhanced OPCAT Simulating yields MATLAB-Enhanced OPCAT Simulation.
Original OPM model consists of OPL and OPD.
User handles MATLAB Code Enhancing.

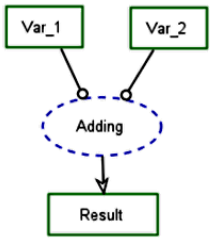
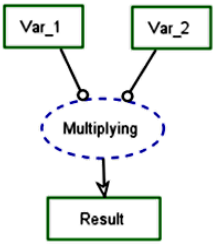
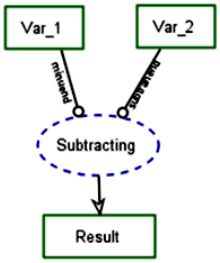
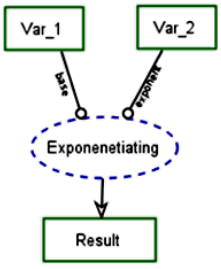
3.2. OPM-MATLAB equivalence

In order to generate the AUTOMATLAB layer efficiently and accurately in the code generating stage, it is necessary to understand the computational meaning of different things (objects and processes) and connections between them in the OPM model. To do so, we define the syntax and semantics for the various OPM-MATLAB translations. The purpose of this definition is to enable the user—the model creator—to create the OPM model of the AUTOMATLAB layer as independently as possible, while still conveying the computational

meaning of the model. A guideline for defining the OPM-MATLAB syntax is to minimize modifying the core OPM syntax, as defined in Dori et al. (2011).

As a first step, we have mapped the main basic built-in MATLAB functions in the MATLAB documentation (MathWorks Documentation Center MATLAB Functions, 2011) to their OPM model equivalents. To distinguish these built-in functions from user-created functions that have different meanings, we defined the process names listed in Table 1 through Table 4 as reserved words.

Table 1: AUTOMATLAB arithmetic operators

Symbol	**Operator/Process Name**	**OPD**
+	Addition / Adding	
*	Multiplication / Multiplying	
–	Subtraction / Subtracting	
^	Exponentiation / Exponentiating	

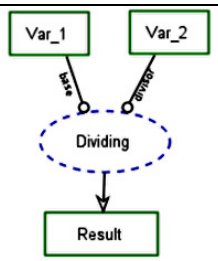
\	Division / Dividing	

Table 2: AUTOMATLAB loops and control structures

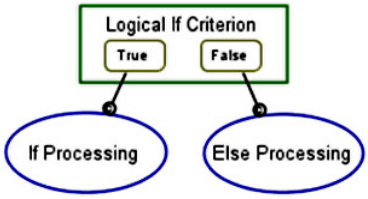
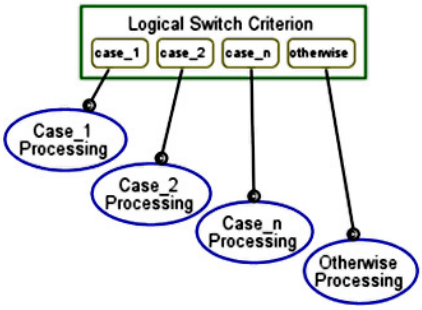
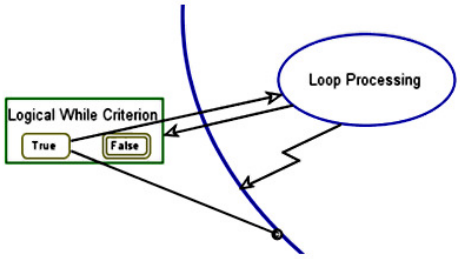
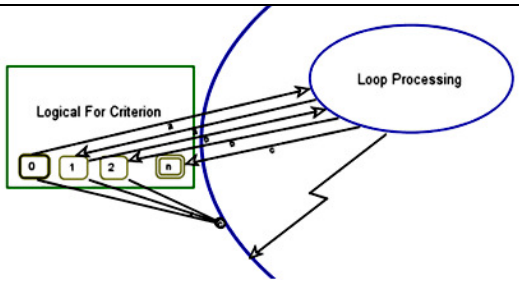
Operator / Process Name	**OPD**
if then…else	
switch	
While	
for	

Table 3: AUTOMATLAB trigonometric & exponential functions

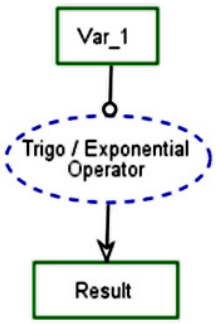
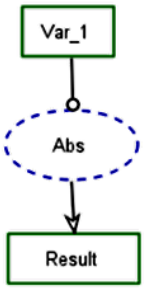
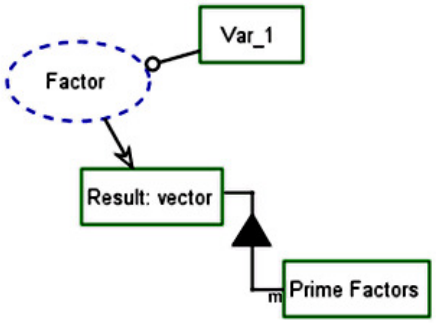
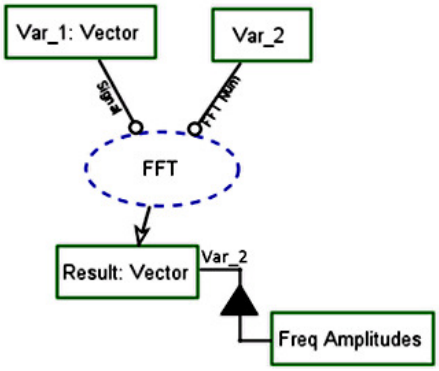
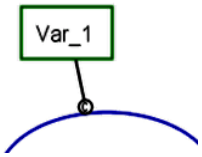
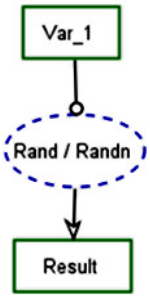
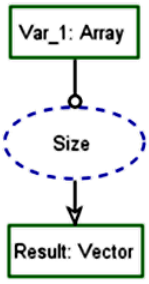
Operator / Process Name	Description	OPD
acos	Inverse cosine; result in radians	
asin	Inverse sine; result in radians	
atan	Inverse tangent; result in radians	
cot	Cotangent of argument in radians	
sin	Sine of argument in radians	
tan	Tangent of argument in radians	
exp	Exponential	
log	Natural logarithm	
log10	Common (base 10) logarithm	
sqrt	Square root	

Table 4: AUTOMATLAB miscellaneous functions

Operator / Process Name	Description	OPD
abs	Absolute value	
factor	Returns a row vector containing the prime factors of input.	

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 of code if is empty)	
rand	Uniformly distributed pseudorandom numbers	
randn	Normally distributed pseudorandom numbers	
size	returns the sizes of each dimension of array	

3.3. The AUTOMATLAB code generator

As part of the code generating stage in the AUTOMATLAB approach, a MATLAB code which is equivalent to the OPM model is generated. In order to minimize changes to the OPCAT tool, we generate the code using an exported html file of the OPM model from OPCAT. The html file is read by the code generator, and then processes, objects, values and relations are identified from the different html statements. Using the created list of all the things—processes and objects—their relations are mapped in three matrices: process-to-process relations, object-to-object relations, and process-to-object relations. Then, each relation is translated into the appropriate code segment in a separate MATLAB file called *m file*. For example, as Figure 4 shows, a 'requires' relation between a 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
```

Figure 4: 'A requires B' MATLAB code

When other relations are translated, the code will change accordingly. For example, suppose that in addition to the example in Figure 4, a 'yields' relation exists between process A and object C, which means that C results from executing A. In that case, the code in Figure 4 will be altered to the one shown in Figure 5.

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

Figure 5: 'A requires B 'and 'A yields C' MATLAB code

In the MATLAB function representing the process A, the code in Figure 6 is generated.

```
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
```

Figure 6: MATLAB code equivalent to process A

The AUTOMATLAB-generated MATLAB code is simple, and it does not convey any information beyond the original OPM model. The code is generated so that it is executable without the need for further alteration beyond what the AUTOMATLAB code generator produces. When the OPM model does not have the information needed to complete a legal code segment, that piece is left commented. For example, we noted that the process A requires B, but as long as there is no in-zoomed view of A containing the nature of this relation, the use of B in function A remains as a comment, as shown in line 5 in Figure 6. In a similar way, all the process-to-process relations (consists of, zooms into, etc.), object-to-object relations (consists of, exhibits, is a, etc.) and process to object relations (consumes,

requires, changes, etc.) are translated into corresponding code segments that convey the same semantics as their OPM counterparts. The complete set of code segments produced by the AUTOMATLAB code generator appears in appendix B.

3.4. mRNA Lifecycle: an AUTOMATLAB Case study

We demonstrate an OPM model of a molecular biology system that AUTOMATLAB enhances. We used an OPM model of a biological process called **mRNA Lifecycle** presented in (Somekh et al. 2012).

As the OPD in Figure 7 shows, **mRNA Lifecycle** was in-zoomed to expose the processes **Transcription** and **Translation**. **Transcription** is further in-zoomed into three subprocesses: **Initiation**, **Elongation**, and **Termination**. Each of these subprocesses affects the objects **DNA**, **Transcription Factors**, **PolI II**, **rpb4/7**, **Export Factor** and **mRNA** by creating them, changing their states, creating a link between them, and changing their location.

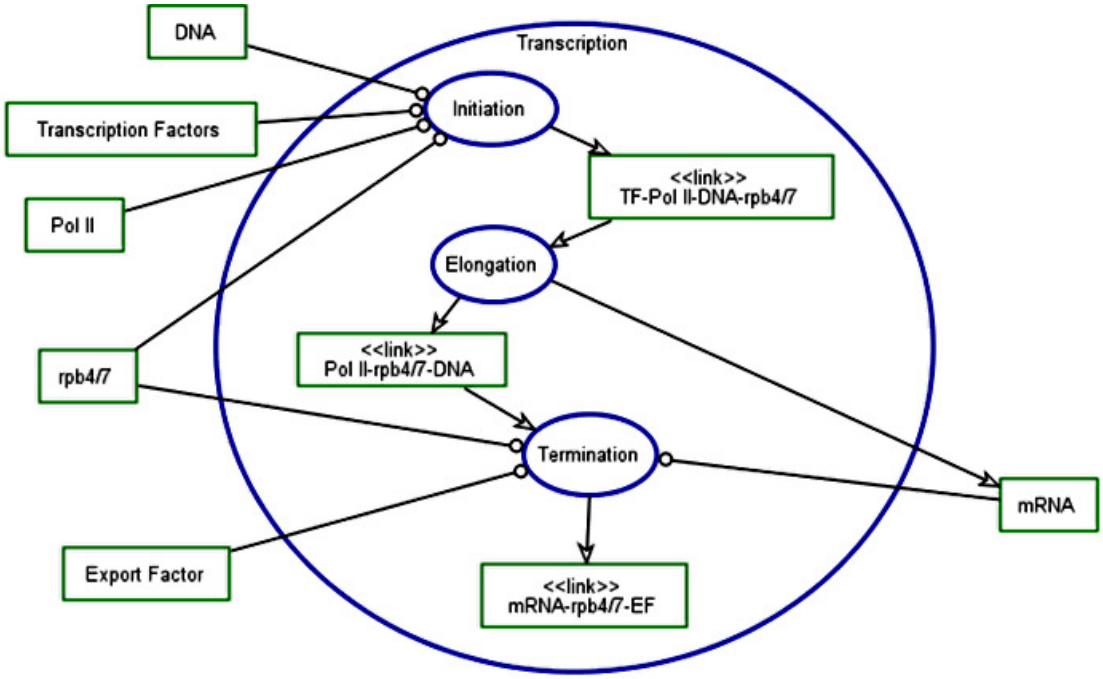


Figure 7: The OPD of the **Transcription** process in-zoomed

The original OPM model was deterministic, whereas the actual biological process being modeled is assumed to have a stochastic behavior. The original model does not account for the duration of each process either. These two features are implemented as part of AUTOMATLAB. To this end, the following capabilities have been added to the OPM model:

- Upon triggering, the **Initiation** process occurs randomly with probability 0.7.

- The **Initiation** process has random duration, which is uniformly distributed from 2 to 3 milliseconds.
- A failure of the **Initiation** process to occur has a duration, which is uniformly distributed from 0.5 to 1 millisecond.

The AUTOMATLAB layer was created in two stages. First, in the code generating stage, the basic MATLAB code, shown in Figure 8, was created by AUTOMATLAB's automatic MATLAB code generator, as described in the previous section. This code represents the original OPM model, without the AUTOMATLAB enhancements.

```
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
```

Figure 8: Basic MATLAB code of the in-zoomed **Transcription** process, generated by AUTOMATLAB

Next, in the code enhancing stage, we added user enhancements into the MATLAB layer. The added model properties are boxed and marked in red in Figure 9. The simulation was then executed repeatedly a sufficient number of times to enable collecting statistics on different parameters and plot various graphs. For example, Figure 10 shows a graph of the **Initiation** process duration over 1000 runs of the simulation. In this case, the communication between MATLAB and OPCAT utilized the communication used for a Vivid OPM demo (Bolshchikov et al., 2010), which allows for stopping and resuming the OPCAT simulation at will. The execution demonstrated that when the random process **Initiation** did not occur, the OPCAT simulation did not proceed to the next process. By break-pointing the MATLAB code, we could verify that the progress of the simulation as seen in OPCAT was linked to the MATLAB simulation progress, and proceeds only as break-points were released.

```
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
```

Figure 9: Enhanced MATLAB code of the process **Transcription**, added code boxed and marked red

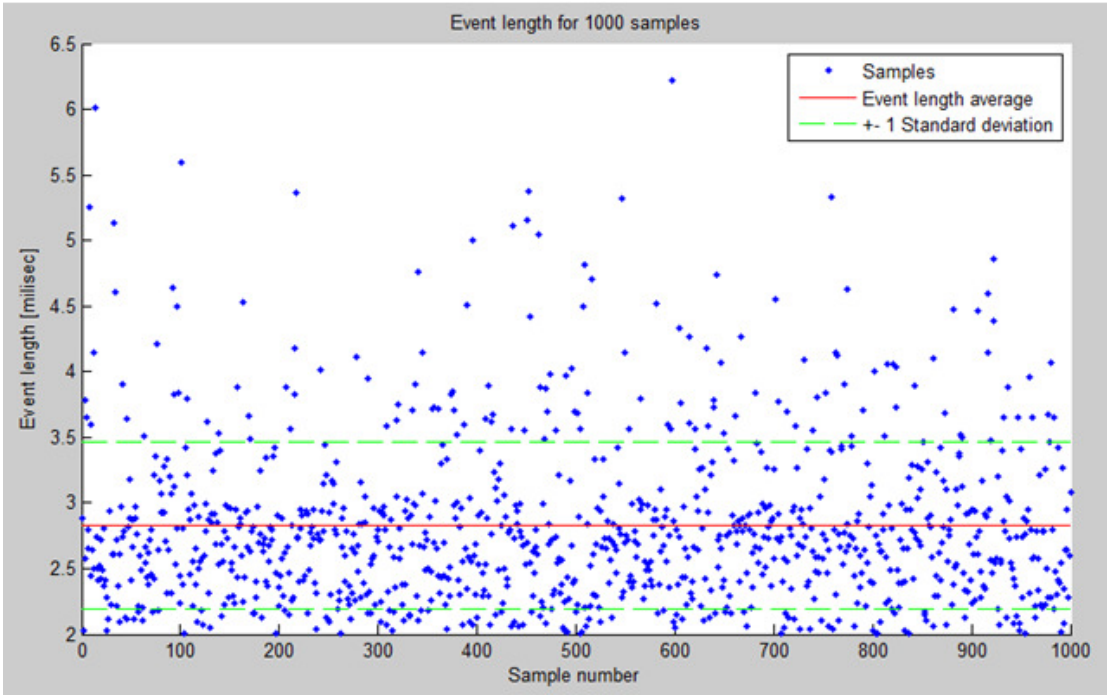


Figure 10: Event length of the **Transcription** process (y axis) as computed over 1000 runs of the simulation (x axis)

A demonstration of the simulation in OPCAT and MATLAB for this example can be found at https://www.youtube.com/watch?v=sto-bvUkGfg.

4. OPM Computational Subcontractor – OPM/CS

4.1. OPM/CS Description

One of drawbacks of the AUTOMATLAB approach presented in Section 3 is that it allows the violation of the OPM semantics by manipulating the MATLAB code such that the result is not a legal OPM model. This implies that in order to utilize the AUTOMATLAB approach correctly, one needs to master the OPM semantics. When creating a standard OPM model, this is not absolutely necessary, since OPCAT enforces many of the OPM rules.

To overcome this problem, we have developed the OPM MATLAB Computational Subcontractor (OPM/CS) as an alternative solution to the computational simplification problem. This solution uses MATLAB or Simulink as a "computational subcontractor" for the OPM model simulation. We augment a regular OPM model, such that any process can be in-zoomed by a MATLAB or Simulink diagram instead of the standard OPM in-zooming mechanism. When the OPM simulation is executed, it runs normally according to the OPM semantics until it reaches a process that was in-zoomed by the computational subcontractor. At this point, the input to the process – the existing objects and their states – are sent to the MATLAB function or to the Simulink diagram, and the sub-simulation function is called. The outcome of this sub-simulation defines the outcome of the in-zoomed process, which might include newly created objects, the state of an object upon exiting the process, or a value of an attribute. The OPCAT simulation then continues accordingly.

If the subcontracted process is a commonly used mathematical function, the user does not need to create it in MATLAB or Simulink, but rather mark it with the notation <<CS>> and it will be in-zoomed by a predefined function from a function library, as presented in Table 1 through Table 4. In this case, the user does not need to have any knowledge of MATLAB or Simulink, and can still use them as computational subcontractors. If the process is in-zoomed by a user-defined function or diagram, the process is linked to the appropriate MATLAB function or Simulink diagram by specifying the folder in which the user-defined function or diagrams are saved, using the same name for both the OPM process and the MATLAB function or Simulink diagram.

In contrast to AUTOMATLAB, the OPM semantics in OPM/CS cannot be breached, since each called MATLAB or Simulink function exists only within a single in-zoomed process, and it can only affect objects within the scope of that process. To ensure that this is the case, the use of global variables in the subcontracted functions is not allowed.

4.2. Implementation

The OPM/CS approach contains three parts: the OPM model in OPCAT, The MATLAB or Simulink models library, and an OPM/CS manager. The OPM/CS manager was designed for full compatibility to the future OPM environment Web OPCAT, and it maintains partial compatibility with the current OPCAT academic version. The subcontractor manager includes support for both MATLAB and Simulink as alternative subcontractors and the use of predefined common functions as defined in Table 1 through Table 4. The manager controls the bidirectional information transactions between the OPM environment and the MATLAB/Simulink environment. It includes a patch for OPCAT to manage the process inputs and outputs (see Figure 11 and Appendix C).

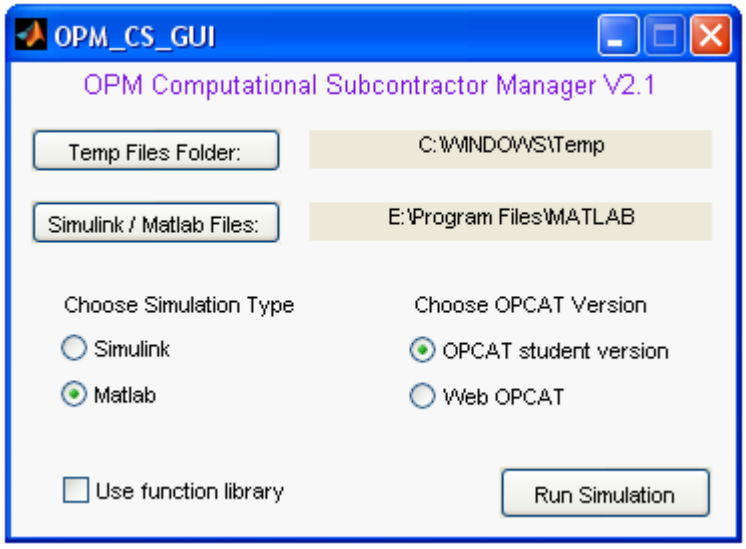


Figure 11: The OPM Computational Subcontractor Manager

An OPD describing the Architecture of OPM Computational Subcontractor is presented in Figure 12.

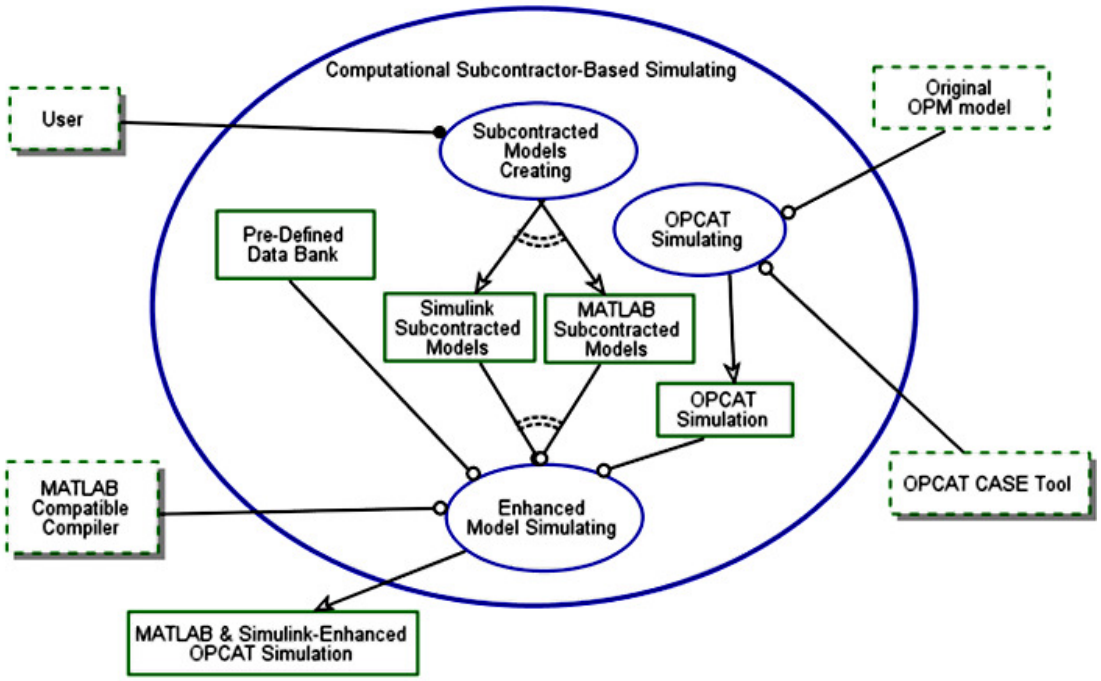


Figure 12: Architecture of OPM Computational Subcontractor.

The corresponding OPL follows.

Computational Subcontractor-Based Simulating exhibits MATLAB Subcontracted Models, Pre-Defined Data Bank, Simulink Subcontracted Models, and OPCAT Simulation.
Computational Subcontractor-Based Simulating consists of Subcontracted Models Creating, Enhanced Model Simulating, and OPCAT Simulating.
Computational Subcontractor-Based Simulating zooms into Subcontracted Models Creating, OPCAT Simulating, and Enhanced Model Simulating, as well as OPCAT Simulation, Simulink Subcontracted Models, Pre-Defined Data Bank, and MATLAB Subcontracted Models.
Subcontracted Models Creating yields Simulink Subcontracted Models or MATLAB Subcontracted Models.
OPCAT Simulating requires Original OPM model and OPCAT CASE Tool.
OPCAT Simulating yields OPCAT Simulation.
Enhanced Model Simulating requires OPCAT Simulation, MATLAB Compatible Compiler, and Pre-Defined Data Bank.
Enhanced Model Simulating requires Simulink Subcontracted Models or MATLAB Subcontracted Models.
Enhanced Model Simulating yields MATLAB & Simulink-Enhanced OPCAT Simulation.
User handles Subcontracted Models Creating.

As can be seen in Figure 13, Unlike the AUTOMATLAB approach, in OPM/CS we do not have at any point a full model including the enhanced aspects but rather have external enhancements in the form of the MATLAB files or Simulink diagrams. Therefore the OPM Computational Subcontractor model does not include an object **Enhanced OPM Model**, This

is in contrast to the AUTOMATLAB model (Figure 3), that has the object **Enhanced Matlab Code** which represents the enhanced OPM Model.

The information transaction from the OPM/CS manager to the OPM environment is implemented using temporary text files, allowing for compatibility with the future browser-based Web OPCAT. The OPM/CS manager runs in the MATLAB environment, and it calls the MATLAB functions (expressed as m files) or Simulink diagrams (expressed as mdl files) directly from this environment.

4.3. Radar Search & Tracking: an OPM/CS Case study

The case study that demonstrated the use of OPM/CS is a search and tracking radar system. The OPM model of this system is presented in Figure 13.

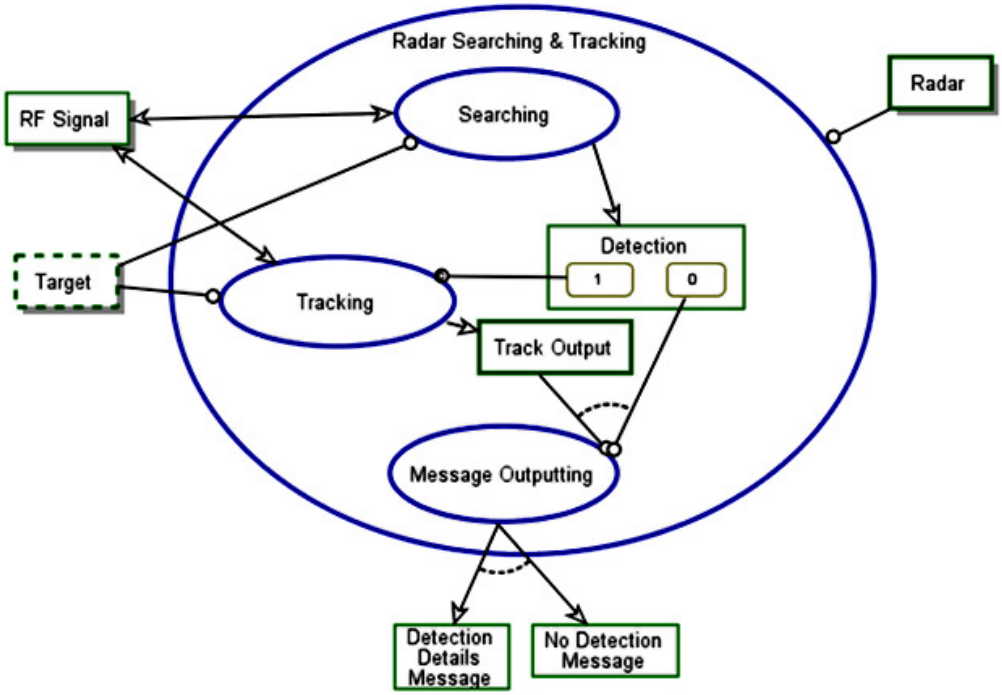


Figure 13: Radar Searching & Tracking in-zoomed

The **Searching** process creates a **Detection** output, which can be either **0** (no detection) or **1** (positive detection). If detection is achieved, the **Tracking** process tracks the **Target** and creates a **Track Output**, and when this process ends, either **Detection Details Message** or **No Detection Message** is created.

Zooming into the **Searching** process (see Figure 14), we see that it consist of two subprocesses: **Transmitting** and **Receiving & Processing**.

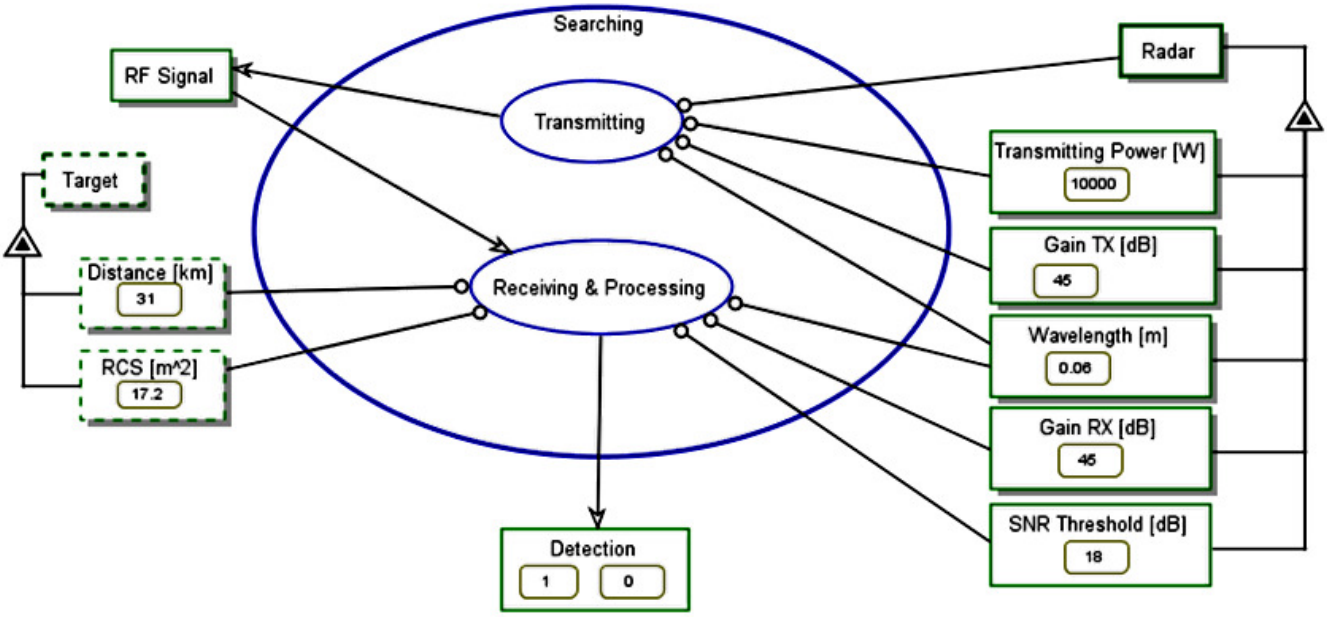


Figure 14: SD1.1 - Searching in-zoomed

Transmitting requires the radar characteristics **Transmitting Power [W]**, **Gain TX [dB]**, and **Wavelength [m]**, and creates the **RF signal**. **Receiving & Processing** consumes the **RF signal**, and it also requires the **Target** characteristics and some of the **Radar** characteristics.

When running the OPM simulation of the model shown in Figure 13 and in Figure 14, **Searching** creates detection in either the **1** state or the **0** state randomly, regardless of the **Target** and **Radar** attribute values. However, the received power of a target is described by the following Radar Equation (Scolnik, 1980).

$$P_r = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 R^4 L} \qquad (1)$$

In most cases, the Radar Equation is written in log 10 form ($10 \cdot \log_{10}(X)$), so that it takes the following form:

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

In this equation, P_r is the received power, P_t is the transmitted power, G_t is the transmitter gain, G_r is the receiver gain, σ is the target radar cross-section, λ is the radar wavelength, R is the target distance, and L represents radar losses. If the ratio (or difference, in the log form) between P_r and the level of noise is above the SNR (signal-to-noise) threshold, the target will be detected.

A classical OPM model of the Radar Equation implementation for **Searching** is complex and unintuitive due to lack of ability to represent formulae in a simple form. For example, in Figure 15 we used the definitions for addition, subtraction, multiplication, common log and the control structure 'if then…else' defined in Table 1 through Table 4.

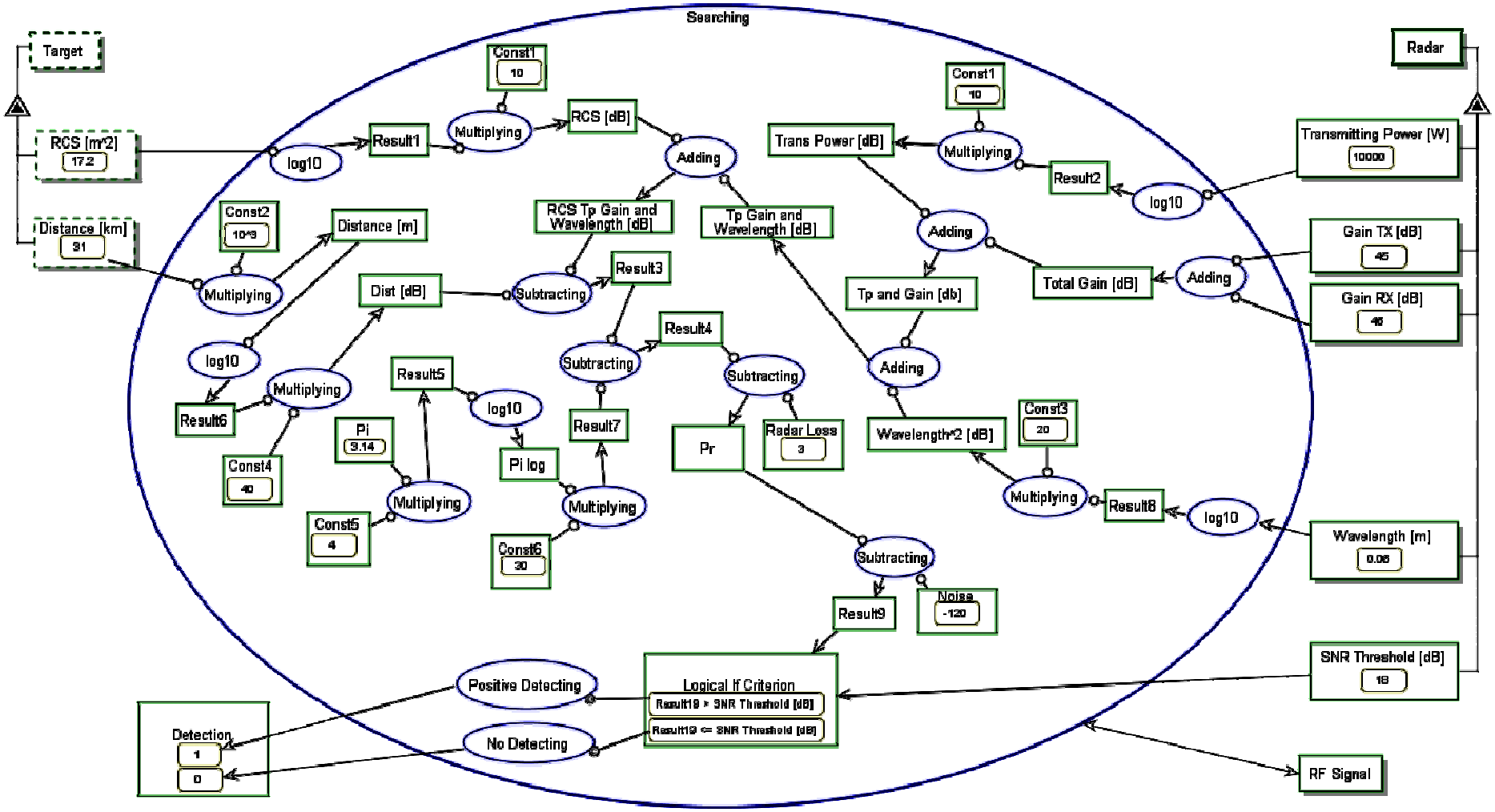


Figure 15: OPM model of **Searching** with the simple Radar Detection condition

The in-zoomed content of **Searching** can be replaced by the MATLAB code in Figure 16, which describes the Radar Equation.

```
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
```

Figure 16: MATLAB code of the simple Radar Detection condition

Evidently, this OPM model for representing the relatively simple radar equation is complicated. It is much simpler to do so using MATLAB code in Figure 16. Alternatively, the in-zoomed content of **Searching** can be represented by the Simulink diagram in Figure 17. Since we have chosen at this point a simple representation of the radar equation, the Simulink model seems over-complicated, yet less complicated than the OPM model in Figure 15. The MATLAB code captures the simplicity of the mathematical condition. As we show below, using Simulink is better suited for more complex cases.

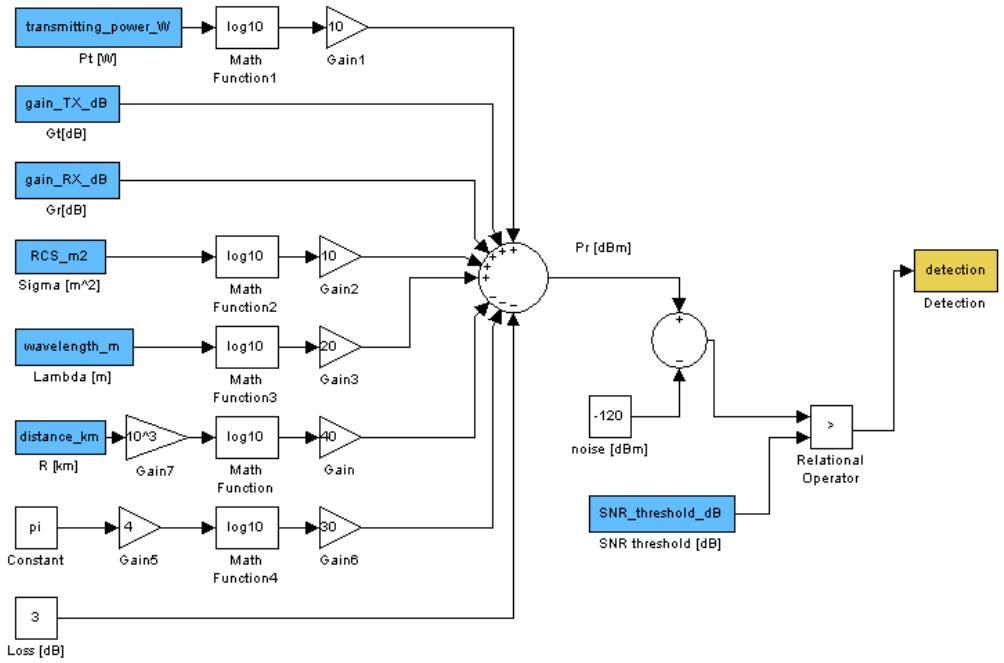


Figure 17: A Simulink diagram of the Radar Detection condition

When using the computational subcontractor approach, the OPM simulation, executed using OPCAT, arrives at the **Searching** process. When MATLAB or Simulink is called, it reads the data files containing the inputs to the diagram, calculates the outcome, and returns the appropriate output using the data files. A demonstration of a successful detection, based on computing **Pr** in the Radar Equation, is shown in Figure 18 trough Figure 21.

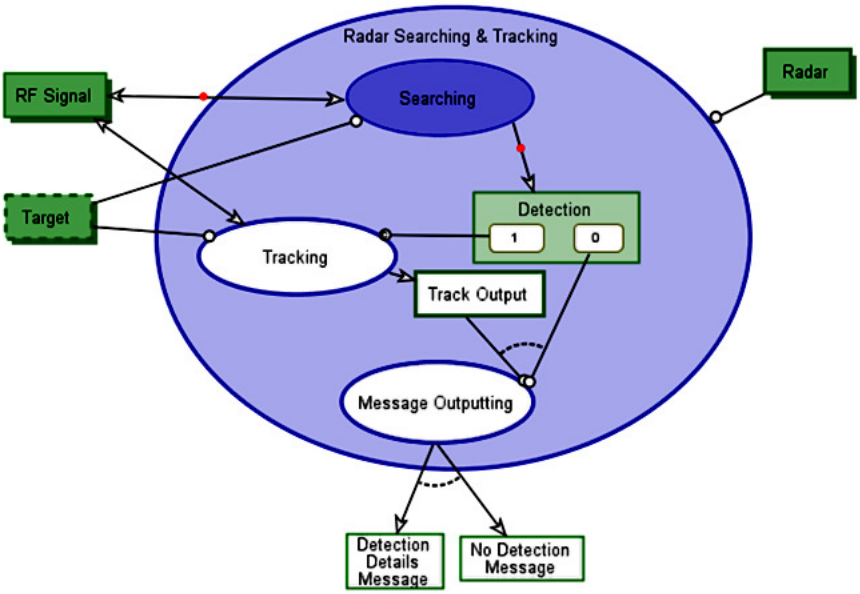


Figure 18: The MATLAB-enhanced OPCAT simulation: The OPCAT simulation calls the subcontracted **Searching** MATLAB function.

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"}
```

Figure 19: A call for the subcontracted **Searching** function with its input values as an example of a message sent from OPCAT to MATLAB

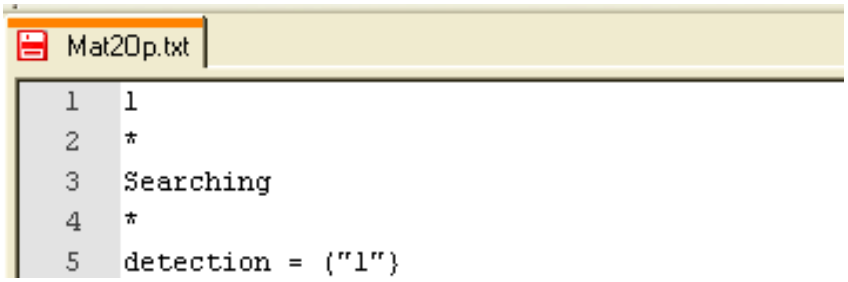


Figure 20: The outcome message of the subcontracted function **Searching** with **Detection** value set to 1 returned from MATLAB to OPCAT

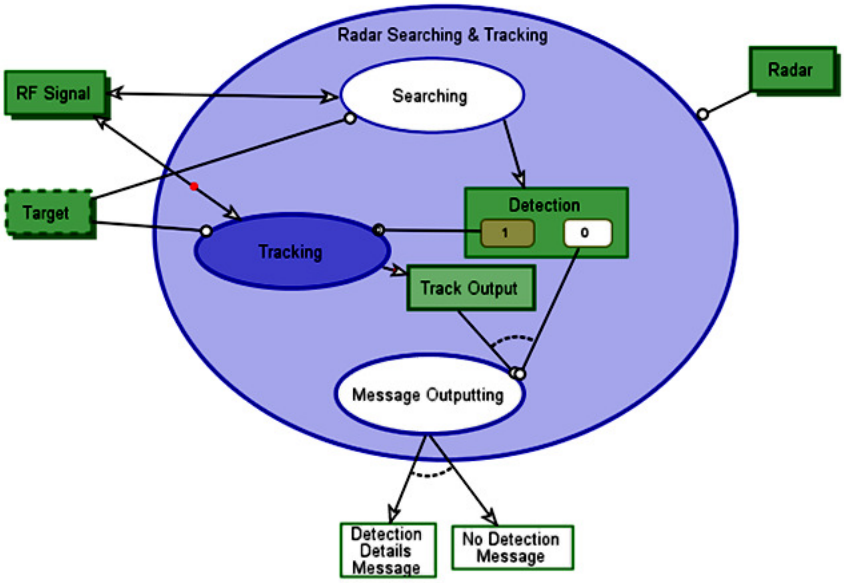


Figure 21: The outcome of the subcontracted **Searching** function in MATLAB with **Detection** value set to 1, causing the OPCAT simulation to proceed accordingly.

An important aspect of using MATLAB or Simulink as a subcontractor for the OPM model is the ability to change the level of complexity of the Simulink model without changing the OPM model itself. For example, we can replace the simple Simulink model in Figure 17 with the more complex model presented in Figure 22. This more elaborate model can be used instead of the model in Figure 17 without altering the OPM model whatsoever, providing for further simulation of the radar systems according to the designers' needs.

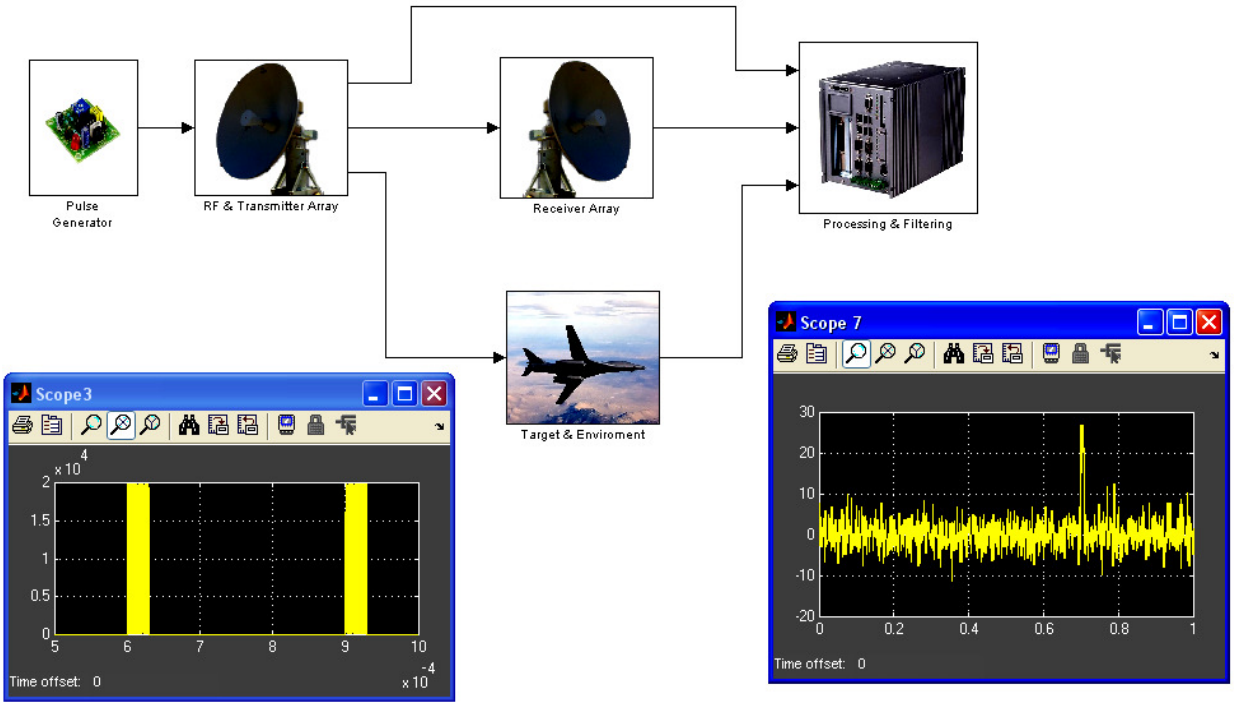


Figure 22: A Simulink diagram of the full Radar simulation, Radar modulated pulse signal (left scope) and SNR graph with positive detection (right scope)

5. Evaluation

The AUTOMATLAB approach and the Computational Subcontractor approach were evaluated with case studies described in sections 3.4 and 4.3. A more thorough evaluation of the AUTOMATLAB approach was conducted with human subjects in order to more thoroughly assess its benefits and outcomes.

5.1. Evaluation background and population

The evaluation took place during the spring semester 2013, at the Technion, Israel Institute of Technology. A total of 12 undergraduate students from the Faculty of Industrial Engineering and Management who participated in the course 'Specification and Analysis of Information Systems' took part in the evaluation at the end of the semester.

All students (N=12) had knowledge of OPM, as it was taught throughout the semester as part of the course. Some students (N_1=5) had prior knowledge of MATLAB from previous courses or work, while the rest (N_2=7) had none or very little knowledge of MATLAB. 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 shown in Figure 23 through Figure 26, which had been created by students in the course. The students who were not part of the group that created the model were briefed about the system and its model.

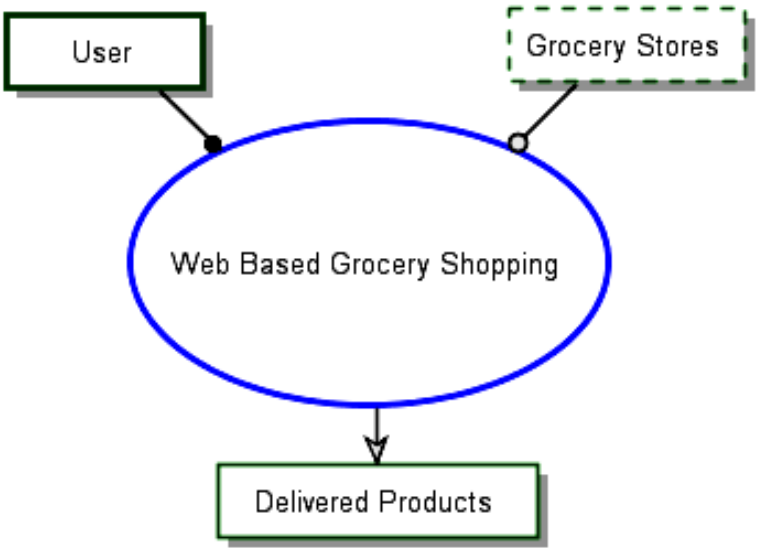


Figure 23: Top-level, system diagram (SD) of the **Web Based Grocery Shopping** system

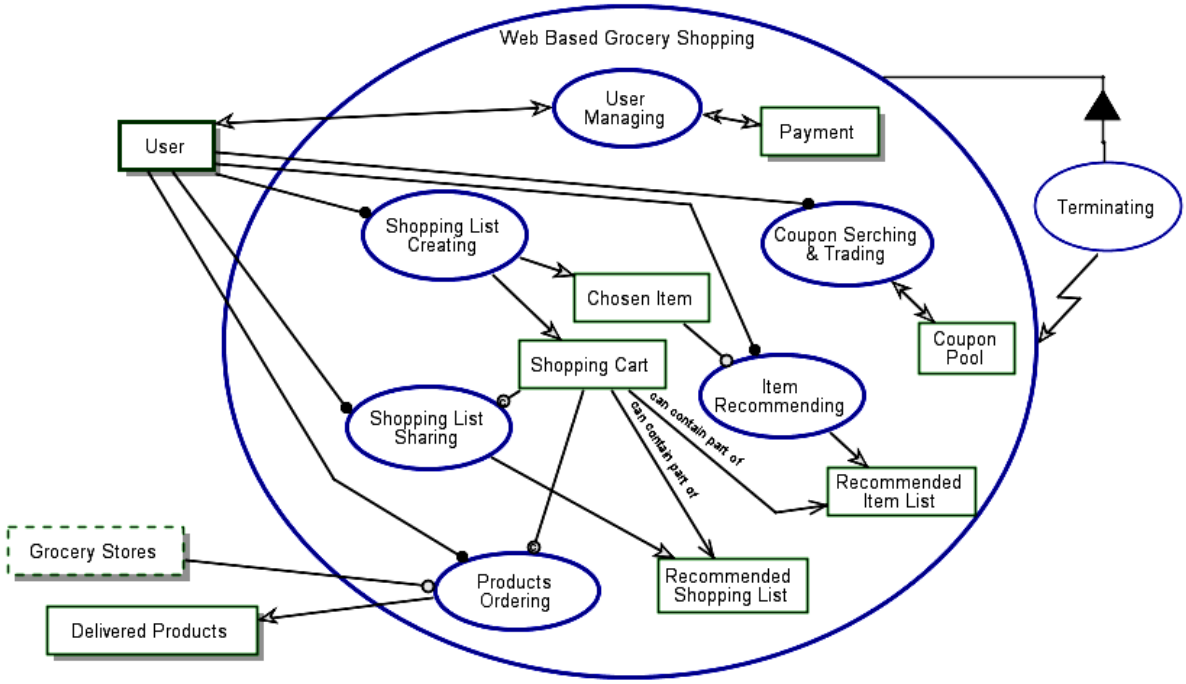


Figure 24: SD1 - **Web Based Grocery Shopping** in-zoomed

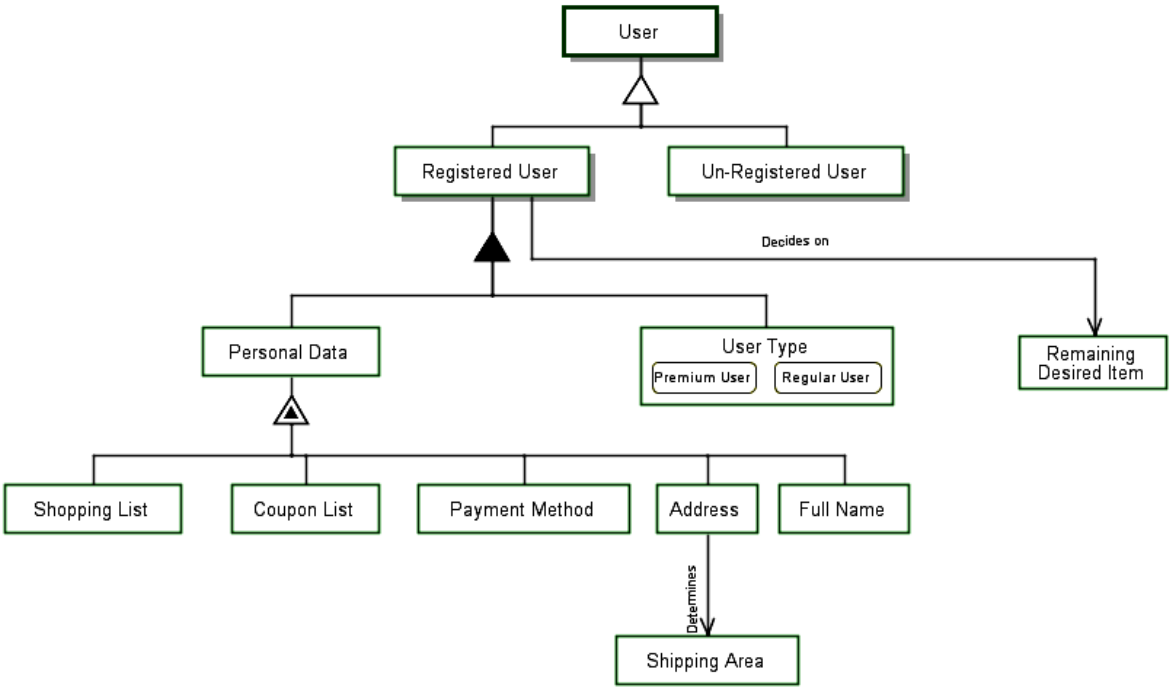


Figure 25: View 2 - **User** unfolded

The evaluation focused on the **Shopping List Creating** process, which is shown in Figure 26.

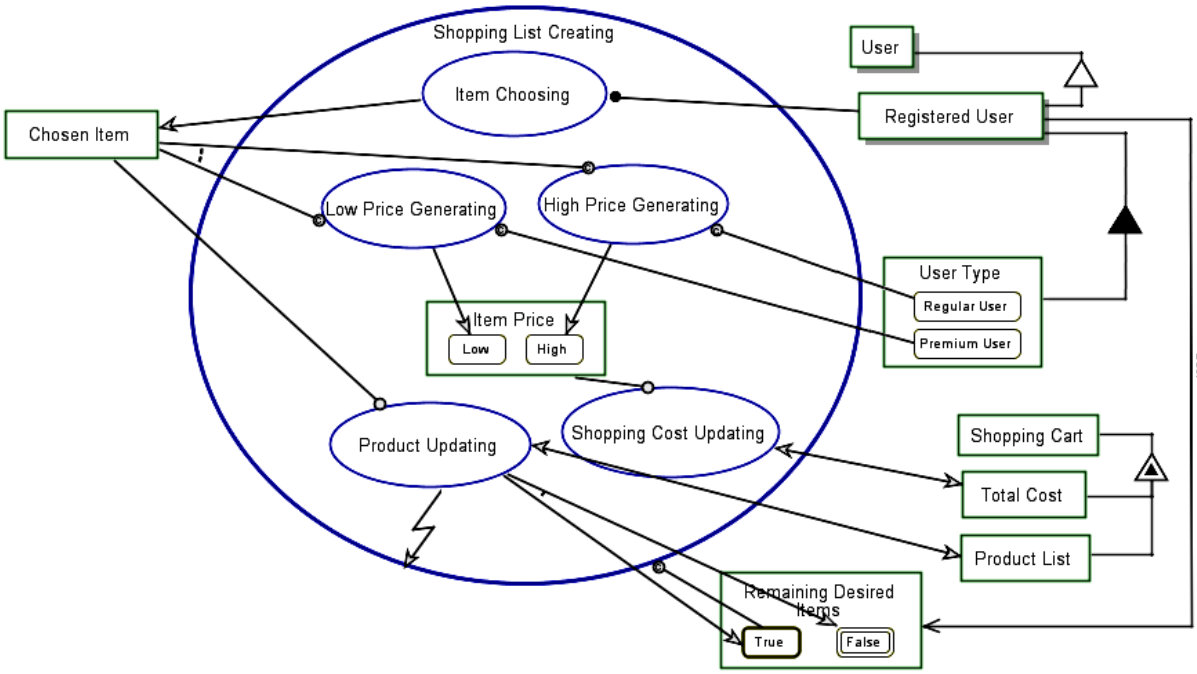


Figure 26: SD1.3 - **Shopping List Creating** in-zoomed

5.2. Evaluation hypothesis

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

1. Users of AUTOMATLAB will gain deeper, more accurate understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.
2. AUTOMATLAB users will understand the system's computational and quantitative aspects quicker than users who used OPM without AUTOMATLAB.
3. AUTOMATLAB users will be more confident in their understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB.
4. AUTOMATLAB users will understand the system's computational and quantitative aspects better, with less difficulty, than who used OPM without AUTOMATLAB.

5.3. Evaluation process

The students were requested to analyze some potential factors of a **Web Based Grocery Shopping** system called iBuy, which is specified in the Scope & Requirements Document (Appendix D) and to use the OPM model they had created, shown in Figure 23 through Figure 26. The different factors to be analyzed referred mainly to the **Shopping List Creating** process in the OPM model shown in Figure 26. The students were requested to evaluate the answer to four different questions, and answer a few questions regarding their evaluation, confidence, difficulty, and the time it took them to achieve their answer. The analysis was preformed on two data sets with different levels of difficulty.

The students were asked to answer the following questions:

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

For each answer, they were then asked to explain how they had deduced their answer, how accurate they thought their answer was and why, how difficult it was to complete their

answer and why, and how long it took them to complete their answer. The first three pages of the questionnaire are shown in Figure 27 through Figure 29. The full questionnaire is provided in Appendix E.

Since our sample was not sufficient for accurate statistical analysis, we combined qualitative analysis of the evaluation with the statistical analysis. In order to increase the statistical significance of the evaluation, each student was required to participate twice, each time with a different data set, where the second data set was more complex than the first one.

Advanced Topic Assignment

Specification and Analysis of Information Systems (094222) - Spring 2013

(Page 1 of 7)

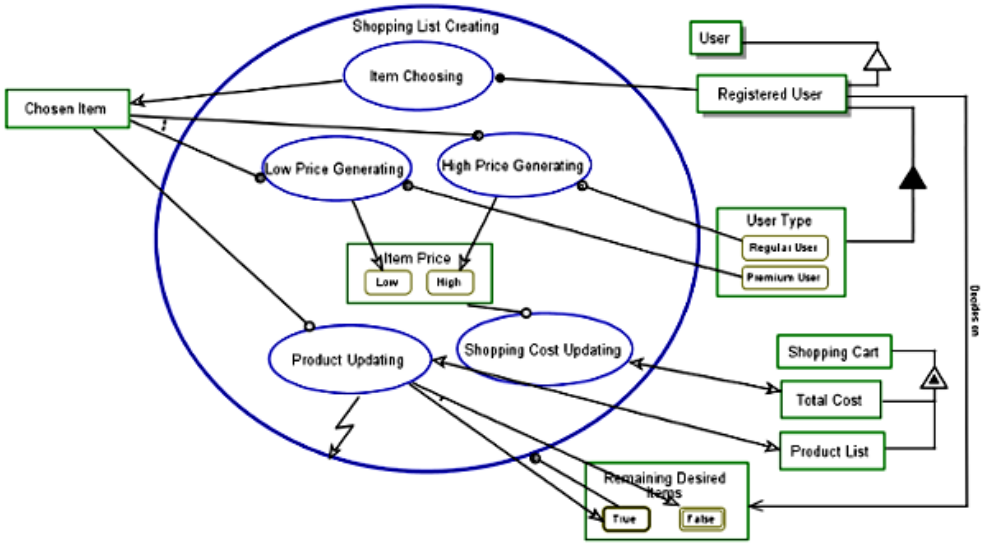
Background

As part of your course assignments, you are required to analyze some potential factors of the 'iBuy' system detailed in the Scope & Requirements Document, using the OPM model created by you or your fellow students.
The different factors to be analyzed will refer mainly to the 'Shopping List Creating' process in the OPM model.

As part of the assignment you are requested to evaluate the answer to four different questions. After each answer you will be asked to answer a few questions regarding your evaluation and confidence in the answer you provided.

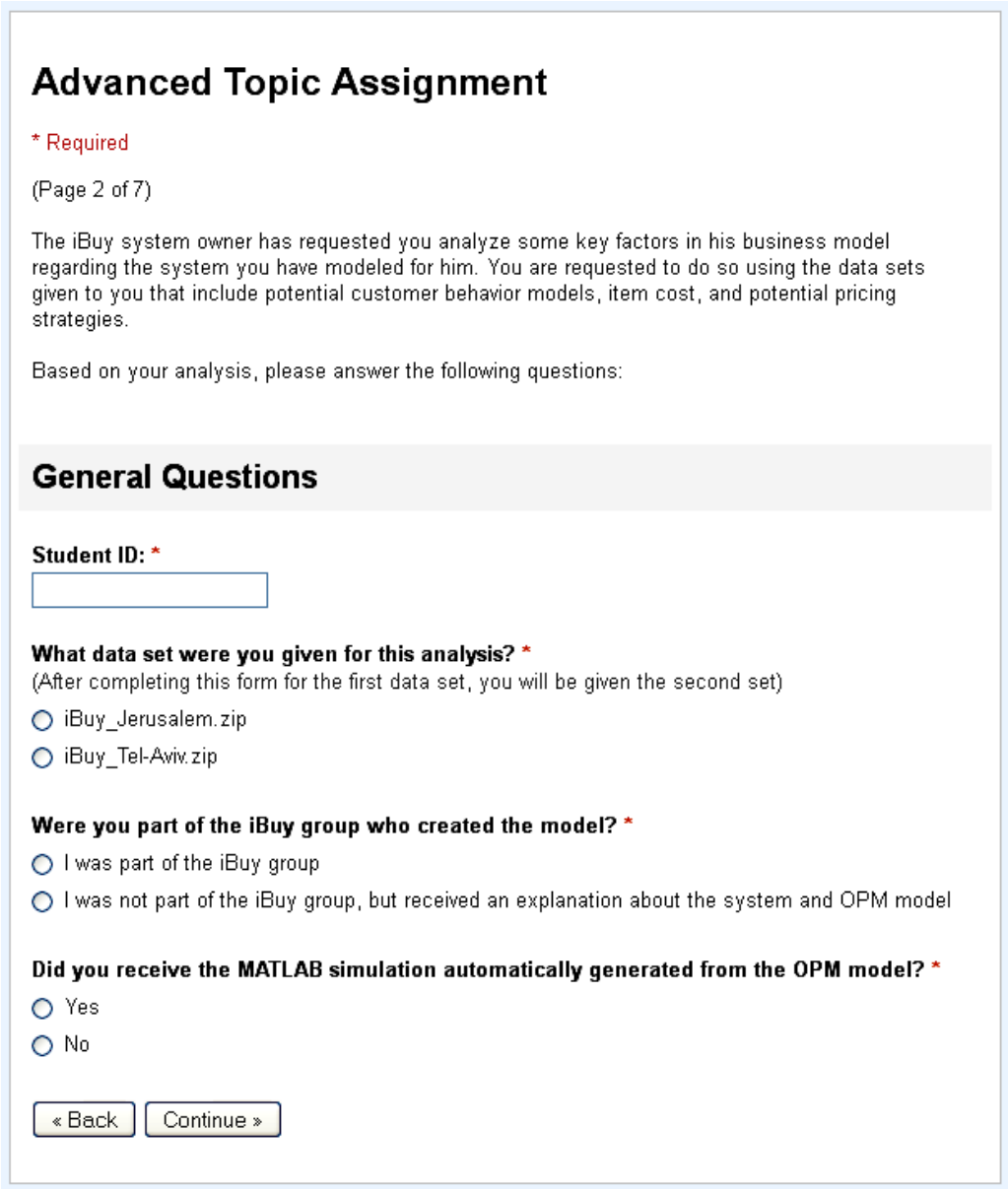
Note: your grade for this assignment will not be based on the correctness of you answers, but rather on the analysis process you present and creativity in achieving the answers.
In addition to filling this form, you are requested to submit any material used in your analysis.

Good luck!



Continue »

Figure 27: AUTOMATLAB evaluation first page



Advanced Topic Assignment

* Required

(Page 2 of 7)

The iBuy system owner has requested you analyze some key factors in his business model regarding the system you have modeled for him. You are requested to do so using the data sets given to you that include potential customer behavior models, item cost, and potential pricing strategies.

Based on your analysis, please answer the following questions:

General Questions

Student ID: *

What data set were you given for this analysis? *
(After completing this form for the first data set, you will be given the second set)

- iBuy_Jerusalem.zip
- iBuy_Tel-Aviv.zip

Were you part of the iBuy group who created the model? *

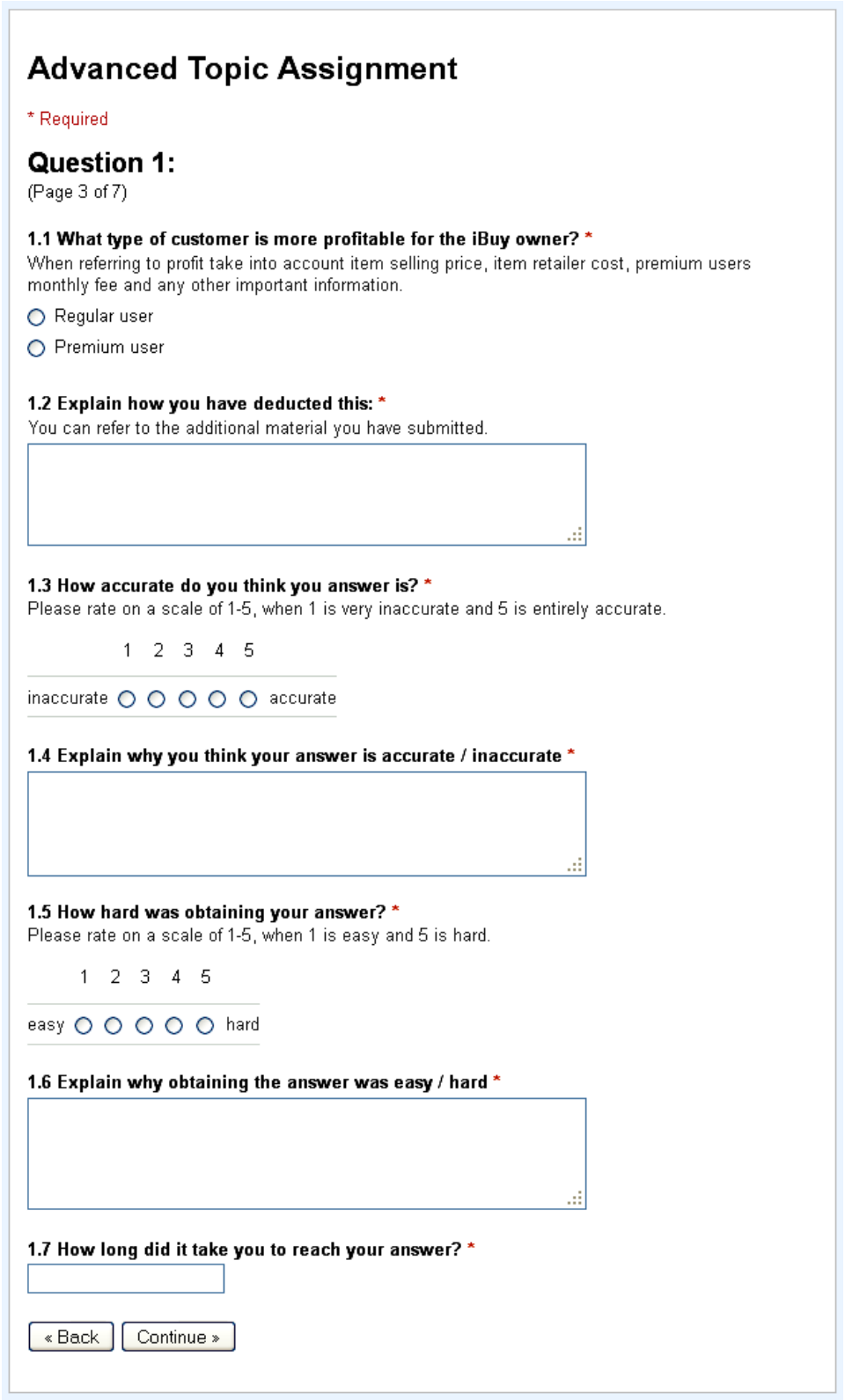
- I was part of the iBuy group
- I was not part of the iBuy group, but received an explanation about the system and OPM model

Did you receive the MATLAB simulation automatically generated from the OPM model? *

- Yes
- No

« Back Continue »

Figure 28: AUTOMATLAB evaluation second page



Advanced Topic Assignment

* Required

Question 1:

(Page 3 of 7)

1.1 What type of customer is more profitable for the iBuy owner? *

When referring to profit take into account item selling price, item retailer cost, premium users monthly fee and any other important information.

- ○ Regular user
- ○ Premium user

1.2 Explain how you have deducted this: *

You can refer to the additional material you have submitted.

1.3 How accurate do you think you answer is? *

Please rate on a scale of 1-5, when 1 is very inaccurate and 5 is entirely accurate.

	1	2	3	4	5	
inaccurate	○	○	○	○	○	accurate

1.4 Explain why you think your answer is accurate / inaccurate *

1.5 How hard was obtaining your answer? *

Please rate on a scale of 1-5, when 1 is easy and 5 is hard.

	1	2	3	4	5	
easy	○	○	○	○	○	hard

1.6 Explain why obtaining the answer was easy / hard *

1.7 How long did it take you to reach your answer? *

« Back | Continue »

Figure 29: AUTOMATLAB evaluation third page

The students were given data sets containing a list of items sold by the **Web Based Grocery Shopping** system (see Figure 30), monthly fee for premium users, a list of customers, and potential shopping lists for customers (see Figure 31).

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

Figure 30: A partial list of items sold by the **Web-based Grocery Shopping** system

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

Figure 31: A partial shopping list for customer 218207544

The complete questionnaire and data sets are provided in appendix E.

Identical questionnaires and data sets were given both to the experimental and to the control groups. The control group was instructed to answer the questions using whatever tool they desire. The experimental group students received the automatically-generated MATLAB code from the AUTOMATLAB approach shown in Figure 32 through Figure 37, and were instructed to use it in order to answer the four questions. The MATLAB code was intended to help them gain better understanding of the model and to serve as a basis for a simulation of the system, including computational and stochastic aspects required to answer the four questions.

```
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
```

Figure 32: MATLAB code generated from the process **Shopping List Creating**

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


% [] = RegisteredUser; % RegisteredUser handles ItemChoosing.

ChosenItem = 1; % ItemChoosing yields ChosenItem.

end
```

Figure 33: MATLAB code generated from the process **Item Choosing**

```
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
```

Figure 34: MATLAB code generated from the process **High Price Generating**

```
function [ItemPrice] = LowPriceGenerating(ChosenItem,UserEntity)
% LowPriceGenerating occurs if ChosenItem is in existent and UserEntity is Premium user.
% LowPriceGenerating yields Low ItemPrice.



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

% [] = UserEntity; % LowPriceGenerating occurs if UserEntity is Premium user

ItemPrice = 'Low'; % LowPriceGenerating yields Low ItemPrice.

end
```

Figure 35: MATLAB code generated from the process **Low Price Generating**

```
function [TotalCost] = ShoppingCostUpdating(ItemPrice,TotalCost)
% ShoppingCostUpdating requires ItemPrice.
% ShoppingCostUpdating affects TotalCost.


% [] = ItemPrice; % ShoppingCostUpdating requires ItemPrice.

[TotalCost] = TotalCost; % ShoppingCostUpdating affects TotalCost.

end
```

Figure 36: MATLAB code generated from the process **Shopping Cost Updating**

```
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
```

Figure 37: MATLAB code generated from the process **Product Updating**

The students from the experimental group who received the AUTOMATLAB generated MATLAB code submitted their code when answering the questionnaire. An example of the **Shopping List Creating** enhanced code is presented in Figure 38. A complete example of MATLAB code expanded from the original AUTOMATLAB code used to obtain answers for the evaluation is found in Appendix F.

```
function [TotalCost,TotalCostForPremuim,TotalRetailerCost,ItemsList,...
    TotalPurchasedProducts] = ShoppingListCreating(RegisteredUser,UserEntity,...
    ItemsList, IsNewPremiumUser)

% ProductList = zeros(length(RegisteredUser),2);
% ProductListIndex = 0;
RemainingDesiredItems = 'True';
itemIndex = 0;
TotalCost = 0;
TotalCostForPremuim = 0;
TotalRetailerCost = 0;
TotalPurchasedProducts = 0;

while isequal(RemainingDesiredItems,'True')
    % ShoppingListCreating occurs if RemainingDesiredItems is True.
    itemIndex = itemIndex+1;

    if ~isempty(RegisteredUser)
        [ChosenItem] = ItemChoosing(RegisteredUser, itemIndex);
    end

    [ItemPrice, ItemPriceWithDiscount,itemRetailerCost,ItemIndexInList] =...
        PriceGenerating(ChosenItem,ItemsList);

    if ~isempty(ChosenItem) && isequal(UserEntity,'Regular user')
        %update amount of buyed products
        TotalPurchasedProducts = TotalPurchasedProducts + ChosenItem(2);
        [ItemsList] = UpdateTotalBuyedProducts( ItemsList, ItemIndexInList,...
            ChosenItem(2), UserEntity, IsNewPremiumUser, 1 );
    end

    if ~isempty(ChosenItem) && isequal(UserEntity,'Premium user')
         IsBuying = IsBuyingProduct(ItemPrice, ItemPriceWithDiscount );
         if (IsBuying == 1)
             TotalPurchasedProducts = TotalPurchasedProducts + ChosenItem(2);
             ItemPrice = ItemPriceWithDiscount;
         else
             ItemPrice = 0;
             itemRetailerCost = 0;
         end
          [ItemsList] = UpdateTotalBuyedProducts( ItemsList, ItemIndexInList,...
              ChosenItem(2), UserEntity, IsNewPremiumUser, IsBuying );
    end

    if ~isempty(ItemPrice) && ~isempty(TotalCost)
        [TotalCost] = ShoppingCostUpdating(ItemPrice,TotalCost);
        [TotalCostForPremuim] = ShoppingCostUpdating(ItemPriceWithDiscount,...
            TotalCostForPremuim); %low price for ALL the shopping list
        TotalRetailerCost = TotalRetailerCost + itemRetailerCost;
    end

    % ProductUpdating invokes ShoppingListCreating.
    if (itemIndex>= length(RegisteredUser))
        RemainingDesiredItems = 'False';
    end
end

end
```

Figure 38: **Shopping List Creating** enhanced code from AUTOMATLAB evaluation

5.4. Data analysis and evaluation results

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

The amount of questionnaires submitted from each group is presented in Table 5.

Table 5: Amount of questionnaires submitted from each group

	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

We analyze the data from the evaluation for the three variables: *group* (experimental or control), *level* ('Jerusalem' – level-1 easy data set or 'Tel-Aviv' – level-2 difficult data set) and *question* (Q1, Q2, Q2, or Q4) 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. Independent t-test and one-way ANOVA with a Bonferroni correction served as our post-hoc tests, where it was needed.

First hypothesis analysis

Our first hypothesis was that users of AUTOMATLAB will gain deeper, more accurate understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB. In line with this hypothesis, we found a significant main effect for *group* ($F(1, 10) = 5.23$, $p < .05$, $\eta^2 = .34$), indicating that students who used AUTOMATLAB scored higher ($M = .71$, $SD = .05$) than students who answered the questions without using AUTOMATLAB ($M = .55$, $SD = .05$). Likewise, there was a significant main effect for *level* ($F(1,10) = 5.99$, $p < .05$, $\eta^2 = .37$) which means that the level-1 dataset yield higher grades ($M = .72$, $SD = .06$) than level-2 ($M = .54$, $SD = .04$). Finally, we found a significant main effect for *question* ($F(3, 30) = 6.15$, $p < .01$, $\eta^2 = .38$). Post-hoc analysis using Bonferroni correction revealed that Q4's grades ($M = .29$, $SD = .10$) were significantly lower than the grades of Q1 ($M = .81$, $SD = .07$) and Q2 ($M = .82$, $SD = .05$). The grades of Q3 ($M = .60$, $SD = .12$) did not differ significantly from the rest of the questions.

Significant interaction was found between *group* and *level* ($F(1,10) = 4.88$, $p < .05$, $\eta^2 = .33$). Continued tests found that this difference is due to the interaction between *group* and level-2 ($F(1,10) = 16.77$, $p < .01$, $\eta^2 = .62$). The difference between the experimental and control group is due mainly to Q2 ($t(10) = 3.04$, $p < .01$)) and Q3 ($t(10) = 1.86$, $p < .05$)), indicating that the grades of the experimental group on question 2 ($M = 1$, $SD = 0$) were higher than those of the control group ($M = .55$, $SD = .39$), and that the grades of the experimental group in question 3 ($M = .80$, $SD = .45$) were higher than the grades of the control group ($M = .29$, $SD = .49$).

Second hypothesis

Our second hypothesis was that AUTOMATLAB users will understand the system's computational and quantitative aspects quicker than users who used OPM without AUTOMATLAB.

The main effect for *group* ($F(1,10) = .39$, $P > .05$, $\eta^2 = .04$) was not significant, indicating that the experimental ($M = 2.02$, $SD = .53$) and control groups ($M = 1.59$, $SD = .45$) did not significantly differ in the time required to solve the questions.

According to the hypothesis there was a significant main effect for *question* ($F(3, 30) = 11.87$, $p < .001$, $\eta^2 = .54$). Post-hoc analysis using Bonferroni correction revealed that the time needed to achieve an answer for Q1 ($M = 3.4$, $SD = .56$) was significantly longer than the time needed for Q2 ($M = 1.43$, $SD = .37$), Q3 ($M = 1.12$, $SD = .39$) and Q4 ($M = 1.29$, $SD = .42$).

Third Hypothesis

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

The main effect for *group* ($F(1,10) = .62$, $P > .05$, $\eta^2 = .06$) was not significant, indicating that the experimental and control groups did not significantly differ in the confidence they had in the accuracy of their results.

These test results indicate in addition that there was a significant main effect for *question* ($F(3, 30) = 7.14$, $p = .001$, $\eta^2 = .42$). Post-hoc analysis using Bonferroni correction revealed that confidence in answers for Q1 ($M = 4.41$, $SD = .15$) were significantly higher than the grades for Q3 ($M = 3.36$, $SD = .22$) and Q4 ($M = 3.70$, SD

= .23). Confidence in answers of Q2 (M = 3.96, SD = .18) did not differ significantly from the rest of the questions.

Fourth Hypothesis

Our fourth hypothesis was that AUTOMATLAB users will understand the system's computational and quantitative aspects better, with less difficulty, than who used OPM without AUTOMATLAB. The multi-way repeated measures test revealed that the main effect for *group* ($F(1, 10) = 4.00$, $p = .07$, $\eta^2 = .29$) has borderline significance. Since our hypothesis is one-tailed, we can deduct a significant difference between the groups, indicating that students who did not use AUTOMATLAB indicated a higher level of difficulty (M = 3.27, SD = .25) than students who did use AUTOMATLAB (M = 2.5, SD = .29), suggesting that the experimental group subjectively experienced a lower level of difficulty than the control group when solving the questions.

The main effect for *level* ($F(1,10) = .59$, $P > .05$, $\eta^2 = .06$) was not significant, but significant interaction was found between *group* and *level* ($F(1,10) = 14.49$, $p < .005$, $\eta^2 = .59$). Follow-up tests revealed significant interaction between *group* and level-2 ($t(10) = 3.09$, $p \leq .01$) resulting from difference between groups for Q2 in level-2 ($t(10) = 2.71$, $p < .05$) and difference for Q4 in level-2 ($t(10) = 2.72$, $p < .05$) , as can be seen in Table 6.

Table 6: Results of Continued tests for interaction between *group* and *level*

		Experimental group		Control group		
	question	***mean***	***std***	***mean***	***std***	***t(10)***
	Q1	4.20	1.30	3.00	.82	1.972[a]
	Q2	2.80	.84	3.29	1.11	.82
level-1	Q3	2.20	1.10	2.43	1.27	.32
	Q4	2.20	.84	3.43	.53	3.12*
	Total	**11.40**	**3.29**	**12.14**	**2.12**	**.48**
	Q1	2.60	1.34	3.71	1.11	1.57
	Q2	1.40	.55	3.14	1.35	2.71*
level-2	Q3	2.60	1.82	3.57	.98	1.21
	Q4	2.00	.71	3.57	1.13	2.72*
	Total	**8.60**	**3.78**	**14.0**	**2.31**	**3.09***

* $P < .05$

[a] Borderline significance was found, for apposite of hypothesis.

5.5. Evaluation results and discussion

Our first hypothesis was that users of AUTOMATLAB will gain deeper, more accurate understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB. The results indeed show that the experimental group achieved higher accuracy levels than the control group in our experiment. It can be noticed the accuracy increased for the experimental group more significantly for the more complicated data set (for questions 2 and 3), suggesting that the benefits of AUTOMATLAB are more prominent for more complex situations and needs.

Analysis of the explanations provided by the students when submitting their answers suggests that the students in the experimental group attempted to create a more accurate simulation of the system behavior: *"The calculation is preformed in the MATLAB code… For every customer I calculate the profit… Dealing with a customer is one iteration of the function 'ShopingListCreating'… The simulation results show…"* Students in the control group used other method (mainly simple Excel spreadsheets or pen-and-paper calculations), ignoring seemingly unnecessary aspects: *"Didn't compare one by one…"*, *"Response was relatively difficult since lots of considerations came in and I needed to make assumptions"*. This is assumed to be the cause for the difference in accuracy between their answers.

Our second hypothesis was that AUTOMATLAB users will understand the system's computational and quantitative aspects quicker than users who used OPM without AUTOMATLAB. The results did not show a conclusive difference between the experimental and control groups. We have seen that the time needed to answer Question 1 was significantly longer than the time needed for questions 2 through 4. For the experimental group, we assume that the longer time required to answer the first question is due to the need to enhance the automatically generated MATLAB code when solving the first question, while this code is later used to solve the rest of the questions, as can be seen in student comments: *"Calculations were very similar to previous questions…"*, *"The code was already prepared – only one function needed to be changed"*.

Our third hypothesis was that AUTOMATLAB users will be more confident in their understanding of the system's computational and quantitative aspects than users who used OPM without AUTOMATLAB. The results received for this hypothesis were not conclusive. No clear differences were found for the different groups or datasets. Since the experimental

group students had limited level of experience with MATLAB, perhaps some of their lack of confidence was due to the tool being used and not due to lack of understanding of the model. This may be researched in future work.

Our fourth hypothesis was that AUTOMATLAB users will understand the system's computational and quantitative aspects better, with less difficulty, than who used OPM without AUTOMATLAB. The results indeed show that the students who did not use AUTOMATLAB rated their difficulty as higher.

Analysis of the students' explanations suggests that the difficulty in the control group is associated mainly with the challenge in representing the customer behavior model in simple tools like Excel or hand written calculations: *"It took a long time to get the calculations since I didn't know an appropriate action to do so in Excel."*, *"I had to go over every buyer and every product which is a lot of intersections!!!"*. Explanations from the experimental group in questions 2-4 repeatedly mentioned using the previously created code as a reason for low difficulty*: "No changes were needed from previous…", "From the way I implemented the solution of the answer in the first part… no more changes had to be made"*, supporting our hypothesis.

From both the statistical and qualitative analysis of the evaluation we have seen that the AUTOMATLAB method does indeed benefit the user in several ways. AUTOMATLAB may improve the accuracy of understanding a system's quantitative aspects and may decreas the difficulty of reaching such understanding. Results regarding the time needed to understand the quantitative aspects and user's confidence regarding his understanding are not significant, probably due to the lack of participants' experience with the MATLAB environment. This aspect should be tested with more proficient participants in a future research.

6. Conclusion and Future Research

This research has tackled 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 are weak or missing altogether. Other modeling and simulation methods that do provide the computational aspects generally lack the high level of abstraction required from a conceptual modeling language.

The computational simplification problem defined in this work relates to the difficulty of incorporating quantitative aspects into conceptual models, which focus on the qualitative aspects of the system. We presented two possible solutions for this problem, based on expanding OPM to combine MATLAB and Simulink.

In the first, AUTOMATLAB approach, we aim to solve the computational simplification problem by adding a parallel MATLAB-based representation of the OPM model, which can be augmented with any desired computational aspects in the MATLAB representation. A major advantage of the AUTOMATLAB approach is that the OPM model becomes an integral part of the augmented MATLAB model, and thus it can evolve and serve for increasingly more quantitative-oriented simulations in downstream lifecycle stages of the system as it is being developed or researched.

In the second, OPM/CS approach, we aim to solve the computational simplification problem by augmenting the capabilities of the OPM in-zooming mechanism. The additional capability provides for the content of an in-zoomed process to be replaced by a MATLAB function or Simulink model containing any necessary computational aspects. The main advantages of the OPM/CS approach are (1) the simplicity of the enhancement that uses an intuitive extension of the OPM in-zooming mechanism, and (2) the preservation of the original OPM conceptual model with its semantics.

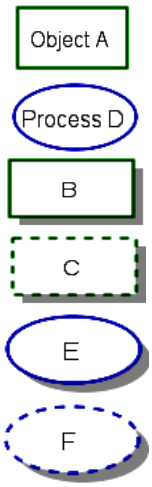
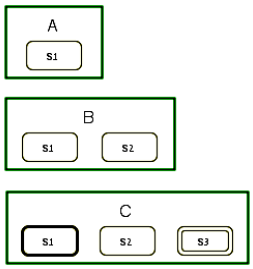
The two approaches were demonstrated via examples and case studies. A more thorough evaluation of the AUTOMATLAB approach was conducted with human subjects, showing benefits of this approach in terms of better system understanding. Results regarding user confidence in system understanding and time required to achieve such understanding where not conclusive, perhaps due to the small sample. The statistical results were supported by qualitative content analysis of the subjects' responses to questions.

Both approaches have been designed and implemented with forward compatibility to the future online OPM CASE tool, WebOPCAT, and partial compatibility to the development

version of the current OPCAT. When the WebOPCAT tool will be completed, a large scale test and comparison of AUTOMATLAB and OPM/CS should be preformed.

Appendix A – OPM Summary

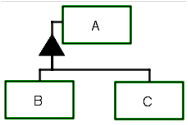
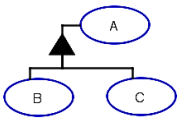
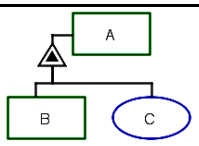
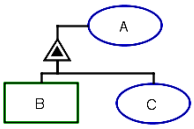
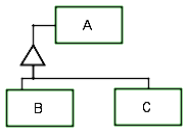
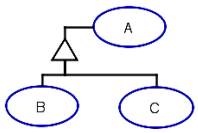
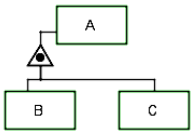
1. Entities

Name		Symbol	OPL	Definitions
Things	**Object** **Process**		**B is physical.** *(shaded rectangle)* **C is physical and environmental.** *(shaded dashed rectangle)* **E is physical.** *(shaded ellipse)* **F is physical and environmental.** *(shaded dashed ellipse)*	An **object** is a thing that exists. A **process** is a thing that transforms at least one object. Transformation is object generation or consumption, or effect—a change in the state of an object.
State			**A is s1.** **B can be s1 or s2.** **C can be s1, s2, or s3.** **s1 is initial.** **s3 is final.**	A **state** is situation an object can be at or a value it can assume. States are always within the object that owns them. A state can be initial, final, or both.

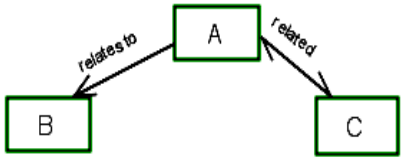
2. Structural Links

Symbol	Name	OPL	Allowed Source-to-Destination connections	Semantics/ Effect on the system flow/ Comments
	Aggregation-Participation	**A** consist of **B**.	Object-Object Process- Process	Whole -Part
	Exhibition-Characterization	**A** exhibits **B**.	Object-Object Object-Process Process-Object Process- Process	
	Generalization-Specialization	**B** is an **A**. (objects) **B** is **A**. (processes)	Object-Object Process- Process	
	Classification-Instantiation	**B** is an instance of **A**.	Object-Object Process- Process	
	Tagged structural links: Unidirectional Bidirectional	According to text added by user	Object-Object Process- Process	Describes structural information.

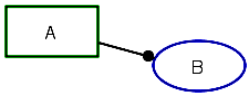
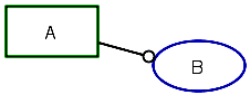
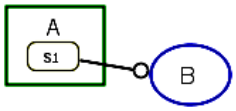
3. Fundamental Structural Links

Name	Symbol	OPL	Semantics
Aggregation-Participation 		**A** consists of **B** and **C**.	A is the whole, B and C are parts.
		**A** consists of **B** and **C**.	
Exhibition-Characterization 		**A** exhibits **B**, as well as **C**.	Object B is an attribute of A and process C is its operation (method). A can be an object or a process.
		**A** exhibits **B**, as well as **C**.	
Generalization-Specialization 		**B** is an **A**. **C** is an **A**.	A specializes into B and C. A, B, and C can be either all objects or all processes.
		**B** is **A**. **C** is **A**.	
Classification-Instantiation 		**B** is an instance of **A**. **C** is an instance of **A**.	Object A is the class, for which B and C are instances. Applicable to processes too.

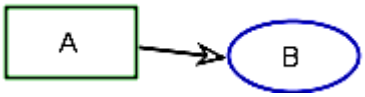
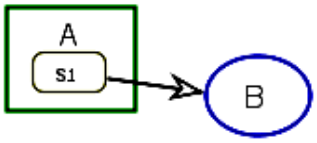
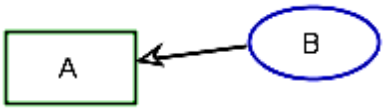
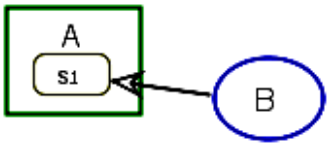
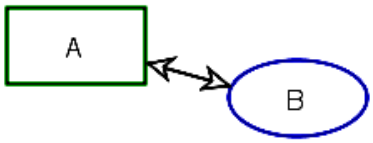
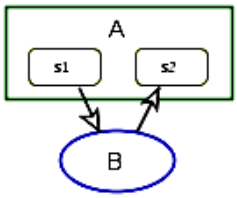
4. Tagged Structural Links

Name	Symbol	OPL	Semantics
Unidirectional & bidirectional tagged structural links		**A** relates to **B**. (for unidirectional) **A** and **C** are related. (for bidirectional)	A user-defined textual tag describes any structural relation between two objects or between two processes.

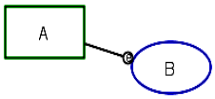
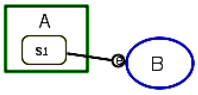
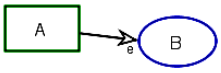
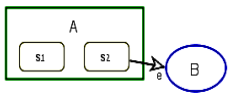
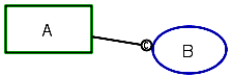
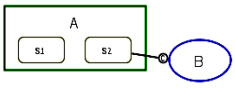
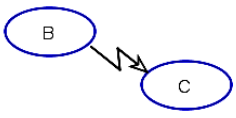
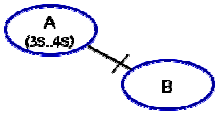
5. Procedural Enabling Links

Name	Symbol	OPL	Semantics
Agent Link		**A** handles **B**.	Denotes that object A is a human operator who triggers process B.
Instrument Link		**B** requires **A**.	"Wait until" semantics: Process B cannot happen if object A does not exist.
State-Specified Instrument Link		**B** requires **s1** **A**.	"Wait until" semantics: Process B cannot happen if object A is not at state s1.

6. Procedural Transforming Links

Name	Symbol	OPL	Semantics
Consumption Link		**B** consumes **A.**	Process B consumes Object A.
State-Specified Consumption Link		**B** consumes **s1 A.**	Process B consumes Object A when it is at State s1.
Result Link		**B** yields **A**.	Process B creates Object A.
State-Specified Result Link		**B** yields **s1 A**.	Process B creates Object A at State s1.
Effect Link		**B** affects **A**.	Process B changes the state of Object A; the details of the effect may be added at a lower level.
State-Specified Effect Link (*Input-Output Links Pair*)		**B** changes **A** from **s1** to **s2**.	Process B changes the state of Object A from State s1 to State s2.

7. Procedural Links: Control Links

Name	Symbol	OPL	Semantics
Instrument Event Link		**A** triggers **B**. **B** requires **A**.	Generation of object A is an event that triggers process B. B will start executing if its precondition is met. Since A is instrument it will not be affected by B.
State-Specified Instrument Event Link		A triggers **B**. when it enters **s1**. **B** requires **s1** **A**.	Entering state s1 of object A is an event that triggers process B. B will start executing if its precondition is met. Since A is instrument it will not be affected by B.
Consumption Event Link		**A** triggers **B**. **B** consumes **A**.	Generation of object A is an event that triggers process B. B will start executing if its precondition is met, and if so it will consume A.
State-Specified Consumption Event Link		**A** triggers **B** when it enters **s2**. **B** consumes **s2** **A**.	Entering state s2 of A is an event that triggers process B. If B is triggered, it will consume A. B will start executing if its precondition is met, and if so it will consume A.
Condition Link		**B** occurs if **A** exists.	Existence of object A is a condition for the execution of B. If A does not exist, then B is skipped and regular system flow continues.
State-Specified Condition Link		**B** occurs if **A** is **s1**.	Existence of object A at state s2 is a condition for the execution of B. If A is not in s2, then B is skipped and regular system flow continues.
Invocation Link		**B** invokes **C**.	Execution termination of process B is an event that triggers process C. B yields a temporary object that is immediately consumed by C and therefore not be shown explicitly in the model.
Exception Link		**A** triggers **B** when it lasts more than 4 seconds.	Process A has to be assigned with maximal acceptable time duration, which, if exceeded, triggers process B.

Appendix B – AUTOMATLAB Layer Auto Generator Code

```
function [] = OPMML_Auto_generator()
%%
clear all;
clc;

[TempStatement,TYPES] = MakeCellArr;

n = length(TempStatement(:,1));
m = length(TempStatement(1,:));

% replacing ~ ! @ # $ % ^ & * ( ) { } [ ] < > . \ / - with _
for i = 1:1:n
    for j = 1:1:m
        if ~isequal(TempStatement(i,j),{[]}) && ~(TYPES(i,j)=='c')
            Temp = char(TempStatement(i,j));
            while sum(['~ ! @ # $ % ^ & * ( ) { } [ ] < > . \ / -'] == Temp(1))
                Temp = Temp(2:1:end);
            end
            for q = 2:1:length(Temp)
                if sum(['~ ! @ # $ % ^ & * ( ) { } [ ] < > . \ / -'] == Temp(q))
                    Temp(q) = '_';
                    TempStatement(i,j) = {Temp};
                end
            end
        end
    end
end


%% list of processes & objects
% list of processes:
for i = 1:1:n
    for j = 1:1:m
        if TYPES(i,j)=='p'
            procs(1,j+m*(i-1)) = TempStatement(i,j);
        end
    end
end

i = 1;
while i<length(procs)
    if isequal(procs(i),{[]})
        procs = procs([1:1:i-1,i+1:1:end]);
    else
        i = i+1;
    end
end

procs = Distinct(procs)';

% list of objects:
for i = 1:1:n
    for j = 1:1:m
        if TYPES(i,j)=='o'
            Objs(1,j+m*(i-1)) = TempStatement(i,j);
        end
    end
end

i = 1;
while i<length(Objs)
```

```
    if isequal(Objs(i),{[]})
        Objs = Objs([1:1:i-1,i+1:1:end]);
    else
        i = i+1;
    end
end


Objs = Distinct(Objs)';

%% Process 2 Process relations
% Proccess call matrix
P2P = zeros(length(procs),length(procs));
for l = 1:1:n
    if (isequal(TempStatement(l,2),{'consistsof'}) ||
isequal(TempStatement(l,2),{'zoomsinto'})) && TYPES(l,1)=='p'
        for i = 1:1:m
            for t = 1:1:length(procs)
                for g = 1:1:length(procs)
                    if isequal(TempStatement(l,i),procs(t)) &&
isequal(TempStatement(l,1),procs(g)) && ~(g==t)
                        P2P(g,t) = 1;
                    end
                end
            end
        end
    end
end

%% Object 2 Object relations
O2O = zeros(length(Objs),length(Objs));
for l = 1:1:n
    if isequal(TempStatement(l,2),{'consistsof'}) && TYPES(l,1)=='o'
        for i = 1:1:m
            for t = 1:1:length(Objs)
                for g = 1:1:length(Objs)
                    if isequal(TempStatement(l,i),Objs(t)) &&
isequal(TempStatement(l,1),Objs(g)) && ~(g==t)
                        O2O(g,t) = 1;
                    end
                end
            end
        end
    end
end

%% Process 2 Object relations
P2O = zeros(length(procs),length(Objs));
ChangeFrom = {[]};
for l = 1:1:n
    for i = 1:1:m
        for t = 1:1:length(Objs)
            for g = 1:1:length(procs)
                if TYPES(l,1)=='p' && isequal(TempStatement(l,i),Objs(t)) &&
isequal(TempStatement(l,1),procs(g))

                    switch char(TempStatement(l,2))
                        case 'exhibits'
                            P2O(g,t) = P2O(g,t)+1;

                        case 'consumes'
                            P2O(g,t) = P2O(g,t)+2;

                        case 'zoomsinto'
                            P2O(g,t) = P2O(g,t)+4;
```

```
                        case 'yields'
                            P2O(g,t) = P2O(g,t)+8;

                        case 'requires'
                            P2O(g,t) = P2O(g,t)+16;

                        case 'changes'
                            P2O(g,t) = P2O(g,t)+32;
                            [ni,nj] = size(ChangeFrom);
                            for q = 3:1:m-2
                                if TYPES(l,q)=='v' && TYPES(l,q+2)=='v' %&& (g>ni
|| t>nj || isequal(ChangeFrom(g,t),{[]}))
                                    ChangeFrom(g,t) = TempStatement(l,q);
                                    ChangeTo(g,t) = TempStatement(l,q+2);
                                    TYPES(l,q)='x';
                                    TYPES(l,q+2)='x';
                                    break;
                                end
                            end

                        case 'affects'
                            P2O(g,t) = P2O(g,t)+64;

                        otherwise
                            fprinf('Error - unknown realtion in P20 building');
                    end

                end
            end
        end
    end
end

%% CREATING THE FILES:
for i = 1:1:length(P2P)
    FuncDef(i) = {['] = ' char(procs(i)) '()']};
    FuncBod(i) = {[]};
    FuncEnd(i) = {[]};
    FuncCom(i) = {[]};

    for j = 1:1:length(P2O)
        %% consume
        if ContainCheck(P2O(i,j),2) % need to be in () and [] - consume
            NewLine = [char(FuncDef(i))];
            if NewLine(end-1)=='('
                NewLine = [NewLine(1:1:end-1) char(Objs(j)) NewLine(end)];
            else
                NewLine = [NewLine(1:1:end-1) ',' char(Objs(j)) NewLine(end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            NewLine = [char(FuncDef(i))];
            t = 1;
            while ~(NewLine(t)==']')
                t = t+1;
            end
            if t==2
                NewLine = [NewLine(1) char(Objs(j)) NewLine(2:1:end)];
            else
                NewLine = [NewLine(1:1:t-1) ',' char(Objs(j)) NewLine(t:1:end)];
            end
            FuncDef(i) = {[char(NewLine)]};
```

```
            if isequal(FuncCom(i),{[]})
                AddLine = [];
            else
                AddLine = char(FuncCom(i));
            end
            AddLine = [AddLine '%%' char(procs(i)) ' consumes ' char(Objs(j))
'\n'];
            FuncCom(i) = {AddLine};

            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            NewLine = [NewLine '%% ' char(Objs(j)) '; %%' char(procs(i)) ' consumes
' char(Objs(j)) '\n'];
            FuncBod(i) = {[char(NewLine)]};

            if isequal(FuncEnd(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncEnd(i));
            end
            NewLine = [NewLine char(Objs(j)) ' = []; ' '%%' char(procs(i)) '
consumes ' char(Objs(j)) '\n'];
            FuncEnd(i) = {[char(NewLine)]};
        end
        %% require
        if ContainCheck(P2O(i,j),16) % need to be in () - require
            NewLine = [char(FuncDef(i))];
            if NewLine(end-1)=='('
                NewLine = [NewLine(1:1:end-1) char(Objs(j)) NewLine(end)];
            else
                NewLine = [NewLine(1:1:end-1) ',' char(Objs(j)) NewLine(end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            if isequal(FuncCom(i),{[]})
                AddLine = [];
            else
                AddLine = char(FuncCom(i));
            end
            AddLine = [AddLine '%%' char(procs(i)) ' requires ' char(Objs(j))
'\n'];
            FuncCom(i) = {AddLine};

            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            NewLine = [NewLine '%% ' char(Objs(j)) '; %%' char(procs(i)) ' requires
' char(Objs(j)) '\n'];
            FuncBod(i) = {[char(NewLine)]};
        end
        %% yields
        if ContainCheck(P2O(i,j),8) % need to be in []= - yields
            NewLine = [char(FuncDef(i))];
            t = 1;
            while ~(NewLine(t)==']')
                t = t+1;
            end
            if t==2
                NewLine = [NewLine(1) char(Objs(j)) NewLine(2:1:end)];
            else
                NewLine = [NewLine(1:1:t-1) ',' char(Objs(j)) NewLine(t:1:end)];
```

```
            end
            FuncDef(i) = {[char(NewLine)]};

            if isequal(FuncCom(i),{[]})
                AddLine = [];
            else
                AddLine = char(FuncCom(i));
            end
            AddLine = [AddLine '%%' char(procs(i)) ' yields ' char(Objs(j)) '\n'];
            FuncCom(i) = {AddLine};

            if ~UnDirectYield(i,i,j,P2O,P2P)
                if isequal(FuncEnd(i),{[]})
                    NewLine = [];
                else
                    NewLine = char(FuncEnd(i));
                end
                NewLine = [NewLine char(Objs(j)) ' = 1; ' '%%' char(procs(i)) '
yields ' char(Objs(j)) '\n'];
                FuncEnd(i) = {[char(NewLine)]};
            end
        end
        %% affects
        if ContainCheck(P2O(i,j),64) % need to be in () and [] - affects
            NewLine = [char(FuncDef(i))];
            if NewLine(end-1)=='('
                NewLine = [NewLine(1:1:end-1) char(Objs(j)) NewLine(end)];
            else
                NewLine = [NewLine(1:1:end-1) ',' char(Objs(j)) NewLine(end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            NewLine = [char(FuncDef(i))];
            t = 1;
            while ~(NewLine(t)==']')
                t = t+1;
            end
            if t==2
                NewLine = [NewLine(1) char(Objs(j)) NewLine(2:1:end)];
            else
                NewLine = [NewLine(1:1:t-1) ',' char(Objs(j)) NewLine(t:1:end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            if isequal(FuncCom(i),{[]})
                AddLine = [];
            else
                AddLine = char(FuncCom(i));
            end
            AddLine = [AddLine '%%' char(procs(i)) ' affects ' char(Objs(j)) '\n'];
            FuncCom(i) = {AddLine};

            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            NewLine = [NewLine '%% ' char(Objs(j)) '; %%' char(procs(i)) ' affects
' char(Objs(j)) '\n'];
            FuncBod(i) = {[char(NewLine)]};

        end
        %% changes
        if ContainCheck(P2O(i,j),32) % need to be in () and [] - changes
            NewLine = [char(FuncDef(i))];
```

```
            if NewLine(end-1)=='('
                NewLine = [NewLine(1:1:end-1) char(Objs(j)) NewLine(end)];
            else
                NewLine = [NewLine(1:1:end-1) ',' char(Objs(j)) NewLine(end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            NewLine = [char(FuncDef(i))];
            t = 1;
            while ~(NewLine(t)==']')
                t = t+1;
            end
            if t==2
                NewLine = [NewLine(1) char(Objs(j)) NewLine(2:1:end)];
            else
                NewLine = [NewLine(1:1:t-1) ',' char(Objs(j)) NewLine(t:1:end)];
            end
            FuncDef(i) = {[char(NewLine)]};

            if isequal(FuncCom(i),{[]})
                AddLine = [];
            else
                AddLine = char(FuncCom(i));
            end
            AddLine = [AddLine '%%' char(procs(i)) ' changes ' char(Objs(j)) ' from
' char(ChangeFrom(i,j)) ' to ' char(ChangeTo(i,j)) '\n'];
            FuncCom(i) = {AddLine};

            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            NewLine = [NewLine char(Objs(j)) ' = ' char(ChangeTo(i,j)) ';' '%%'
char(procs(i)) ' changes ' char(Objs(j)) ' from ' char(ChangeFrom(i,j)) ' to '
char(ChangeTo(i,j)) '\n'];
            FuncBod(i) = {[char(NewLine)]};

        end

        %% exibets
        if ContainCheck(P2O(i,j),1) % need to be in function body - exibets.
initialize to empty
            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            NewLine = [NewLine char(Objs(j)) ' = []; ' '%%' char(procs(i)) '
exibits ' char(Objs(j)) '\n'];
            FuncBod(i) = {[char(NewLine)]};
        end
    end
end
%%
for i = 1:1:length(P2P) % Adding functions called by other functions
    for j = 1:1:length(P2P)
        if P2P(i,j)==1
            if isequal(FuncBod(i),{[]})
                NewLine = [];
            else
                NewLine = char(FuncBod(i));
            end
            AddLine = [char(FuncDef(j)) ';'];
            if AddLine(2)==']'
```

```
                AddLine = AddLine(6:1:end);
            end

            % find variables that are required, cosumed, affected or changed:
            ReqNCon = [];
            for t = 1:1:length(P2O)
                if ContainCheck(P2O(j,t),2) || ContainCheck(P2O(j,t),16) ||
ContainCheck(P2O(j,t),64)
                    if isempty(ReqNCon)
                        ReqNCon = ['~isempty(' char(Objs(t)) ')'];
                    else
                        ReqNCon = [ReqNCon ' && ~isempty(' char(Objs(t)) ')'];
                    end
                end
            end
            for t = 1:1:length(P2O)
                if ContainCheck(P2O(j,t),32)
                    [IsNum] = Char2Num(char(ChangeFrom(j,t)));
                    if IsNum
                        Equility = [char(Objs(t)) '==' char(ChangeFrom(j,t))];
                    else
                        Equility = ['isequal(' char(Objs(t)) ',''' 
char(ChangeFrom(j,t)) ''')'];
                    end

                    if isempty(ReqNCon)
                        ReqNCon = [Equility];
                    else
                        ReqNCon = [ReqNCon ' && ' Equility];
                    end

                end
            end
            if ~isempty(ReqNCon) % has a requirment, consume or affect
                FuncBod(i) = {[NewLine '\n' 'if ' ReqNCon '\n    ' AddLine ' %%' 
char(procs(i)) ' consists of ' char(procs(j)) '\n' 'end\n']};
            else
                FuncBod(i) = {[NewLine '\n' AddLine ' %%' char(procs(i)) ' consists
of ' char(procs(j)) '\n']};
            end
        end
    end
end


%% WRITE TO FILES
for i = 1:1:length(P2P)
    filename = [char(procs(i)) '.m'];
    fid = fopen(filename,'w'); % w - overwrite

    text2write = ['function ' char(FuncDef(i)) '\n'];
    if ~isequal(FuncCom(i),{[]})
        text2write = [text2write char(FuncCom(i)) '\n'];
    end
    if ~isequal(FuncBod(i),{[]})
        text2write = [text2write char(FuncBod(i)) '\n'];
    end
    if ~isequal(FuncEnd(i),{[]})
        text2write = [text2write char(FuncEnd(i))];
    end
    text2write = [text2write 'end'];

    fprintf(fid,text2write);
    fclose(fid);
end
```

```
%%
end

%%
function [TempStatement,TYPES] = MakeCellArr()

TXT = textread('Biology.htm','%c')';

% CHECKING OPENING HTML, HEADER and BODY TAGS
if ~strcmp(TXT(1:25),'<HTML><HEAD></HEAD><BODY>')
    fprintf('\nError - no <HTML><HEAD></HEAD><BODY> tag\n');
    return;
end

% Start proceding:
i = 26;
r = 1;
c = 1;

while i<length(TXT)-13
    while ~strcmp(TXT(i:i+6),'</font>')

        if strcmp(TXT(i:i+3),'<BR>')
            r = r+1;
            c = 1;
        end

        i = i+1;

        if i>length(TXT)-6
            return;
        end

    end

    j = i-1;
    while ~strcmp(TXT(j-1:j),'">')
        j = j-1;
    end
    TempStatement(r,c) = {TXT(j+1:i-1)};

    if strcmp(TXT(j-20:j),'<fontcolor="#000000">')
        TYPES(r,c) = 'c'; % Connector
    elseif strcmp(TXT(j-20:j),'<fontcolor="#000078">')
        TYPES(r,c) = 'p'; % Process
    elseif strcmp(TXT(j-20:j),'<fontcolor="#006d00">')
        TYPES(r,c) = 'o'; % Object
    elseif strcmp(TXT(j-20:j),'<fontcolor="#5b5b00">')
        TYPES(r,c) = 'v'; % Value
    else
        fprintf('\nError -  font color not recognized\n');
    end
    c = c+1;

    i = i+7;
end

end

%%
function [YesOrNo] = ContainCheck(A,b)
if b>A
    YesOrNo = 0;
```

```
    return;
end

YesOrNo = 1;
Factor = 1;
while Factor<=A
    Factor = Factor*2;
end

while Factor>=2
    Factor = Factor/2;
    if A >= Factor
        A = A - Factor;
    elseif Factor==b
        YesOrNo = 0;
    end
end

if ~A == 0
    fprintf('Error with ContainCheck');
end

end

%%
function [YesOrNo] = UnDirectYield(original_i,i,j,P2O,P2P)
if ~(i==original_i) && ContainCheck(P2O(i,j),8)
    YesOrNo = 1;
    return;
else
    CalledFuncs = P2P(i,:).*[1:1:length(P2P(i,:))];
    CalledFuncs = CalledFuncs(~(CalledFuncs==0));

    for q = CalledFuncs
        if UnDirectYield(original_i,q,j,P2O,P2P)==1
            YesOrNo = 1;
            return;
        end
    end

end
YesOrNo = 0;

end
%%
function [B] = Distinct(A)
B(1) = A(1);
for i = 2:1:length(A)
    In = 0;
    for j = 1:1:length(B)
        if isequal(A(i),B(j))
            In = 1;
        end
    end
    if ~In
        B(end+1) = A(i);
    end
end
end
%%
function [Y] = Char2Num(A)
Y = 1;
for i = 1:1:length(A)
    if A(i)<48 || A(i)>57
        Y = 0;
```

```
            return;
        end
    end
    end
```

Appendix C – OPM Computational Subcontractor GUI and Controller Code

OPM Computational Subcontractor GUI:

```
function varargout = OPM_CS_GUI_slim(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @OPM_CS_GUI_slim_OpeningFcn, ...
                   'gui_OutputFcn',  @OPM_CS_GUI_slim_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


% --- Executes just before OPM_CS_GUI_slim is made visible.
function OPM_CS_GUI_slim_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = OPM_CS_GUI_slim_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on button press in TempFiles_pushbutton.
function TempFiles_pushbutton_Callback(hObject, eventdata, handles)
SaveFolder = uigetdir;
set(handles.TempFiles_text, 'String', SaveFolder);

% --- Executes on button press in SimulinkMATLAB_pushbutton.
function SimulinkMATLAB_pushbutton_Callback(hObject, eventdata, handles)
SimFolder = uigetdir;
set(handles.SimulinkMATLABFils_text, 'String', SimFolder);

% --- Executes on button press in Run_pushbutton.
function Run_pushbutton_Callback(hObject, eventdata, handles)
Running = SimPlugin(get(handles.TempFiles_text,
'string'),get(handles.SimulinkMATLABFils_text,
'string'),get(handles.SimulationType_uipanel, 'UserData'),1);
if ~Running
    return;
end

% --- Executes when selected object is changed in SimulationType_uipanel.
function SimulationType_uipanel_SelectionChangeFcn(hObject, eventdata, handles)
if strcmp(get(hObject,'Tag'),'Simulink')
    set(hObject,'UserData',1)
else
    set(hObject,'UserData',0)
end

% --- Executes on button press in FuncLib_checkbox.
function FuncLib_checkbox_Callback(hObject, eventdata, handles)
```

OPM Computational Subcontractor Controller:

```
function Running = SimPlugin(TempFilesFolder,SimFolder,SimOrMat,LibOrNot);

% TempFilesFolder - Root for Temporary communication files
% SimFolder - Root for user created MATLAB or Simulink files
% SimOrMat - 1-Simulink (numeric only), 0-MATLAB
% LibOrNot - 1-read pre-defind files from MATLAB. 0-only user created files

Running = 1;
NewFolder = 'OPMSIMtemp';
[B1 B2 B3] = mkdir(TempFilesFolder,NewFolder);
clear B1;
clear B2;
clear B3;
Root = [TempFilesFolder '\' NewFolder '\'];

% Initialize Run.txt to '1'
TempRoot = [Root 'Run.txt'];
fid = fopen(TempRoot,'w+');
fprintf(fid,'1');
fclose(fid);

% Initialize Op2Mat.txt to '0' (this file will contain messages from Opcat
% - is set to 0 since counter starts at 0.
TempRoot = [Root 'Op2Mat.txt'];
fid = fopen(TempRoot,'w+');
fprintf(fid,'0');
fclose(fid);

% Create list of procedures acording to all *.mdl files in folder
if SimOrMat
    Temp1 = dir([SimFolder '/*.mdl']);
    TempRoot = [Root 'Procedures.txt'];
    fid = fopen(TempRoot,'w+');
    for i = 1:1:length(Temp1)
        Temp2 = Temp1(i);
        Temp3 = Temp2.name;
        fprintf(fid,'%s\r\n',Temp3(1:1:end-4));
    end
else
    Temp1 = dir([SimFolder '/*.m']);
    TempRoot = [Root 'Procedures.txt'];
    fid = fopen(TempRoot,'w+');
    for i = 1:1:length(Temp1)
        Temp2 = Temp1(i);
        Temp3 = Temp2.name;
        fprintf(fid,'%s\r\n',Temp3(1:1:end-2));
    end
end
fclose(fid);

% Start Simulation
KeepRunning = 1;
CurrentCounter = 0;
while KeepRunning
    % Wait for new message from OPCAT
    TempRoot = [Root 'Op2Mat.txt'];
    fid = fopen(TempRoot,'r');
    Temp1 = textscan(fid,'%d',1);
    Counter = cell2mat(Temp1);
    if ~isnumeric(Counter)
        fprintf('Counter Error - Not Numeric');
    end
    if Counter == CurrentCounter+1
```

```
        CurrentCounter = Counter;
        Temp1 = textscan(fid,'%s');
        Temp2 = Temp1{1,1};
        Proc = Temp2(2);
        i = 4;
        while i<=length(Temp2) && ~strcmp(cell2mat(Temp2(i)),'*')
            Inputs((i-1)/3,1) = Temp2(i);
            Inputs((i-1)/3,2) = Temp2(i+2);
            i = i+3;
        end

        k = i-2;
        i = i+1;
        while i<=length(Temp2) && ~strcmp(cell2mat(Temp2(i)),'*')
            Outputs((i-k)/3,1) = Temp2(i);
            Outputs((i-k)/3,2) = Temp2(i+2);
            i = i+3;
        end

        %Inputs and outputs will be sent & recieved to\from MATLAB in
        %alphabitcal order
        [O1,O2] = sort(Inputs(:,1));
        [Inputs(:,1),O2] = sort(Inputs(:,1));
        Inputs(:,2) = Inputs(O2,2);

        [O3,O4] = sort(Outputs(:,1));
        [Outputs(:,1),O4] = sort(Outputs(:,1));
        Outputs(:,2) = Outputs(O4,2);

        fclose(fid);
        if SimOrMat
            % Run Simulation
            cd(SimFolder);
            for i = 1:1:length(Inputs(:,1))
                Temp4 = cell2mat(Inputs(i,1));
                if ~strcmp(Temp4(1),'$') && ~strcmp(Temp4(1),'@')
                    Temp5 = cell2mat(Inputs(i,2));
                    Temp5 = Temp5(2:1:end-1); % got rid of " "
                    eval([Temp4 '.time = 0;']);
                    eval([Temp4 '.signals.values = ' Temp5 ';']);
                end
            end

            eval(['simOut = sim(' '''' cell2mat(Temp2(2)) '''' ',' '''solver''' ','
'''VariableStepDiscrete''' ',' '''Max Step Size''' ',' '''10''' ',' ''''
'SrcWorkspace' '''' ',' '''' 'current' '''' ');']);

            for i = 1:1:length(Outputs(:,1))
                Temp6 = cell2mat(Outputs(i,1));
                if ~strcmp(Temp6(1),'$') && ~strcmp(Temp6(1),'@')
                    eval(['yout = simOut.find(''' Temp6 ''''  ');']);
                    eval([Temp6 ' = yout.signals.values(1)']);
                end
            end
        else
            %%% call m file
            cd(SimFolder);
            for i = 1:1:length(Inputs(:,1))
                Temp4 = cell2mat(Inputs(i,1));
                if ~strcmp(Temp4(1),'$') && ~strcmp(Temp4(1),'@')
                    Temp5 = cell2mat(Inputs(i,2));
                    Temp5 = Temp5(2:1:end-1); % got rid of " "
                    eval([Temp4 ' = ' Temp5 ';']);
                end
            end
```

```
            fid = fopen('C:\Thesis work\OPM Subcontracter code\gen.m','r');
            Temp7 = textscan(fid,'%s');
            Temp8 = Temp7{1,1};
            Temp9 = cell2mat(Temp8(4));

            eval(['Temp6 = ' Temp9]);
        end

        % Write message 2 Opcat
        TempRoot = [Root 'Mat2Op.txt'];
        fid = fopen(TempRoot,'w+');

        fprintf(fid,[num2str(Counter) '\r\n*\r\n']);

        fprintf(fid,[cell2mat(Temp7(2)) '\r\n*\r\n']);


        for i = 1:1:length(Outputs(:,1))
            Temp6 = cell2mat(Outputs(i,1));
            if ~strcmp(Temp6(1),'$') && ~strcmp(Temp6(1),'@')
                eval(['fprintf(fid,[''' Temp6 ' = {"' '''' ' ' 'num2str(' Temp6 ')
'"}\r\n'' ]);']);
            end
        end




        fclose(fid)

    elseif Counter<CurrentCounter || Counter>CurrentCounter+1
        fprintf('Counter Error - Skiped Counter');
    end

    % Check if Simulation is over
    TempRoot = [Root 'Run.txt'];
    fid = fopen(TempRoot,'r');
    KeepRunning = fscanf(fid,'%d');
    fclose(fid);
end

if ~KeepRunning
    Running = 0;
    return;
end
```

Appendix D – iBuy Scope & Requirements

iBuy

Introduction: the purpose and scope of the system

The main purpose of iBuy is online grocery shopping. iBuy provides a simple, effortless, economical way to shop for a variety of food items, and allows 'social shopping' - recommending items to different users and sharing a shopping list, trading online coupons with other users, etc.

A user of iBuy can create a temporary shopping list or save and edit a list of often purchased items. Based on the user's shopping list and purchase record, iBuy can offer recommended items to the user. The users can be linked to other users, watching their shopping lists and reviews.

iBuy relies on a set number of existing grocery store online ordering sites to provide the purchased goods. iBuy acts as a mediator, purchasing the items from the different online stores and supplying them to the user. The shipment from all the stores arrives to an iBuy warehouse, where it is combined to one package by iBuy shipping department, and then it is sent to the customer.

A user can be a regular user, which means she or he does not pay a membership fee, or a premium user, who pays a yearly fee. A premium user can purchase groceries for lower prices than a regular user.

A user can switch status between regular and premium. If a regular user wants to become a premium user, he can do it whenever he wants; he just needs to pay the yearly fee. After a premium user's yearly fee ends, he automatically becomes a regular user, unless he extends it for another year.

Since iBuy is internet based, the site itself can be accessed worldwide. Yet, the service is limited to different shipping areas.

Top-level function and breakdown into function one level down

Top-level function

The top level function of the iBuy system is grocery shopping.

Users benefit from this function by the ability to shop for desired groceries while paying cheap prices while enjoying a simple purchasing process, a large product variety, and receiving and considering other people's recommendations.

Breakdown into function one level down

The system shall implement as a minimum the following list of sub-functions:

- User managing
- Shopping list creating

- Products ordering
- Item recommending
- Shopping list sharing
- Coupon searching and acquiring

Expected business model

Potential site users and their motivation to use it

iBuy can be used by any internet user in the shipment coverage areas. The user may enter the site and browse through the various items as a viewer. However in order to buy or comment on an item, the user must be registered. Registering is free, and requires a valid name and home address.

The motivation to use iBuy varies for different users. The site gives users an easy and simple way to purchases their needed groceries, for low prices. The system uses different online grocery stores and orders the goods from the cheapest source, eliminating the user's need to check each online store and compare their prices, making the process simple and quick.

For users who like to shop using online coupons at the different online grocery stores, reading product reviews and recommending various products, iBuy provides a 'social-shopping' section. Users interested in reading reviews on different brands of food, or new products, can use iBuy to do so. The coupons are unique to iBuy, and can't be gotten anywhere else.

Benefits to site owners and/or operators

iBuy owners benefit from the profit margin of selling some of the products from the different online stores for slightly higher prices. Some of the products are sold at the online store cost, and some are sold for even cheaper then the online stores (losing price) in order to attract customers. Alternatively, the owners benefit from the yearly fee premium users pay. For these users the profit margin is minimal since the products are sold at higher prices than those at the online stores.

Another source of owners benefit is payment received from suppliers interested in promoting their products via coupons offered on the site.

Look-and-feel requirements

In order to maximize the user's buying experience, the site should have a clean, intuitive and modern looking design.

After signing in, the user can use the menu located at the right-top corner of the screen:

- The iBuy logo for returning to the homepage
- 'My account' for managing user information and purchasing a premium account
- 'Start Shopping', which allows the user to create a one time shopping list or use his previous lists, and continue the shopping process
- 'Coupon and sales search' for finding special sales, and viewing various coupon. The user can add his or her choose to the shopping list.
- 'Back to my cart' for returning to a previously un-finished shopping session

- ‘Product search’ for finding a single product, seeing it’s price, adding it to a shopping list or writing a review.
- ‘Social Shopping’ for reading product reviews, editing reviews previously written by the user (writing a new review is possible when reaching a product from the ‘Product search’ option)

Special requirements for this system

1.1. Security

Since the system includes online credit card transaction as well as private information such as names and home address, the information must be secured and protected from hackers and breaches.

1.2. Availability

The system shall be available at almost all times – at least 99% availability.

1.3. Reliability

The system will be reliable, such that all prices presented to the user will be correct and products will arrive within 4 hours of purchase

1.4. Usability

Since the expected users of iBuy are from different backgrounds and computer expertise levels, it shall be easy to use. All functions of the system shall be easy to access and use.

1.5. Performance

iBuy shall support fast search and product extraction from it’s database, so the user does not loose patients.

1.6. Compatibility

iBuy shall support the different web browsers commonly used.

Appendix E – AUTOMATLAB Evaluation Questionnaires and Data Sets

Evaluation Questionnaire:

Advanced Topic Assignment

Specification and Analysis of Information Systems (094222) - Spring 2013

(Page 1 of 7)

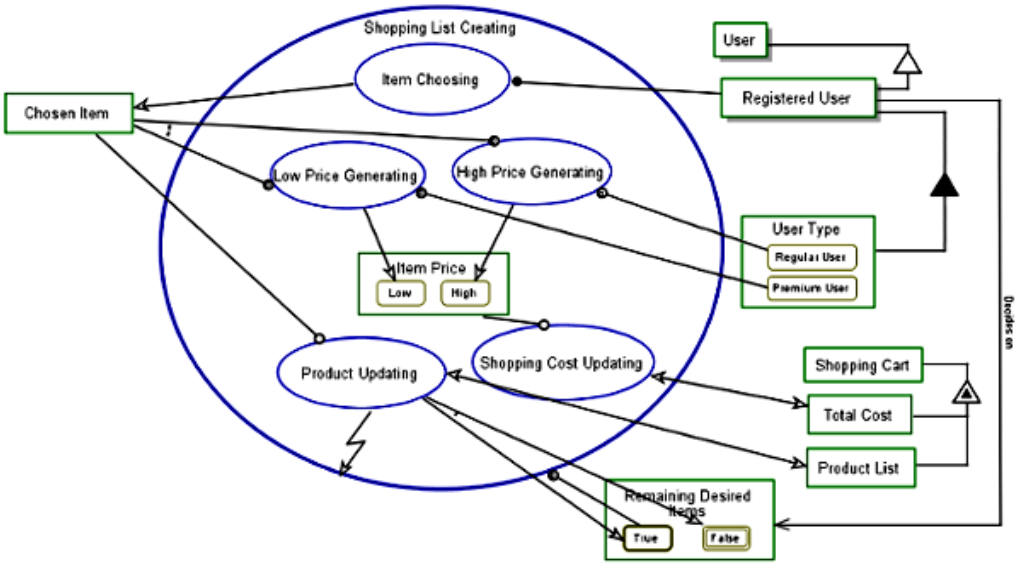
Background

As part of your course assignments, you are required to analyze some potential factors of the 'iBuy' system detailed in the Scope & Requirements Document, using the OPM model created by you or your fellow students.
The different factors to be analyzed will refer mainly to the 'Shopping List Creating' process in the OPM model.

As part of the assignment you are requested to evaluate the answer to four different questions. After each answer you will be asked to answer a few questions regarding your evaluation and confidence in the answer you provided.

Note: your grade for this assignment will not be based on the correctness of you answers, but rather on the analysis process you present and creativity in achieving the answers.
In addition to filling this form, you are requested to submit any material used in your analysis.

Good luck!



Continue »

Advanced Topic Assignment

* Required

(Page 2 of 7)

The iBuy system owner has requested you analyze some key factors in his business model regarding the system you have modeled for him. You are requested to do so using the data sets given to you that include potential customer behavior models, item cost, and potential pricing strategies.

Based on your analysis, please answer the following questions:

General Questions

Student ID: *

What data set were you given for this analysis? *
(After completing this form for the first data set, you will be given the second set)

- ○ iBuy_Jerusalem.zip
- ○ iBuy_Tel-Aviv.zip

Were you part of the iBuy group who created the model? *

- ○ I was part of the iBuy group
- ○ I was not part of the iBuy group, but received an explanation about the system and OPM model

Did you receive the MATLAB simulation automatically generated from the OPM model? *

- ○ Yes
- ○ No

« Back | Continue »

Advanced Topic Assignment

* Required

Question 1:

(Page 3 of 7)

1.1 What type of customer is more profitable for the iBuy owner? *

When referring to profit take into account item selling price, item retailer cost, premium users monthly fee and any other important information.

- ○ Regular user
- ○ Premium user

1.2 Explain how you have deducted this: *

You can refer to the additional material you have submitted.

1.3 How accurate do you think you answer is? *

Please rate on a scale of 1-5, when 1 is very inaccurate and 5 is entirely accurate.

	1	2	3	4	5	
inaccurate	○	○	○	○	○	accurate

1.4 Explain why you think your answer is accurate / inaccurate *

1.5 How hard was obtaining your answer? *

Please rate on a scale of 1-5, when 1 is easy and 5 is hard.

	1	2	3	4	5	
easy	○	○	○	○	○	hard

1.6 Explain why obtaining the answer was easy / hard *

1.7 How long did it take you to reach your answer? *

« Back | Continue »

Advanced Topic Assignment

* Required

Question 2:

(Page 4 of 7)

2.1 What are the 3 most profitable products for the iBuy owner?

When referring to profit take into account item selling price, item retailer cost, premium users monthly fee and any other important information such as the actual amount of items sold.

2.2 Explain how you have deducted this: *

You can refer to the additional material you have submitted.

2.3 How accurate do you think you answer is? *

Please rate on a scale of 1-5, when 1 is very inaccurate and 5 is entirely accurate.

	1	2	3	4	5	
inaccurate	○	○	○	○	○	accurate

2.4 Explain why you think your answer is accurate / inaccurate *

2.5 How hard was obtaining your answer? *

Please rate on a scale of 1-5, when 1 is easy and 5 is hard.

	1	2	3	4	5	
easy	○	○	○	○	○	hard

2.6 Explain why obtaining the answer was easy / hard *

2.7 How long did it take you to reach your answer? *

« Back | Continue »

Advanced Topic Assignment

* Required

Question 3:

(Page 5 of 7)

3.1 What is the premium user monthly fee that will maximize the profit for the iBuy owner? *

This monthly fee is alternative to the one given to you in the data files

3.2 Explain how you have deducted this: *

You can refer to the additional material you have submitted.

3.3 How accurate do you think you answer is? *

Please rate on a scale of 1-5, when 1 is very inaccurate and 5 is entirely accurate.

	1	2	3	4	5	
inaccurate	○	○	○	○	○	accurate

3.4 Explain why you think your answer is accurate / inaccurate *

3.5 How hard was obtaining your answer? *

Please rate on a scale of 1-5, when 1 is easy and 5 is hard.

	1	2	3	4	5	
easy	○	○	○	○	○	hard

3.6 Explain why obtaining the answer was easy / hard *

3.7 How long did it take you to reach your answer? *

« Back | Continue »

Advanced Topic Assignment

* Required

Question 4:

(Page 6 of 7)

4.1 What is the premium user monthly fee that will equalize the amount of items purchased by regular users and premium users? *

4.2 Explain how you have deducted this: *

You can refer to the additional material you have submitted.

4.3 How accurate do you think you answer is? *

Please rate on a scale of 1-5, when 1 is very inaccurate and 5 is entirely accurate.

	1	2	3	4	5	
inaccurate	○	○	○	○	○	accurate

4.4 Explain why you think your answer is accurate / inaccurate *

4.5 How hard was obtaining your answer? *

Please rate on a scale of 1-5, when 1 is easy and 5 is hard.

	1	2	3	4	5	
easy	○	○	○	○	○	hard

4.6 Explain why obtaining the answer was easy / hard *

4.7 How long did it take you to reach your answer? *

« Back | Continue »

Advanced Topic Assignment

* Required

Summation

(Page 7 of 7)

5.1 How long in total did it take you to reach all answers? *

5.2 What level of expertise do you have with the tools and methods you used to reach your answers? *

From 1 to 5, when 1 is low level of expertise and 5 is high level of expertise.

	1	2	3	4	5	
low leval of experties	○	○	○	○	○	high leval of experties

5.3 General comments:

« Back Submit

Data Sets:

Items:

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	16.57	19.02	16.63
9377985	Chicken	Meats	5.12	15.54	10.58
6682349	Chocolate chips	Baking Products	2.77	3.77	2.77
6501778	Cinnamon	Spices and Herbs	0.20	1.20	0.20
3020596	Coffee	Various Groceries	18.14	27.83	21.96
1637042	Cookies	Baked Goods	5.32	9.03	7.08
1819901	Corn	Vegetables	8.93	11.99	8.38
6971585	Cornflakes cereal	Various Groceries	6.66	9.03	8.37
2562072	Cottage cheese	Dairy and Cheese	4.60	6.40	4.40
4688139	Crackers	Various Groceries	6.25	20.76	16.71
2886293	Cream cheese	Dairy and Cheese	3.00	3.78	2.21
6810165	Cucumbers	Vegetables	0.85	3.49	0.58
2858148	Donuts	Baked Goods	8.29	12.99	10.89
9080039	Feta	Dairy and Cheese	3.28	3.35	3.35
9967952	Fish sticks	Various Groceries	10.47	16.39	15.37
2794505	Flour	Baking Products	1.26	2.26	1.26
7183122	Fresh bread	Baked Goods	4.76	5.44	4.16
9432786	Frozen steak	Meats	18.64	43.49	38.55
8780756	Garlic	Spices and Herbs	0.33	1.33	0.33
1407885	Ginger	Spices and Herbs	0.29	0.99	0.29
6224142	Goat cheese	Dairy and Cheese	3.80	4.09	3.63
9216096	Grapefruit	Fruits	21.32	25.03	23.82
1126031	Grapes	Fruits	2.98	4.20	3.47
9148168	Ground beef	Meats	24.66	26.65	16.49

1611773	Gum	Various Groceries	10.66	12.90	10.62
4146529	Hamburgers	Meats	3.30	30.37	8.82
1140283	Honey	Various Groceries	4.14	14.20	10.34
7847177	Hot dogs	Meats	18.51	36.34	34.23
7165095	Kiwis	Fruits	3.35	20.38	14.90
7437078	Lemon juice	Various Groceries	10.96	25.79	20.72
6201507	Lemons	Fruits	10.96	11.16	10.07
2445804	Lettuce	Vegetables	0.80	5.00	5.00
9580629	Margarine	Dairy and Cheese	7.54	7.99	7.08
3480016	Mayonnaise	Various Groceries	5.22	8.63	6.09
2465482	Melon	Fruits	15.55	16.97	15.30
1614747	Milk	Dairy and Cheese	1.29	4.00	1.86
5760988	Mint	Spices and Herbs	1.05	2.05	1.05
8742749	Mozzarella	Dairy and Cheese	4.75	9.73	7.93
8918899	Mushrooms	Vegetables	7.10	12.00	10.00
7341510	Nectarines	Fruits	14.29	15.81	10.29
7857195	Nuts	Various Groceries	5.07	8.06	7.28
4080826	Olive oil	Various Groceries	23.78	31.01	26.35
3690880	Onions	Vegetables	2.99	3.99	3.99
7009592	Oranges	Fruits	7.23	19.74	15.66
1375077	Oregano	Spices and Herbs	2.29	3.29	2.29
1089432	Paprika	Spices and Herbs	2.28	3.28	2.28
6600213	Parmesan	Dairy and Cheese	3.61	7.32	4.53
1841910	Parsley	Spices and Herbs	2.30	3.30	2.30
3636857	Pasta	Various Groceries	16.48	19.33	17.25
7681356	Pastrami	Meats	13.21	17.34	15.98
9252650	Peaches	Fruits	22.90	23.78	22.24
9180625	Peanut butter	Various Groceries	9.77	16.86	15.08
1819795	Pears	Fruits	2.14	9.24	6.89
7893620	Peppers	Vegetables	0.39	0.59	0.29
4249388	Pita bread	Baked Goods	5.15	10.00	5.26
3997128	Plums	Fruits	8.55	15.31	10.53
6951364	Potatoes	Vegetables	2.80	2.99	2.09
6038031	Pretzels	Various Groceries	5.77	11.02	5.57
3295812	Rice	Various Groceries	20.58	31.90	25.56
3995896	Rolls	Baked Goods	3.59	4.60	3.99
9658098	Salami	Meats	13.99	38.18	20.75
5808082	Salt	Spices and Herbs	1.00	2.00	1.00
8763108	Sliced bread	Baked Goods	3.50	4.12	3.79
5714720	Sour cream	Dairy and Cheese	6.95	9.27	8.91
7100442	Spinach	Vegetables	14.16	16.20	15.09
7555115	Squash	Vegetables	6.64	17.99	15.99
5897823	Sugar	Baking Products	0.29	0.95	0.29
9399664	Tea	Various Groceries	7.56	9.02	5.50
1260252	Turkey	Meats	5.79	23.32	14.41
2953514	Vanilla extract	Spices and Herbs	0.74	1.74	0.74
8279853	Vegetable oil	Various Groceries	7.61	26.52	12.87
9002834	Vinegar	Various Groceries	7.92	10.68	8.87
1648305	Whipped cream	Dairy and Cheese	3.82	5.90	3.22
9052801	Yeast	Baking Products	2.65	3.65	2.65
4776807	Yogurt	Dairy and Cheese	4.07	4.63	4.42
8523762	Zucchini	Vegetables	6.25	13.90	9.00

* Item Cost and Selling Price are per Item or per Kg, according to context

Premium user monthly fee:

2200 NIS

Jerusalem customer list:

Customer ID:
c218207544
c391189783
c551591397
c370330601
c268373537
c893842243
c155993966
c143487860
c908306102
c599532786

Tel-Aviv customer list:

Customer ID:
c218207544
c391189783
c551591397
c370330601
c268373537
c893842243
c155993966
c143487860
c908306102
c765581903
c879340534
c136383197
c759933163
c753189715
c988442207
c596474918
c360693327
c185365991
c216912694
c599532786

Example of customer potential shopping list:

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
3690880	Onions	6
1375077	Oregano	7
1841910	Parsley	4
9252650	Peaches	6
1819795	Pears	4
7893620	Peppers	1
3997128	Plums	5
6951364	Potatoes	4
6038031	Pretzels	8
9658098	Salami	6
5808082	Salt	5
8763108	Sliced bread	6
5714720	Sour cream	1
7555115	Squash	2
9002834	Vinegar	5

Customer behavior model - Jerusalem:

- Customers buy according to their shopping list.
- If a Premium user wants to buy a product, and receives less than 20% discount from the Regular users price, there is a 50% chance he will not buy the product (he will continue purchasing from the next item on the list).
- If a Regular user can pay less on his entire shopping list if he was a premium user (including the Premium user fee), he will become a Premium user.

Customer behavior model – Tel Aviv:

- Customers buy according to their shopping list.
- If a Premium user wants to buy a product, according to the discount he receives (or lack of) he might decide not to purchase the product. The chance of purchasing is in correlation to the discount received: for a X% discount from the Regular user price, there is a 2*X% chance the customer will purchase the item (up to 100% chance). For example, if the Premium price is a 30% discount compared to the Regular price, there is a 60% chance the Premium user will purchase the item.
- If a Regular user can pay less on his entire shopping list if he was a Premium user (including the Premium user fee), there is an 80% chance he will become a Premium user, a 10 % chance he will stay a Regular user, and 10% he will not purchase any items at all.
- If a Premium user can pay less on his entire shopping list if he was a Regular user (including not paying the Premium user fee), there is a 70% chance he will become a Regular user and 30% chance he will not purchase any items at all.
- Fruits and Vegetables are on extra 10% discount for Premium user, unless they also purchase Spices and Herbs, in which case they are not entitled to the 10% discount.

Appendix F – Example of Enhanced MATLAB code

An example of enhanced MATLAB code created by one of the students as part of the AUTOMATLAB evaluation can be seen here. The code was expanded from the automatically generated code from the AUTOMATLAB approach.

MainIBuy.m:

```
function [ output_args ] = MainIBuy( input_args )

%[A,BT]=
xlsread('C:\Users\shani\matlab\iBuy_Jerusalem\iBuy_Jerusalem.xlsx','Customer
List');
[A,BT]= xlsread('iBuy_Jerusalem.xlsx','Customer List');
%[Items_int, Items_str] =
xlsread('C:\Users\shani\matlab\iBuy_Jerusalem\iBuy_Jerusalem.xlsx','Items');
[Items_int, Items_str] = xlsread('iBuy_Jerusalem.xlsx','Items');
%PremiumFee =
xlsread('C:\Users\shani\matlab\iBuy_Jerusalem\iBuy_Jerusalem.xlsx','Premium Fee');
PremiumFee =  xlsread('iBuy_Jerusalem.xlsx','Premium Fee');
RegularProfit = 0;
PremiumProfit = 0;
z = zeros(length(Items_int), 4);
ItemsList = [Items_int z];
RegularUserCounter = 0;
PremiumUserCounter = 0;
differCost = 0;
feeLevels = zeros(length(BT),1);
TotalPurchasedProducts_Regular = 0;
TotalPurchasedProducts_Premium = 0;
    for i=2:(length(BT))
        Customer =char( BT(i) );
       % [C_A,C_BT]=
xlsread('C:\Users\shani\matlab\iBuy_Jerusalem\iBuy_Jerusalem.xlsx',Customer);
         [C_A,C_BT]= xlsread('iBuy_Jerusalem.xlsx',Customer);
        UserEntity = char(C_BT(2,2));%Primume or Private customer
  %       ProductList = C_BT( [5:end], 2 );
        RegisteredUser = C_A( [5:end], [1,3] );

[TotalCost,TotalCostForPremuim,TotalRetailerCost,ItemsList,PurchasedProducts] =
ShoppingListCreating(RegisteredUser,UserEntity,ItemsList,'False');
         if isequal(UserEntity,'Regular user')
             feeLevels(i) = TotalCost - TotalCostForPremuim;
             differCost = max(differCost, TotalCost - TotalCostForPremuim);
              TotalCostForPremuim = TotalCostForPremuim + PremiumFee;
              if TotalCostForPremuim < TotalCost
                  %regular user becomes a premium user
                   PremiumUserCounter = PremiumUserCounter + 1;
                   [TotalCost,~,TotalRetailerCost,ItemsList,PurchasedProducts] =
ShoppingListCreating(RegisteredUser,'Premium user',ItemsList,'True');
                    profit = (TotalCost + PremiumFee) - TotalRetailerCost;
                    PremiumProfit = PremiumProfit + profit ;
                    TotalPurchasedProducts_Premium = TotalPurchasedProducts_Premium
+ PurchasedProducts;
              else
                  %Stay a regular user
                     RegularUserCounter = RegularUserCounter+1;
                     profit = TotalCost  - TotalRetailerCost;
                     RegularProfit = RegularProfit + profit ;
```

```
                    TotalPurchasedProducts_Regular =
TotalPurchasedProducts_Regular + PurchasedProducts;
                end
            else
                %premium user
                PremiumUserCounter = PremiumUserCounter + 1;
                TotalPurchasedProducts_Premium = TotalPurchasedProducts_Premium +
PurchasedProducts;
                profit = (TotalCost + PremiumFee) - TotalRetailerCost;
                PremiumProfit = PremiumProfit + profit ;
            end
    end
PremiumProfitAvg = PremiumProfit/PremiumUserCounter;
RegularProfitAvg = RegularProfit/RegularUserCounter;
[p1,p2,p3]= MostProfitableProducts(ItemsList, Items_str);
display(RegularProfitAvg);
display(PremiumProfitAvg);
display(sprintf('the 3 most profitable products are: ''%s'' ''%s''
''%s',char(p1),char(p2),char(p3)));
display(sprintf('the desired fee is at least: ''%d', differCost));
display(feeLevels);
display(sprintf('Total purchased products for Premium: ''%d',
TotalPurchasedProducts_Premium));
display(sprintf('Total purchased products for Regular: ''%d',
TotalPurchasedProducts_Regular));
end
```

ShoppingListCreating.m:

```
function
[TotalCost,TotalCostForPremuim,TotalRetailerCost,ItemsList,TotalPurchasedProducts]
= ShoppingListCreating(RegisteredUser,UserEntity, ItemsList, IsNewPremiumUser)

% ProductList = zeros(length(RegisteredUser),2);
% ProductListIndex = 0;
RemainingDesiredItems = 'True';
itemIndex = 0;
TotalCost = 0;
TotalCostForPremuim = 0;
TotalRetailerCost = 0;
TotalPurchasedProducts = 0;

while isequal(RemainingDesiredItems,'True') % ShoppingListCreating occurs if
RemainingDesiredItems is True.
    itemIndex = itemIndex+1;

    if ~isempty(RegisteredUser)
        [ChosenItem] = ItemChoosing(RegisteredUser, itemIndex);
    end

    [ItemPrice, ItemPriceWithDiscount,itemRetailerCost,ItemIndexInList] =
PriceGenerating(ChosenItem,ItemsList);

    if ~isempty(ChosenItem) && isequal(UserEntity,'Regular user')
        %update amount of buyed products
        TotalPurchasedProducts = TotalPurchasedProducts + ChosenItem(2);
        [ItemsList] = UpdateTotalBuyedProducts( ItemsList, ItemIndexInList,
ChosenItem(2), UserEntity, IsNewPremiumUser, 1 );
    end

    if ~isempty(ChosenItem) && isequal(UserEntity,'Premium user')
         IsBuying = IsBuyingProduct(ItemPrice, ItemPriceWithDiscount );
         if (IsBuying == 1)
```

```
            TotalPurchasedProducts = TotalPurchasedProducts + ChosenItem(2);
            ItemPrice = ItemPriceWithDiscount;
        else
            ItemPrice = 0;
            itemRetailerCost = 0;
        end
         [ItemsList] = UpdateTotalBuyedProducts( ItemsList, ItemIndexInList,
ChosenItem(2), UserEntity, IsNewPremiumUser, IsBuying );
    end

    if ~isempty(ItemPrice) && ~isempty(TotalCost)
        [TotalCost] = ShoppingCostUpdating(ItemPrice,TotalCost);
        [TotalCostForPremuim] =
ShoppingCostUpdating(ItemPriceWithDiscount,TotalCostForPremuim); %low price for ALL
the shopping list
        TotalRetailerCost = TotalRetailerCost + itemRetailerCost;
    end

    % ProductUpdating invokes ShoppingListCreating.
    if (itemIndex>= length(RegisteredUser))
        RemainingDesiredItems = 'False';
    end
end

end
```

FindItemBySerialNumber.m:

```
function [ indexItem ] = FindItemBySerialNumber( ItemsList, SerialNum )

indexItem = 1;
found = 0;
while(found == 0 && indexItem<=length(ItemsList))
    if (~isnan(ItemsList (indexItem, 1)))
        if (ItemsList (indexItem, 1) == SerialNum)
            found = 1;
        else
            indexItem = indexItem + 1;
        end
    end
end

if (found == 0)
    indexItem = 0;
end

end
```

HighPriceGenerating.m:

```
function [ItemPrice] = HighPriceGenerating(ChosenItem,ItemsList)

ItemPrice = 0;
SerialNum = ChosenItem(1);
indexItem = FindItemBySerialNumber( ItemsList, SerialNum );
if (indexItem ~=0 )
    costForUnit = ItemsList(indexItem,5); %price for Regular user
    amount = ChosenItem(2);
```

```
    ItemPrice = costForUnit * amount;
end

end
```

IsBuyingProduct.m:

```
function [ result ] = IsBuyingProduct(ItemPrice, ItemPriceWithDiscount )
        if( ItemPriceWithDiscount > (0.8 * ItemPrice))    %  discaunt < 20%
            y = rand;
            if (y<0.5)
                result = 0;
            else
                result = 1;
            end
        else
            result = 1;
        end
end
```

ItemChoosing.m

```
function [ChosenItem] = ItemChoosing(RegisteredUser, itemIndex)
ChosenItem = RegisteredUser(itemIndex, [1,2]); % ItemChoosing yields ChosenItem.
end
```

LowPriceGenerating.m:

```
function [ItemPrice] = LowPriceGenerating(ChosenItem,ItemsList)

ItemPrice = 0;
SerialNum = ChosenItem(1);
indexItem = FindItemBySerialNumber( ItemsList, SerialNum );
if (indexItem ~=0 )
    costForUnit = ItemsList(indexItem,6); %price for Premium user
    amount = ChosenItem(2);
    ItemPrice = costForUnit * amount;
end

end
```

MostProfitableProducts.m

```
function [ p1,p2,p3 ] = MostProfitableProducts( ItemsList, ItemsNamesList )
ItemNameCol = 2;
RetailerCostCol = 4;
RegularCostCol = 5;
PremiumCostCol = 6;
RegularAmountCol = 7;
```

```
PremiumAmountCol = 8;
ProfitCol = 9;
OriginalIndexCol = 10;
for i=1:length(ItemsList)
     ItemsList(i, OriginalIndexCol) = i ;
end
for i=1:length(ItemsList)
    RegularCost = ItemsList(i, RegularCostCol)*ItemsList(i, RegularAmountCol);
    PremiumCost = ItemsList(i,PremiumCostCol )*ItemsList(i,PremiumAmountCol);
    RetailerCost =  ItemsList(i,RetailerCostCol )* ( ItemsList(i,PremiumAmountCol)
+ ItemsList(i, RegularAmountCol)  );
    ItemsList(i, ProfitCol) = ItemsList(i, ProfitCol) + RegularCost + PremiumCost -
RetailerCost ;
end
[sortedCol, sorter] = sort(ItemsList(:,ProfitCol), 'descend');
ItemsList = ItemsList(sorter, :);
p1 = ItemsNamesList( ItemsList(1, OriginalIndexCol) + 1, ItemNameCol);
p2 = ItemsNamesList( ItemsList(2, OriginalIndexCol) + 1, ItemNameCol);
p3 = ItemsNamesList( ItemsList(3, OriginalIndexCol) + 1, ItemNameCol);
end
```

PriceGenerating.m:

```
function [ ItemPrice, ItemPriceWithDiscount,RetailerCost,indexItem ] =
PriceGenerating(ChosenItem,ItemsList)

ItemPrice = 0;
SerialNum = ChosenItem(1);
indexItem = FindItemBySerialNumber( ItemsList, SerialNum );
if (indexItem ~=0 )
    amount = ChosenItem(2);
    costForUnit = ItemsList(indexItem,5); %price for Regular user
    ItemPrice = costForUnit * amount;
    costForUnit = ItemsList(indexItem,6); %price for Premium user
    ItemPriceWithDiscount = costForUnit * amount;
     costForUnit = ItemsList(indexItem,4); %Retailer Cost
    RetailerCost = costForUnit * amount;
end
end
```

ProductUpdating.m:

```
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
```

ShoppingCostUpdating.m:

```
function [TotalCost] = ShoppingCostUpdating(ItemPrice,TotalCost)
[TotalCost] = TotalCost + ItemPrice;
end
```

UpdateTotalBuyedProducts.m:

```
function [ ItemsList ] = UpdateTotalBuyedProducts( ItemsList, ItemIndexInList,
amountBuyed, UserEntity, IsNewPremiumUser, IsProductBuyed )
RegularBuyColumn = 7;
PremiumBuyColumn = 8;

if isequal(UserEntity,'Regular user')
    ItemsList( ItemIndexInList, RegularBuyColumn ) = ItemsList( ItemIndexInList,
RegularBuyColumn ) + amountBuyed;
else %premium user
    if isequal(IsNewPremiumUser, 'True')
        %need to delete the buyed products from the Regular sum and then
        % (if the product buyed) add it to the Premuim sum
         ItemsList( ItemIndexInList, RegularBuyColumn ) = ItemsList(
ItemIndexInList, RegularBuyColumn ) - amountBuyed;
    end
    %now, if the product buyed,  add the buyed amount to the Premium sum
    if IsProductBuyed == 1
        ItemsList( ItemIndexInList, PremiumBuyColumn ) = ItemsList(
ItemIndexInList, PremiumBuyColumn ) + amountBuyed;
    end
end
end
```

References

[1] Dori, D. Object-Process Methodology – A Holistic Systems Paradigm, Springer Verlag, Berlin, Heidelberg, New York, 2002.

[2] Dori, D., Linchevski, C., and Manor, R. OPCAT – A Software Environment for Object-Process Methodology Based Conceptual Modeling of Complex Systems. Proc. 1st International Conference on Modeling and Management of Engineering Processes, University of Cambridge, Cambridge, UK, Heisig, P., Clarkson, J., and Vajna, S. (Eds.), pp. 147-151, July 19-20, 2010.

[3] Houcque, D. Introduction to MATLAB for Engineering Students. Northwestern University, Version 1.2, August 2005.

[4] Mattsson, S., Elmqvist, H. Modelica – An International Effort to Design the Next Generation Modeling Language. 1997.

[5] Harel, D., Marelly, R. Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-engine, Springer Verlag, 2003.

[6] Schultz, M., Zerbe,V., and Marwedel, S. Using the Object Process Methodology to Build Simulation Models, Proc. 3rdInternational Conference on Model-Based Systems Engineering, Fairfax, VA, USA, 2010.

[7] Karris, S.T., Introduction to Simulink with Engineering Applications. Orchard Publications, 2nd edition, 2008.

[8] Kelton D.W., Sadowski R.P., and Sadowski D.A. Simulation with Arena. McGraw-Hill, 2000.

[9] Gilat, T. A Framework for Simulation of Discrete Events Systems Based on the Object-Process Methodology, Technion, PhD thesis, 2002.

[10] Bolshchikov, S., Somekh, J., Mazor, S., Monadeev, M., Hertz, S., Choder, M., and Dori, D. Visualizing the Dynamics of Conceptual Behavior Models: The Vivid OPM Scene Player. Proc. 3rdInternational Conference on Model-Based Systems Engineering, Fairfax, VA, USA, 2010.

[11] Weilkiens, T. Systems Engineering with SysML/UML: Modeling, Analysis, Design, 2007.

[12] Operational Semantics for OPM, Dov Dori, Ofer Strichman, Valeria Perelman. March 2011, DRAFT.

[13] MathWorks Documentation Center MATLAB Functions, July 2011 http://www.mathworks.com/help/releases/R2009b/helpdesk.html

[14] Judith Somekh, Mordechai Choder, and Dov Dori, Conceptual Model-Based Systems Biology: Mapping Knowledge and Discovering Gaps in the mRNA Transcription Cycle. PLoS ONE, 7(12): e51430. doi:10.1371/journal.pone.0051430, Dec. 20, 2012. http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0051430

[15] OMG Unified Modeling Language (OMG UML) Infrastructure, Version 2.4.1, Object Management Group, August2011.

[16] OMG Systems Modeling Language (OMG SysML), Version 1.3, Object Management Group, June 2012.

[17] B.P. Zeigler. Theory of Modeling and Simulation. Wiley, New York. 1976

[18] R. Davies, P. Roderick and J. Raftery. The Evaluation of Disease Prevention and Treatment using Simulation Models. European Journal of Operational Research, 150, 53-66. 2003.

[19] R. Sinha, V.C. Liang, C.J.J. Paredis, and P.K. Khosla. Modeling and Simulation Methods for Design of Engineering Systems. Journal of Computing and Information Science in Engineering. Vol. 1, pp. 84-91, 2001.

[20] S. Robinson. Conceptual Modeling for Simulation part I: Definition and Requirements. Journal of the Operational Research Society, Vol. 59, No. 3 (Mar., 2008), pp. 278-290

[21] S. Robinson. Conceptual Modeling for Simulation. Wiley Encyclopedia of Operations Research and Management Science, 2010.

[22] A. Maria. Introduction to Modeling and Simulation. Proceedings of the 29th conference on winter simulation (WSC '97). 1997.

[23] J. S. Carson, II. Introduction to Modeling and Simulation. Proceedings of the 36th conference on winter simulation (WSC '04). 2004.

[24] Web OPCAT Project - Enterprise Systems Modeling Lab, April 2013, http://esml.iem.technion.ac.il/?page_id=952

[25] N. Sharma, and M.K. Gobbert, A comparative evaluation of MATLAB, Octave, FreeMat, and Scilab for research and teaching. 2010.

[26] Scolnik, M. Introduction to Radar Systems. New York, NY:McGraw-Hill, 1980.

שילוב היבטים כמותיים לתוך מודלים קונספטואלים מבוססי מתוגולגיית עצמים-תהליכים עם יכולות חישוביות של מאטלאב

אהרון רניק

שילוב היבטים כמותיים לתוך מודלים קונספטואלים מבוססי מתוגולגיית עצמים-תהליכים עם יכולות חישוביות של מאטלאב

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת ניהול מידע

אהרון רניק

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

אלול תשס״ג חיפה אוגוסט 2013

המחקר נעשה בהנחיית פרופסור דב דורי בפקולטה להנדסת תעשייה וניהול.

ברצוני להודות לפרופ׳ דב דורי על ההנחייה המסורה ועל היחס האישי והחם.

תודה לדר׳ ניבה ונגרוביץ, מר סרגיי בולישניקוב ומר אלכס בלכמן על כל הסיוע והרעיונות המעולים.

תודה מיוחדת לאשתי האהובה ליאת ולבנותי הנפלאות לינוי, שהם ושקד, עבור התמיכה והסבלנות לאורך לימודי.

תקציר

מידול הינו חלק חשוב במחזור החיים של מערכות, החל מהשלבים המוקדמים של התכנון. מידול שימושי ביותר גם בתהליך הלמידה והחקר של מערכות קיימות אשר אינן מוכרות. מתודולוגיות מידול שונות מאפשרות למדל מערכת בצורה קונספטואלית על-ידי התעלמות מהיבטים מסויימים של המערכת, תוך הקלת תהליך ההבנה או המידול של המערכת בזכות העברת ההיבטים החשובים של המערכת בצורה אפקטיבית.

מתודולגיות מידול שונות, כגון מתודולוגיית עצמים-תהליכים (OPM), UML ו-SysML מאפשרות למדל בצורה קונספטואלית את המערכת הרצויה. מידול זה נעשה על-ידי פישוט היבטים מסויימים של המציאות, כגון משוואות דיפרציליות, פונקציות הסתברות או היבטים כמותיים וחישוביים אחרים במערכת הממודלת, המשפיעים על פעולתה. פישוט זה מאפשר אמנם הבנה טובה של המודל קונספטואלי, אך ההיבטים הכמותיים המוזנחים עלולים לכלול מרכיבים החשובים להבנה מעמיקה של אופי המערכת ואופן פעולתה המפורט. קיימות מתודולוגיות מידול המאפשרות מידול עמוק של ההיבטים הכמותיים במערכת, אך גישה זו מתנגשת עם הגישה ההוליסטית המאפשרת מידול גם כאשר ההיבטים הכמותיים המדוייקים אינם נדרשים או אינם ידועים.

אנו מכנים את הבעיה הנ״ל ״בעיית הפישוט הכמותי״. בעיה זו מתייחסת לאחד החסרונות של מתודולגיות מידול קונספטואליות — הפישוט של המערכת הממודלת על חשבון דיכוי אפשרי של ההיבטים הכמותיים. מחקר זה מציג שתי גישות לפתרון בעיית הפישוט הכמותי של מודלים קונספטואליים המשתמשים במתודולגיית עצמים-תהליכים (OPM), שהינו סטנדרט ISO 19450 בהתהוות למתודולוגיית מידול.

מתודולגיית עצמים-תהליכים (OPM) מציעה גישה הוליסטית למידול מערכות המשלבת את המבנה וההתנהגות של המערכת בסוג דיאגרמה אחד. מודל OPM מורכב משני ישויות: אובייקטים בעלי מצבים ותהליכים. אובייקטים הם מרכיבים אשר יוצרים את המערכת, ותהליכים הם דברים אשר משנים את האובייקטים ע״י יצירת אובייקט, צריכת אובייקט, או השפעה על מצבו של אובייקט. כמו כן, המודל מכיל קשרים מבניים בין הדברים השונים: אובייקטים לאובייקטים אחרים, תהליכים לתהליכים אחרים, וקשרים תהליכיים בין אובייקטים לתהליכים. מודל OPM מוצג בשני אופנים חליפיים אשר מייצגים באופן מתואם את כל המידע הכלול במודל ומשלימים זה את זה מבחינה קוגניטיבית: גרפי וטקסטואלי.

אנו מרחיבים את ההיבטים הכמותיים של המודל על-ידי ייצוג התנהגות כמותית מורכבת, בעזרת שתי גישות חלופיות הנעזרות ב-MATLAB או Simulink, מבלי לפגוע בשלמות ופשטות מודל ה-OPM הקונספטואלי.

הגישה הראשונה, AUTOMATLAB, מרחיבה את מודל ה-OPM לסימולציה מלאה מבוססת מאטלאב. הגישה השנייה, קבלן משנה כמותי, מחליפה תת-תהליך במודל, ברמות עומק שונות, בפונקציית MATLAB או בדיאגרמת Simulink בעלת המאפיינים הכמותיים והחישוביים הרצויים.

בגישת AUTOMATLAB אנו מרחיבים את מודל ה-OPM ע״י יצירת סימולציה מבוססת MATLAB אשר מחליפה במלואה את מנוע הסימולציה של מודל ה-OPM. השלב הראשון בגישה זו הינו יצירת קוד MATLAB בסיסי. תהליך זה נעשה באופן אוטומטי בעזרת מחולל קוד שיצרנו. השלב השני הינו הרחבת הקוד ע״י הוספת המאפיינים הכמותיים באופן ידני על-ידי המשתמש, או מתוך בנק מודלים כמותיים

מוגדר מראש. לצורך כך, יצרנו מיפוי חד-חד ערכי בין תתי-מודלים לפונקציות MATLAB מובנות, אשר מאפשר את התרגום בין המודל לפונקציה המתאימה. השלב השלישי הינו הרצת הסימולציה. בשלב זה המשתמש עדיין צופה במודל בתצורתו הרגילה, אך קוד MATLAB הוא זה שמשפיע על מהלך הסימולציה לאורך ההתקדמות במודל, תוך התחשבות בכלל המאפיינים החישוביים וכמותיים במודל.

אחד החסרונות בגישת AUTOMATLAB הינו היכולת של המשתמש להפר את הסמנטיקה של מתודולוגיית עצמים-תהליכים על-ידי שינוי קוד MATLAB בצורה חופשית. על-מנת לאפשר שימוש נכון בגישת AUTOMATLAB נדרשת מהמשתמש שליטה טובה בסמנטיקה של מתודולוגיית עצמים-תהליכים. כחלופה לשיטה זו הצענו את הגישה של קבלן משנה כמותי, אשר אינה מאפשרת הפרה של הסמנטיקה של מתודולגיית עצמים-תהליכים ואינה דורשת שליטה מלאה בסמנטיקה זו. בגישת הקבלן המשנה הכמותי אנו מאפשרים החלפת תהליכים פנימיים במודל בפונקציית MATLAB או דיאגרמת Simulink אשר מכילה בתוכה את ההיבטים הכמותיים הרצויים לתת-מודל זה. השלב הראשון בגישה זו הינו יצירת פונקציות ב-MATLAB או דיאגרמות Simulink, המיועדות לשמש כקבלן משנה כמותי לתתי-מודלים שונים. לחילופין, ניתן להשתמש באותם תתי-מודלים שהוגדר עבורם מראש התרגול לפונקציית MATLAB , כפי שהוגדר בגישת AUTOMATLAB .

השלב השני הינו הרצת הסימולציה באופן רגיל במנוע סימולציית ה-OPM המקורי, כך שהסימולציה מתקדמת על-פי המודל הרגיל. השלב השלישי מתרחש כאשר הסימולציה מגיעה לתת-מודל שהוחלף על-ידי פונקציה או דיאגרמה המשמשת כקבלן משנה. מצבי המערכת נשלחים כקלט לאותה פונקציה או דיאגרמה, והסימולציה של תת-המודל מופעלת MATLAB או Simulink. מצבי המערכת אשר התקבלו מהסימולציה החיצונית נשלחים חזרה כפלט להמשך ריצה רגילה של סימולציית ה-OPM, עד הגעה לתת-מודל נוסף שהוחלף על-ידי קבלן משנה, וחוזר חלילה. בגישה זו, נדרש לשמור על תאימות בין הקלט והפלט של פונקציות ה-MATLAB ודיאגרמות ה-Simulink המחליפות את תתי-המודלים במודל המקורי, אך בתוך תת-המודל אין מגבלות הנובעות מהחוקים ב-OPM.

הדגמנו את שתי הגישות בעזרת מודלים של מערכת ביולוגית ומערכת גילוי מבוססת מכ״ם, שהורחבו בעזרת MATLAB ו-Simulink. בהדגמת המערכת הביולוגית הורחבו מספר מאפיינים כמותיים במערכת, כגון הסתברויות להתרחשות תהליכים שונים וזמני פעולה. הודגם כיצד הוספת מאפיינים אלו משפיעה על ריצת הסימולציה, ובנוסף כיצד ניתן להשתמש בקוד הנוצר לבחינת ביצועי המערכת בצורות שונות. בהדגמת מערכת הגילוי המכ״מית הורחב תת-המודל האחראי על הגילוי הראשוני במספר שיטות שונות. ראשית, תת-המודל הוחלף בדיאגרמת Simulink אשר מימשה את משוואת המכ״ם הפשוטה, ממנה נקבע האם התקבל גילוי של המטרה. בהמשך, הוחלפה דיאגרמה זו בפונקציית MATLAB אשר מימשה את אותה משוואת מכ״ם בצורה פשוטה וברורה יותר. לבסוף, הוחלפה פונקציית ה-MATLAB בדיאגרמת Simulink מורחבת, אשר כללה בנוסף למשוואת המכ״ם הפשוטה מרכיבים נוספים במערכת, כגון אלמנטים של שידור וקליטה, מודל איום מפורט ועוד. החלפת הדיאגרמות והפונקציות ששימשו כקבלני משנה בוצעה ללא שינוי במודל ה-OPM המקורי, והיתה שקופה למשתמש הצופה במודל ה-OPM.

בנוסף, בוצע ניסוי על משתמשים אנושיים אשר השווה את התועלת של גישת AUTOMATLAB אל מול השימוש ב-OPM ללא AUTOMATLAB. הניסוי כלל מודל OPM שנבנה על-ידי חלק מהסטודנטים בקורס "אפיון וניתוח מערכות מידע" בסמסטר אביב 2013 בטכניון – מכון טכנולוגי לישראל. המודל תיאר מערכת לרכישות אינטרנטית ותיאר את תהליכי ניהול המשתמשים, חיפוש מוצרים, יצירת רשימת קניות, הזמנה ועוד. הסטודנטים התבקשו לנתח מאפיינים שונים של המערכת על-פי נתוני מוצרים, עלויות, רשימת לקוחות ומודלי התנהגות לקוחות שונים אשר ניתנו להם. המאפיינים אותם נתבקשו הסטודנטים לנתח היוו תוצר כמותי של המודל. לדוגמה, הסטודנטים התבקשו למצוא את שלושת המוצרים הרווחיים ביותר עבור בעל החנות או את התשלום החודשי ללקוחות פרימיום, שישווה בין כמות הלקוחות שיבחרו להיות לקוחות פרימיום וכמות הלקוחות שיבחרו להשאר לקוחות רגילים.

כמחצית מהסטודנטים אשר היוו את קבוצת המחקר קיבלו את קוד ה-MATLAB המחולל אוטומטית ממודל ה-OPM והרחיבו אותו לסימולציית MATLAB שסייעה לבצע את הניתוח הנדרש. שאר הסטודנטים אשר היוו את קבוצת הביקורת, השתמשו במודל ה-OPM המקורי וביצעו את הניתוח בעזרת כל כלי אחר בו בחרו.

השערת המחקר שנבחנה בניסוי היתה ששימוש בגישת AUTOMATLAB תביא לתועלות שונות ביחס לשימוש ב-OPM ללא AUTOMATLAB. בפרט, נבחנו ההשערות הבאות:

1. המשתמשים ב-AUTOMATLAB ישיגו הבנה טובה יותר של ההיבטים החישוביים והכמותיים של המערכת, ביחס למשתמשים ב OPM ללא AUTOMATLAB.
2. המשתמשים AUTOMATLAB יבינו את ההיבטים החישוביים והכמותיים של המערכת תוך פחות זמן, ביחס למשתמשים ב OPM ללא AUTOMATLAB.
3. המשתמשים AUTOMATLAB ישיגו רמת ביטחון גבוהה יותר בהבנתם את ההיבטים החישוביים והכמותיים של המערכת, ביחס למשתמשים ב OPM ללא AUTOMATLAB.
4. המשתמשים של AUTOMATLAB ;ישיגו הבנה של ההיבטים החישוביים והכמותיים של המערכת תוך פחות קושי, ביחס למשתמשים ב OPM ללא AUTOMATLAB.

תוצאות הניסוי הראו כי אכן שימוש ב-AUTOMATLAB מאפשר הבנה טובה יותר בהיבטים החישוביים והכמותיים של המערכת (השערה מספר 1) וכי הבנה זו מתקבלת תוך פחות קושי (השערה מספר 4). תוצאות המחקר לא היו חד-משמעיות ביחס לזמן הנדרש ורמת הביטחון ביחס להבנת ההיבטים החישוביים והכמותיים של המערכת (השערות 2 ו-3).

ניתן להרחיב את הניסוי לצורך קבלת תוצאות חד משמעיות, ולבחון גם את התוצאות עבור שימוש בגישת קבלן משנה כמותי.