Grounded System Dynamics: A Procedure for Underpinning System Dynamics with a Domain Modeling Method

F. Tulinayo, P. Bommel, H. Proper

Grounded System Dynamics: A Procedure for Underpinning System Dynamics with a Domain Modeling Method

F.P (Fiona) Tulinayo[1], P. (Patrick) van Bommel[1], and H.A (Erik) Proper[1,2]

[1] Institute of Computing and Information Sciences, Radboud University Nijmegen
Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands, EU.
[2] Public Research Centre Henri Tudor, 29 avenue John F. Kennedy
Luxembourg-Kirchberg, Luxembourg
F.Tulinayo@science.ru.nl, pvb@cs.ru.nl, e.proper@acm.org

Abstract. In this paper, we present a procedure Grounded System Dynamics (GSD) which we use as a guide to underpin a System Dynamics (SD) model with a domain modeling method called Object-Role Modeling (ORM). GSD is a combination of two existing methods (SD with ORM). By combining these two methods we generate synergy effects by using already existing modeling methods and by so doing we overcome some of the weaknesses of SD model building. Secondly, transformation of information from an ORM model of dynamic domains into an SD model is achieved. To apply the GSD procedure to a real-life case (Mukono Health Center (MHC), we use SD-ORM mapped constructs. As a result from the GSD procedure application, is an SD model to which we define quantitative foundations that result into simulations. Our approach(GSD) has been validated using case studies one of which is described in this paper. From this conclusions are drawn.

Keywords: *System Dynamics, Object-Role Modeling*

1 Introduction

System Dynamics as a method was developed by Jay Forrester and dates as far back as 1961. A review and history can be found in [8]. It combines both quantitative and qualitative aspects [19] to explore complex models. In SD qualitative models, the structural features of the process are made explicit through casual loop diagrams [15]. In this paper however, we draw our attention to the quantitative aspect where stock and flow diagrams [7] are created and sets of equations input into the stock and flow diagram resulting into simulations. These simulations allow a modeler to provide quantitative estimates of system effects [15]. There have been earlier comparative studies between SD and other methods [21,6,2,4,3]. In these studies, a number of views on how SD relates or can work with these methods are given. In the context of enterprise modeling SD is typically used in process analysis design and optimization.

Consideration of many variables and complication in untangling the question of causation makes complex [28] system dynamics models hard to conceptualize and describe. Richardson [20] particularly states that; "*.....of the three tasks-model conceptualization, model formation and model understanding. Unquestionably the two most difficult are the two that are least formal- conceptualization and understanding. The future of the field needs software support for understanding the links between stock and flow/ feedback structure and dynamics behavior."* From this statement we note that, model conceptualization is one of the main issues in SD modeling. This issue is further emphasized in [16,20,13,18]. Sharif [22] also states that; *"....there is a strong case for starting to apply systems dynamics methods more openly in the BPM and MIS research fields, as I feel the tools and techniques available are vastly under-rated in terms of their applicability and capability to provide novel representations of real-world situations....."*. In Sharif's statement, he justifies why SD should be combined with other methods. In this research therefore, we contribute to addressing the issue of SD model conceptualization by adding domain modeling as a support in the creation of SD models. This is because domain modeling identifies relationships among all entities within the scope of the problem domain and provides a structural view of the domain hence, improving model conceptualization. As an example of a domain modeling language, we use ORM in particular because of its conceptual focus and roots in verbalization. Underpinning of SD with ORM will not only improve SD model conceptualization but also make input data reusable and transferable (from one system to another or from one organization to another).

ORM is a fact-oriented approach for modeling information and querying the information content of a business domain at a conceptual level [11]. ORM is comparable to Entity Relationship (ER) Diagrams in use [5]. It has a graphical constraint notation that is claimed to be far more expressive for data modeling purposes than, for example, Unified Modeling Language (UML) class diagrams or industrial Entity Relationships (ER) diagrams [11]. ORM takes a static perspective on the domain in the sense that it aims to capture the fact types and entity types that play a role in the (dynamic) domain, while SD takes a dynamic perspective in which the dynamic behavior of the domain is captured. When modeling ORM only, the information for the dynamic perspective (the richness of an SD model in this regard) is missing.

In figure 1 we present a logical framework where we show the strengths and gaps in both ORM and SD. We also show why grounding SD with ORM is necessary and of what impact it is to the domain of enterprise modeling and strategic decision making.

To underpin SD with a domain modeling method, we use the design science approach because it focuses on first clarifying the goals of the artifacts (which in this case is GSD) and then on building and carefully evaluating the utility of the artifacts, and to a lesser degree, their reliability and validity [12]. Design science approach further places additional emphasis on the iterative construction and evaluation of artifacts which in this case are; the GSD procedure and its resulting

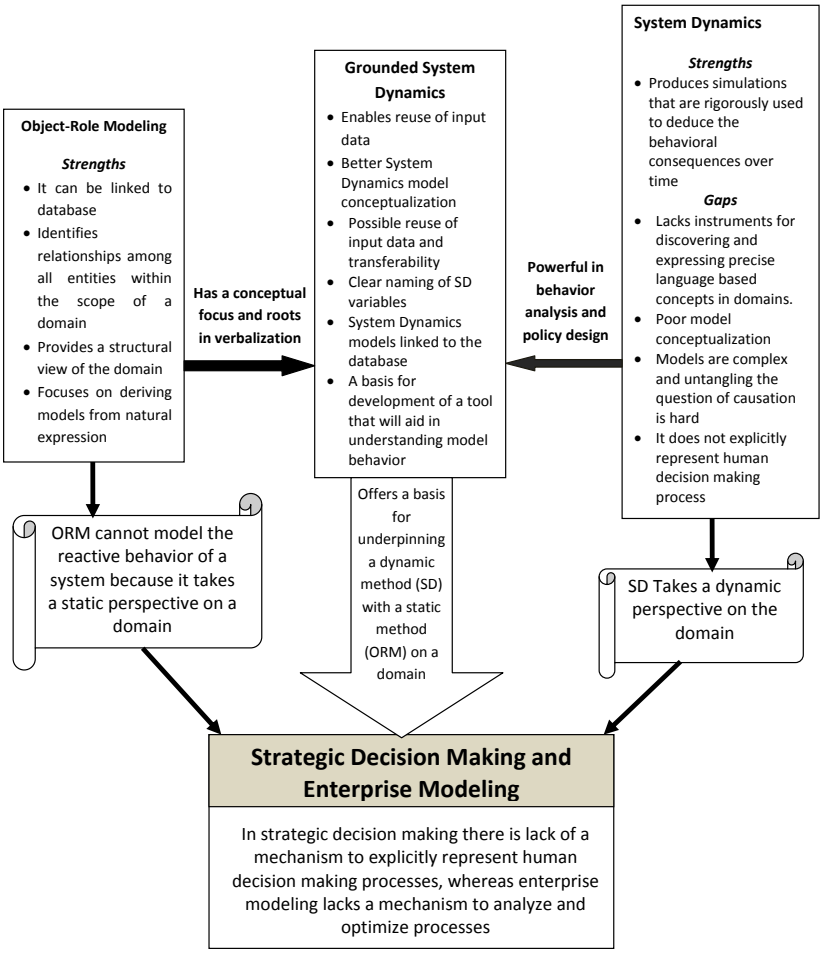

Fig. 1. Research problem and underlying principles

model(s). Design science approach also aims to ensure that the artifact (GSD procedure) is well grounded in both theory and empirical evidence to establish its validity, reliability, and practical utility. In this study we have so far identified the extent to which features of ORM static models can be transformed into SD models [26]; mapped ORM and SD concepts [27] and presented an investigation on the update behavior of the two methods [25].

The aim of this paper therefore, is to use the mapped SD-ORM concepts presented in [27] as a basis for this procedure. This procedure is what we follow while underpinning SD model(s) with a domain modeling method. We also apply this procedure (GSD) to a case where we input quantifications that lead to simulation results. These simulation results provide information about the prob-

lem domain, allow continuous testing of assumptions and sensitivity analysis of parameters [17].

The rest of the paper proceeds as follows; In sections 2 and 3 respectively brief introductions to SD and ORM are given. In section 4, a GSD outline is given and applied with the help of a case. In subsection 4.3, snapshots of some of the simulation results are presented and discussed. Finally, in section 5 conclusions plus hints for further work are given.

2 Brief introduction to SD Stock and Flow Diagrams

The system dynamics stock and flow diagram[3] elements include: stocks, flows, feedback loops (connectors or information links) and converters. In fig 2 we have stock '*Admitted patients*' depicted as a box and defined as a container (reservoir) containing quantities describing the state of the system. The value of a stock changes overtime [3] through flows. Flows can be imagined as pipelines with a valve that controls the rate of accumulation to and from the stocks. They are represented as double solid lines with a direction arrow. The arrow indicates the direction of flow into or from a stock see '*patient*' in fig 2. Flows are influenced by stocks and converters. Converters either represent fixed quantities (constants) or variable quantities (Auxiliaries). Auxiliary variables are informational concepts having an independent meaning that contain information inform of equations or values that can be applied to stocks, flows, and other converters in the model [14]. An example of an auxiliary in fig 2 is '*Patient arrival*'. Constants are state variables which do not change [3] for example *PatientName*. On the STELLA SD software which we are using, both auxiliary variables and

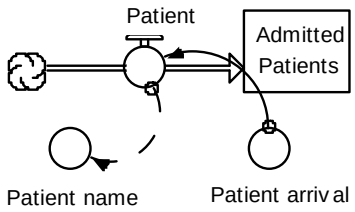

Fig. 2. SD basic building blocks

constants are depicted as small circles. Information from auxiliaries, constants and flows, is shared through connectors (information links). Two types of connectors exist, the action connectors (depicted as solid wires) and information connectors (depicted as broken wires) [24] *see* connectors from '*Patient arrival* to *patient*' and from '*Patient* to *patient Name*' respectively. These connectors

[3] For SD terminologies used in this paper we use [23] and all SD models are drawn using an SD STELLA 9.0.2 software. This is because STELLA is easy to use and offers a practical way to dynamically visualize and communicate how complex systems and ideas really work.

are immaterial and link inputs to decision function of a rate. The underpinned meaning to these connectors is that information about the value at the start of the connector influences information at the arrow tip of that connector. Connectors can feed information into or out of flows and converters but only extract information out of stocks [14]. Lastly, we have the sectors which are subsystems or subcomponents within a system. They hold/handle all decisions, stocks, information about a particular element or area and contain different information that is used in an information system. Sectors have not been represented in fig 2, but can be seen in fig 9.

3 Brief introduction to ORM

ORM basic building blocks are entity types (object types), value types and roles [10][4]. An object type is a collection of objects with similar properties, in the set-theoretical sense. Objects are things of interest, they are either entity or value types. Object types are designated by solid-line named ellipses see; *Patient* and *Labor Ward* in fig 3. Object types have reference modes, *see fig* 3, where object type *Patient* has reference mode *id*. This reference mode indicates how a single value relates to that object type. Instances of value types are constants with a universally understood denotation, and hence require no reference scheme. They are identified solely by their values, their state never changes and are designated by dotted ellipse see *Patient Name*. The semantic connections between object types are depicted as combinations of boxes and are called fact types. Each box represents a role and is connected to an object type or a value type. The roles denote the way entity types participate in that fact type. The number of roles

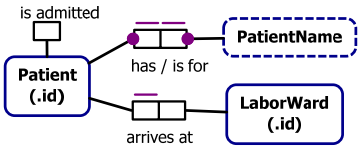

Fig. 3. ORM basic building blocks

in a fact type is referred to as fact type arity and the semantics of the fact type are put in a fact predicate. A predicate is basically a sentence with object holes in it, one for each role as depicted in fig 3 (see; *arrives at*, *is for*, *is admitted*) etc. These predicate names are written beside each role and are read from left to right, or top to bottom. It is through predicates that entity types relate to each other. Note that the constraints in fig 3 are not discussed because they are outside the scope of this paper. But these and more can be found in [10].

[4] For ORM terminologies in this paper, we use [10] and to model ORM models we use Microsoft Visual Modeler 2005.

4 Grounded System Dynamics (GSD)

After mapping the SD-ORM constructs [27], we now present a procedure which we are to follow while underpinning SD models with ORM. To achieve this, we conducted multiple discussions, read various scholarly works and in a step by step manner, we were able to come up with this procedure which we refer to as Grounded System Dynamics (GSD). GSD is a combination of two existing methods (SD with ORM). By combining these two methods we generate synergy effects by using already existing modeling methods and by so doing we overcome some of the weaknesses of SD model building. Secondly, transformation of information (data) from an ORM model of dynamic domains into an SD model is achieved. Underpinning of SD with ORM is done by following five steps which we outline is subsection 4.1. In these steps we map different SD constructs to ORM constructs and explain how they relate to one another. For a clear guide and explanation, we apply this outline to a case study of Mukono Health Center (MCH) in subsection 4.2.

4.1 GSD procedure outline

In this subsection we describe the steps followed in transforming ORM model concepts into SD model concepts.

1. Identify all possible stocks.
2. Identify all relevant flows.
3. Identify possible converters.
4. Identify possible connectors (information links).
5. Create sectors.

4.2 Case Study

At MHC we looked at the process pregnant women go through on their due dates. MHC receives an average of 250 deliveries per month. It has a total of *sixteen* maternal employees that is; *two* doctors who are available on phone in case of any emergency, *nine* nurses and *five* volunteers. There are *eleven* beds in total, available for admissions. Most of these beds are given to patients who have caesarian birth because they require a lot of attention and tend to stay longer at the health center.
The process: A patient comes to the labor ward with her antenatal card from the antenatal clinic. She queues up. Her waiting time depends on a number of factors including; her arrival time, the number of patients around and number of nurses on duty. When her turn comes, the nurse on duty takes her history and then examines her. This examination takes approximately 30 minutes. The nurse also establishes the patient's labor stage. If the patient is 4cm dilated, she is admitted to the general ward. She only returns for examination if there is any complication or after 4 hours. During this time, after every 30 minutes monitoring of the labor progress, status of the mother and cervical dilation is

done. When the patient is 8cm dilate, she is taken to the delivery room which has only two beds. While there, the nurse monitors descending of the head 2 hourly and the sticker. For normal progress the liquor is clear. When the patient has 10 cm dilate, she is ready to give birth. After delivery, she is taken back to the general labor ward. Normal delivery patients stay at the labor ward for a maximum period of 24 hours and patients who have had caesarian birth stay for a period of 4-7 days. On discharge, the baby is taken for immunization. From this given information, we now construct an ORM model shown in fig 4.

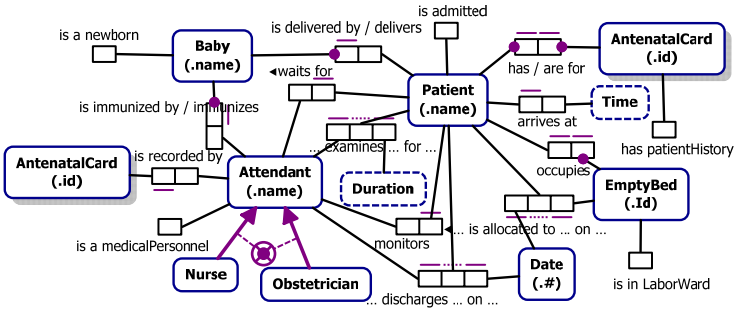

Fig. 4. MHC Labor suite developed ORM model

We use the developed ORM model in fig 4 to apply the GSD outlined procedure in subsection 4.1.

[***Step 1:***] *Identify all possible stocks*

In order for us to generate stocks, we identify an ORM element that has similar characteristics to a system dynamics stock; that is it holds items, accumulates and can be measured. The element found to have these characteristics is a *role*. But not all roles are mapped to a system dynamics stock. It's only the *unary fact types* that are mapped to a SD *Stock*. This is because they relate to one object type and contain objects from that particular object type. The total number of unary fact types therefore is equal to the total number of stocks in an SD model.

The identified unary fact types from fig 4 are; *newborn baby*, *medical personnel*, *admitted patient*, *Patient History* and *Labor ward* and are depicted as stocks in fig 5. Note that all unary fact types have words like '*is a*', '*has*' and '*is in*' at the beginning of each unary fact type predicate but when represented as stocks these prepositions are removed. This enables us make the stock name clear. Secondly, for some stocks for example; *newborn* and *admitted* we concatenate (join two character strings end-to-end) the unary fact type name with the object type name they relate with to get the stock name *newborn baby* and *admitted patient* respectively.

This makes identification of the object type connecting to the unary fact type (stock) plus its quantification easy. Note that in fig 4, each object type has a unary fact type. This may not be the case with all ORM models but we

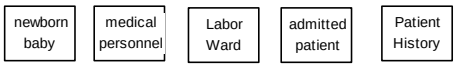

Fig. 5. Identified stocks from MHC Labor suite ORM model

advise the modeler to have a unary fact type attached to each object type so that objects in each object type have a store.

[***Step 2:***] *Identify all relevant flows*
The element identified to be similar to an SD flow in ORM is an *object type*. This is because it connects different roles. That is, for each role connection, objects held by that object type play a unique role. Flows in SD connect to different stocks and converters through connectors.

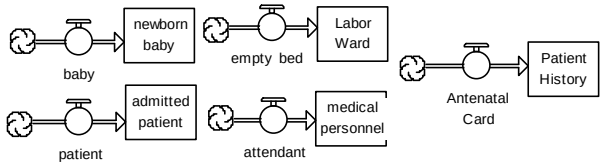

Fig. 6. Flows from MHC labor suite ORM model are connected to stocks

The identified flows from fig 4 are '*Empty bed*', '*Patient*', '*Attendant*', '*Antenatal card*' and '*Baby*' and are represented in fig 6.

[***Step 3:***] *Identify possible converters*
Converters include constants and auxiliary variables. Auxiliary variables from a conceptual point of view are informational concepts having an independent meaning. They are similar to fact types that have more than one role.[5] This is because they combine two or more variables consistently that cause change to the recipient, have an independent meaning and relate to more than one element. The roles contained by these fact types have predicate names.

We use these predicate names to name the auxiliary variables. This is done by concatenating the object type name with the fact type name. For example; roles '*examines*' and '*is examined by*' make a fact type which we refer to as '*examination*', this fact type name is concatenated to object type name *patient* giving us a flow name, '*patient examination*' see fig 7. We also include value types which we map to constants in SD. This is because a value type is identified solely by its value and it never changes its state (i.e. it is a constant). On the other hand, constants in SD are state variables which do not or change slowly [3] that they could be assumed constant for the time scope of the model. Note that the

[5] In this paper we only consider *binary* and *ternary* fact type to be similar to auxiliary variables but all fact types with more than one role are referred to as auxiliary variables

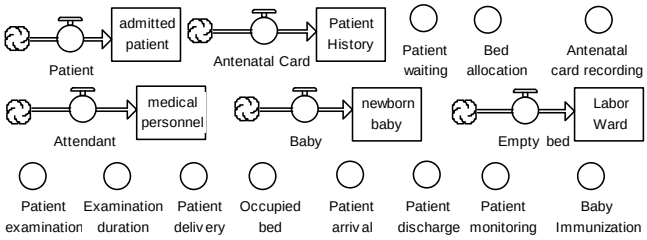

Fig. 7. Auxiliary variables identified from binary and ternary fact types

roles played by these value types are not included in the model as auxiliary because they are assumed constant.

[***Step 4:***] *Identify possible connectors (information links).*
After Identifying the converters, we now introduce the connectors. connectors are immaterial and connect inputs of a decision function to a rate. They are similar to predicators in ORM since they both act as connectors of two elements which are; object types to roles in ORM and, converters to flows and stocks in SD.

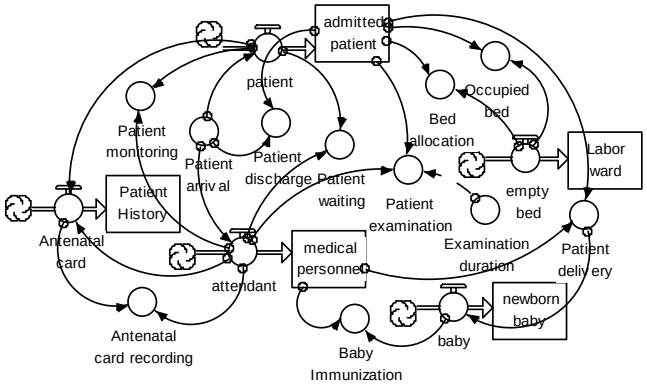

Fig. 8. Information Links introduced in SD MHC Labor suite SD model

With the help of ORM verbalization, the direction of the connectors is determined, that is from an object type to a role. Using some of the verbalization in fig 4 as examples, we show how these verbalizations help in identifying the direction of some connectors in fig 8. For example:

– We have verbalization *"Patient delivers baby"*. In this verbalization, we have two object types which are mapped to SD flows. Here our focus is on the connector connecting auxiliary '*Patient delivery*' to flow '*baby*'. The direction of the arrow is determined by the way a modeler reads from role 'delivers'

to object type (flow) 'baby'. Not forgetting that an auxiliary is made up of more than one role. Therefore, this implies that an auxiliary may have more than one connector either connecting to or from it.

– As a second example let us look at verbalization "*Antenatal card is recorded by Attendant".* In this verbalization we have object types (flows) '*Antenatal card*' and '*Attendant*'. In fig 8 we see that flow '*Antenatal card*' has a connector directional arrow facing auxiliary "*Antenatal card recording*". This is because the verbalization is read from object type '*Antenatal card*' to role 'is recorded by'. We also have another connector coming from flow '*Attendant*' to "*Antenatal card recording*" which comes from verbalization '*Attendant*' records '*Antenatal card.*

We note here that, the verbalization alone cannot give us all the necessary connectors (it can only give us links from one direction but not feedback). We therefore advise the modeler(s) to further identify the feedback.

[***Step 5:***] *Create sectors*

In [9,1] it is stated that ORM conceptual object types act as semantic 'glue'. This means that an ORM model is a network of allied object types and relationship types [9]. Sectors on the other hand are SD elements and are subcomponents within a system that handle all information about a particular element. If they are subcomponents that means they contain different elements and when put ('glued') together they make a complete system. Therefore, object types plus their 'glued' roles are similar to SD sectors because when both are 'glued' or put together they make up a complete model (either ORM or SD) and contain more than one element. The total number of sectors therefore, is equal to the total number of object types in fig 4. In each sector we have an object type plus roles connected to it. The sectors identified from fig 4 are as follows:

Sector *Empty bed* which comprises of object type *Empty bed* plus roles; '*Is occupied by*', '*is allocated to*' and '*is in labor ward*'.

Sector *Attendant* which comprises of object type *Attendant* plus roles; *'Is a medical personnel'*, *'monitors'*, *'records'*, *'examines'*, *'discharges'*, *'is waited for by'* and *'immunizes'*.

Sector *Patient* which comprises of object type *Patient* plus roles; '*is examined by*', '*is discharged by*', '*is admitted*', '*is monitored by*', '*occupies*', '*is allocated*', '*is arrives*', '*has*', '*delivers*', and '*waits for*'.

Sector *Baby* which comprises of object type *Baby* plus roles; '*is a newborn*', '*is immunized by*', and '*is delivered by*'.

Sector *Antenatal Card* which comprises of object type *Antenatal Card* plus roles; '*has patient History*', '*is recorded by*', and '*is for*'.

Here we note that, fact types that make an auxiliary variable have more than one role. Therefore, auxiliary variables relating to more than one flow are put in one of the sectors with the flow it connects too.

In conclusion, having applied the GSD procedure to a case, the modelers achieved two main things: One, better conceptualization of the underlying SD model concepts. And two, an SD model that is underpinned with an ontology domain modeling method.

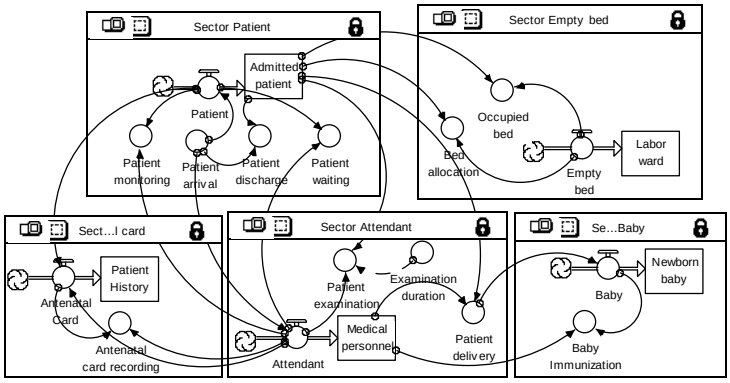

Fig. 9. Sectors identified from MHC Labor suite ORM model

4.3 Simulation Results

In this section, we use figs 10 a) and 10 b) to show some of the resulting SD model simulations. To obtain these simulation results, we started by stating the initial values for all stocks. Then defined input quantities and formulas for all auxiliary variables starting with *patient arrival* which is an input to flow *patient*. Some auxiliary inputs for example *Patient waiting duration* are captured with a time delay. This delay function returns a delayed value input using a fixed time lag. For example in fig 9, we have variable *Patient waiting duration* where its input parameter is equal to *DELAY [Attendant, 30 minutes]*. This input parameter causes variable *patient waiting* lag behind variable *attendants* by 30 minutes. This means that for the first 30 minutes of the simulation, the delay returns the initial value of *Patient* since no initial value is specified.

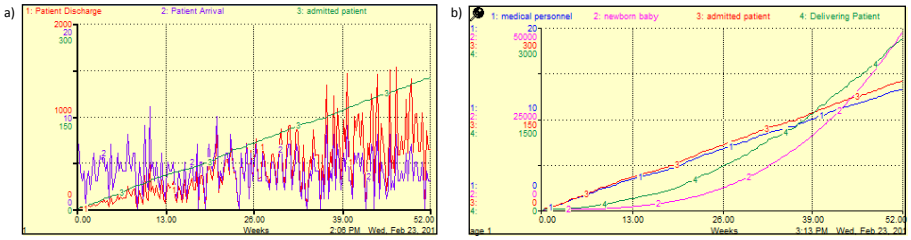

Fig. 10. Some of the simulation results of fig 9

For variables *patient arrival* and *patient discharge* in figure 10 a), we have oscillations. This is so because the quantitative measure to these variables has no defined limit. In fig 10 b), we see that over a period of time the gap between variables *medical personnel* and *admitted patients* keep on increasing. Secondly, in fig 10 b), we see that *delivering patients* are consistently increasing over a

period of time and at the 51^{st} week they rise above the *newborn babies*. The increase of *delivering patients* above *new born babies* at week 51 is an indicator that these simulation results are not completely reliable. This may be due to lack of outflows which drains the stocks.

From the input SD simulation algorithms whose choice depended on variables involved in the MHC labor suite SD/ORM model. In conclusion, the final simulation results in fig 9 do not give conclusive results. This is because some of the system dynamics construct e.g. outflows are not represented in the model. We believe that with more studies and application of the GSD procedure to different case domains, better simulation results will be achieved.

5 Conclusions and further work

This study is a work in progress and as such it is not possible to draw any firm conclusions on the end results. At the current stage it is possible to say that the GSD procedure is a good guide in transforming ORM models into SD models. This procedure gives the modeler better conceptualization of different variables plus reasons as to why a particular SD element is mapped to that ORM element.

Secondly, much as we have followed a GSD procedure to transform ORM model concepts into SD model concepts, we note here that, not all ORM and SD elements have been represented. For example; in SD we have elements like Conveyors, queues, decision process diamonds and Biflows missing and in ORM we have all constraints missing. Therefore, for further research we recommend that an investigation on how these missing elements can be used (mapped) to improve the GSD transformed SD model.

In step 2 of the GSD procedure, we realize that outflows are not represented in fig 6, yet in a system dynamics stock and flow model, a stock has both an inflow and outflow to drain the stock. Stocks accumulation is dependent on their flows and are mathematically calculated as the integration of net inflows:

$$Stock(t) \int_0^t [Inflow(s) - Outflow(s)]ds + Stock(t_0) \tag{1}$$

with $Inflows$ and $Outflows$ denoting the values of the outflow at anytime s between the initial time t_0 and the present time t [23]. The net flow determines the rate of change of any stock:

$$d(Stock)/dt = Inflow(t) - Outflow(t) \tag{2}$$

Therefore, we suggest that some auxiliary variables be referred to as out flows for example; *Patient discharge* can be used as an outflow from *admitted patient* and a new variable like *death* could be introduced in the model to act as an outflow from *newborn baby*. That way we have both inflows and outflows represented in the model.

So far, the GSD procedure has been applied to one case already and we will continue to apply it to other case studies. We will further refine the procedure and

also devote more attention to coming up with a generic meta-model. Finally, a comparison of the SD model development process with the GSD procedure using various case studies will be done.

References

1. A. Bloesch and T.A. Halpin. Conceptual queries using ConQuer-II. In D.R. Embley and R. Goldstein, editors, *Proceedings of 16th Int. Conf. on conceptual modeling*, volume 1331, pages 113–126. Springer LNCS, Los Angeles, November 1997.
2. A. Borshchev and A. Filippov. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. *22nd International Conference of the System Dynamics Society*, 2004.
3. L. Burmester and G. Matthias. Combining System Dynamics and Multidimensional Modelling - A Metamodel Based Approach. In *Proceedings of the 14th Americas Conference on Information Systems*. Toronto, ON, Canada, August 2008.
4. L. C. Chang and Y. M. Tu. Attempt to Integrate System Dynamics and UML in Business Process Modeling . In J.D. Sterman, N.P. Repenning, R.S. Langer, J.I. Rowe, and J.M. Yanni, editors, *Proceedings of the 23rd International Conference of the System Dynamics Society*. System Dynamics Society, Boston, USA, July 2005.
5. P.P. Chen. The Entity-Relationship Model-Towards a Unified View Data. *ACM Transactions of database systems*, 1(1):9–36, March 1976.
6. J. Duggan. A Comparison of Petri Net and System Dynamics Approaches for Modelling Dynamic Feedback Systems. *24th International Conference of the Systems Dynamics Society, Nijmegen, The Netherlands*, July 2006.
7. M. Elf, M. Putilova, L. von Koch, and K. Ohrn. Using System Dynamics for Collaborative Design: a Case Study. *BMC Health Services Research*, 7:123, August 2007.
8. J.W. Forrester. *Industrial Dynamics*. MIT Press, 1961.
9. T.A. Halpin and A. Bloesch. Data modeling in UML and ORM: A comparison. *Journal of Database Management*, 10(4):4–13, 1999.
10. T.A. Halpin and T. Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers, 2nd edition, 2008.
11. T.A. Halpin and G. Wagner. Modeling Reactive Behavior in ORM. In *Conceptual Modeling, Proc. of the 22nd ER2003 Conference*, volume 2813, pages 567–569. Springer LNCS, Chicago, October 2003.
12. A.R. Hevner, S.T. March, J. Park, and S. Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, March 2004.
13. D. C. Lane. The Emergence and use of diagramming in system dynamics: a critical account. *System Research and Behavioral Science*, 43:1135–1150, 2008.
14. J.D. Leaver and C.P. Unsworth. System Dynamics Modeling of Spring Behavior in the Orakeikorako Geothermal Field. *Elsevier Ltd*, 36(2):101–114, April 2007.
15. N. Melao and M. Pidd. A Conceptual Framework for Understanding Business Processes and Business Process Modelling. *Information Systems Journal*, 10:105 – 129, 2000.
16. J.D.W. Morecroft. A Critical Review of Diagramming Tools for Conceptualizing Feedback System Models. *Dynamica*, 8(1):20–29, 1982.
17. J.D.W. Morecroft. System Dynamics and Microworlds for Policymakers. *European Journal of Operations Research*, 35(3):301–320, June 1988.

18. G.N. Papageorgiou and A. Hadjis. New Planning Methodologies in Strategic Management; An Inter-Paradigm System Dynamics Approach. In B. C. Dangerfield, editor, *Proceedings of the 26^{th} International Conference of the System Dynamics Society*. System Dynamics Society, Athens, Greece, July 2008.
19. M. Pidd. *Tools for Thinking: Modelling in Management Science.* John Wiley, The Atrium, South Gate, Chichester, 2^{nd} edition, 2003.
20. G. Richardson. Problems for the future of system dynamics. *System Dynamics Review*, 12:141–157, 1996.
21. J. Scholl. Agent-based and System Dynamics Modeling: A call for cross study and joint research. In *Proceedings of the 34^{th} Annual Hawaii International Conference on System Sciences (HICS)*, volume 3, pages 1–10. Maui, Hawaii, January 2001.
22. A. M. Sharif. Industrial Viewpoint can systems dynamics be effective in modeling dynamic business systems? *Business Process Management Journal*, 11(3):612–615, 2005.
23. J. D. Sterman. *Business Dynamics- Systems Thinking and Modeling for a Complex World*. McGraw Hill Higher Education, edition, 2000.
24. K.S. Tan, M.D. Ahmed, and D. Sundaram. Sustainable Enterprise Modelling and Simulation in a Warehouse Context. In *Business Process Management Journal*, volume 16, pages 871–886. Emerald Group Publishing Limited, 2010.
25. F.P. Tulinayo, A. Groessler, S.J.B.A. Hoppenbrouwers, and P. van Bommel. Complementing System Dynamics with Object-Role Modeling. In A. Ford, D.N Ford, and E.G Anderson, editors, *Proceedings of the 27^{th} International Conference of the System Dynamics Society*. System Dynamics Society, Albuquerque, New Mexico, USA, July 2009.
26. F.P. Tulinayo, S.J.B.A. Hoppenbrouwers, and H.A. Proper. Integrating System Dynamics with Object-Role Modeling. In Janis Stirna and Anne Persson, editors, *The Practice of Enterprise Modeling*, volume 15, pages 77–85. Springer Berlin Heidelberg, Stockholm, Sweden, November 2008.
27. F.P. Tulinayo, S.J.B.A. Hoppenbrouwers, P. van Bommel, and H.A. Proper. Integrating System Dynamics with Object-Role Modeling and Petri Nets. In Werner Esswein Jan Mendling, Stefanie Rinderle-Ma, editor, *Enterprise Modelling and information systems Architectures*, pages 41–54. GI-Edition, IFIP, Ulm, Germany, September 2009.
28. Z. Zhang, L. Jai, and Y. Chai. A New Kind Methodology for Controlling Complex Systems. *International Journal of Engineering Applied Sciences*, 6(2), 2010.