# Integrating i* and process models with stock-flow models

Mahsa Hasani Sadi
Department of Computer Science
University of Toronto
mhsadi@cs.toronto.edu

## ABSTRACT

**While enterprise modeling and conceptualization frameworks mostly provide static views of enterprise architectures, they rarely address the dynamics of enterprise; i.e. the behavior of enterprise over time. This issue has led to insufficient perception of how the statics of enterprise is related to its behavior over time, and how static arrangements should be reconfigured to result into the desired behavior over time. To address this problem, herein, I propose a methodology for integrating i* models and process models (as representatives of static view of enterprise architecture) with stock-flow models (one model for addressing the behavior of enterprise over time). The proposed methodology provides one technique for developing a dynamic model of enterprise based on its static models and offers guidelines for how the static views of enterprise architecture should be reconfigured to change the behavior of enterprise over time.**

## General Terms
Modeling & Design

## Keywords
Enterprise architecture, i* models, process models, stock-flow models, dynamic models, static models.

## 1. Introduction
For managers and decision makers in organizations, there exists one thorny problem: They do not know how to move from the as-is architecture to the to-be architecture. This issue roots into two main problems:

1. They do not know how to conceptualize the as-is and to-be architecture. This consequently results into the lack of a clear perception of defining a transformation process for moving from as-is to the to-be architecture.
2. Many frameworks for the conceptualization of as-is, and to-be enterprise architectures deal with the static complexities; i.e. static structural and behavioral complexities. This is while there are a few rigid frameworks for the conceptualization of dynamic complexities of the architecture. Moreover, the conceptualization models which deal with the dynamic view of the enterprise (such as state-charts and sequence diagrams in UML), provide a low-level abstraction of the enterprise. Hence, they are not suitable for modeling high-level abstraction levels with which managers and decision makers deal. However, movement from the as-is to the to-be architecture not only deals with statics, but to a large extent is related to the dynamic conceptualization of the as-is and to-be architecture.

The lack of such conceptualization frameworks limits the understanding and addressing the issues of change by decision makers in the context of enterprise. This demands two categories of research efforts be conducted:

- Investigation of existing conceptual frameworks which deal with the dynamic complexities of systems, and integrating them with enterprise modeling approaches
- Development of conceptual frameworks which accommodates the dynamism and notion of change in enterprise as first-class concepts.

The investigation of the above two research problems, not only enables the conceptualization of dynamism and change in the context of enterprise, but also provides a framework for discussing its behavioral properties such as adaptiveness and evolution.

Herein, as a first step towards addressing the above issues, I investigate the integration of System Dynamics conceptual framework [1], with two static models of enterprise, namely i* models [2] and process models. i* models elicit the interactions between the elements of a domain under study. Process models elicit process and product flows of enterprise. System dynamics models and more specifically stock-flow models capture the dynamic behavior of enterprise over time. The outcomes of this integration are as follows:

- Proposition of one methodology for integrating the static view (statics) of enterprise with its dynamic view (dynamics). In the proposed methodology, the dynamic view of enterprise conceptualized in terms of stock-flow models are developed from i* models and process models. The integration of i* models as one representative of the interaction between the elements of enterprise with stock-flow diagrams addresses the question that how a change or rearrangement in interactions affect the overall dynamic behavior of enterprise over time.
- The above outcome provides guideline for how the reconfiguration of the as-is architecture based on its dynamic behavior.

The rest of the paper is structured as follows. In Section 2, the main ideas of system dynamics and the requirement for integrating it with enterprise architecture modeling are briefly reviewed. In section 3, the proposed methodology for integrating two static models of enterprise (i* models and process models), and one dynamic model of enterprise (stock-flow model) is delineated. Section 4 discusses the shortcomings of research

conducted herein and how it can be furthered. Section 5 ultimately concludes the paper.

## 2. System Dynamics and Enterprise Architecture Modeling

System Dynamics (SD) [1] is a methodology for understanding the behavior of complex systems over time. It provides fundamental contributions to framing, understanding, and discussing complex issues and problems. System dynamics is centered around modeling and simulating complex systems through systemic representation of the system in terms of stock-flow models. Stock-flow models capture the behavior of systems in terms of concepts of stocks, flows, information feedbacks, and valves (demonstrated in Figure 1). Stocks conceptualize the notion of accumulation in the system; inflows and outflows are the material and information entering into and out of stocks; valves regulate the amount of inflows and outflows according to the variables of information feedbacks with which they are corresponded. For example, bank balance is a stock. The inflow into this stock is deposit interest, and its outflow is withdrawal. The valves are the control decisions which regulate the rate of inflows and outflows into bank balance. One information feedback which controls the inflow rate is the variable of net income, and one information feedback which regulates the outflow rate is expenditure.

SD methods provide "essential insight into situations of dynamic complexity," especially when experimenting the real systems is impossible or not feasible. SD provides significant insights into the behavior of the system over time, but does not provide any implication on how the system elements should be reconfigured to yield a desired behavior. SD, in other words, captures the "what" of the dynamic behavior and does not address "how" the behavior of the system can be modified. On the other hand, enterprise architecture models depict how an enterprise functions. Enterprise modeling techniques provide a static image of a state of a system. Therefore, to link the "What" of dynamic behavior to the "how" of modifying it, it is required to integrate the dynamic and static models of enterprise. Correspondingly, in the following section, I delineate one methodology to address this issue. To model the static view of enterprise, I have chosen i* models, and process models. The proposed methodology provides one technique for the integration of static models of enterprise with its dynamic models. It also addresses how the behavior of enterprise over time can be modified via reconfiguring its statics.

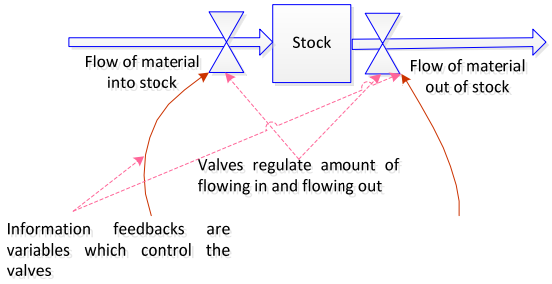


**Figure 1: The main concepts of stock-flow models in modeling the dynamics of the system**

## 3. Integrating Static Modeling with Dynamic Modeling in Enterprise Architectures

To illustrate the steps of the proposed methodology, I first describe an imaginary supply chain (extracted from a real case study of system dynamics) with three front-end suppliers named producer A, producer B, producer C, and one final producer named producer D, in section 3.1. Then, I delineate the steps of the proposed methodology for integrating i* models and process models as representative of static models with stock-flow models as representative of dynamic models of enterprise architecture in section 3.2.

### 3.1 Case Study

The static configuration of the supply chain is as follows:

- Producers A and B provides products of W and X for producer C.
- Producer C processes the products of W, and X and then produces product Y.
- Product Y is fed into the producer D, which releases the final product of Z.
- Producer D sets order for the required number of product Y from producer C.
- Producer C receives the orders of producer D and subsequently sets order for the required number of products of X, and W from producer A, and producer B.
- The decision about setting the order for product Y is taken based on its production time, and production cost; i. e. the number of orders increases as the production time and cost of product Y is decreased.
- The whole supply chain has two common goals: Increasing the number of orders, and increasing the number of produced products.
- There exist a trade-off between production time, and production cost within producer C; i.e. as the production time decreases the production cost increases and vice versa. This case also holds for all the four producers.
- Each producer tries to maximize its cash level, which relates to production cost.
    - Based on the above point, it can be concluded that each of the four producers has one individual goal: increasing their profit (cash level) which is in relation with price, cost, production time, and number of orders.

The initial setting of the as-is architecture is as follows (the performance of the current configuration of the architecture):

- For producer A, it takes one month and costs five thousand dollars to produce product W.
- For producer B, it takes two months and costs sixty thousand dollars to produce product X.
- For producer C, it takes 3 months and costs seventy five thousand dollars to produce product Y.

- In addition to the production cost for producer C, there exist operation cost for the processing of product W, and product X which is equal to five thousands.

Based on the above case study, I limit the scope of study, by studying the common goals of the supply chain, which is increasing the number of orders and increasing the number of products. These two goals can be studied by the monitoring of the profit (cash level) of producer C. Cash level is an indicator of the desired behavior of the whole system; i.e. its increase shows the balanced increase between the production and order in the whole supply chain in presence of the trade-off between time and cost of production.

Accordingly, the dynamic configuration of the supply chain is as follows:

- The desired behavior of the system is to reach the cash-level of producer C from 0 to 2100 over a two-year period.
- The parameters which are related to the decision of producer C for setting order for producer A and B are: The desired storage of W and X in C, time to adjust storage W and X in C, total storage of W, and X in C, and the orders set previously for W and X in C.
- The parameters related to the decision of producer D for setting order for Y are: C's production cost, C's production duration, and the storage of Y in D.

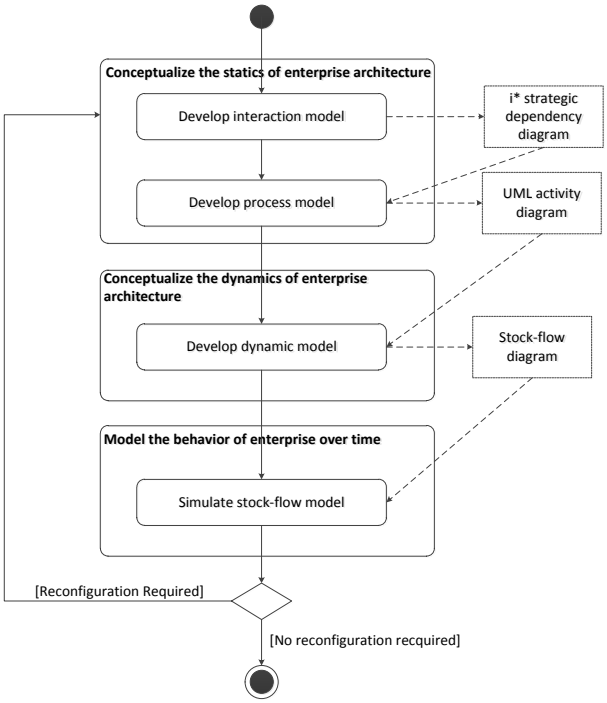


**Figure 2: The steps of the proposed methodology for integrating static modeling and dynamic modeling in enterprise architectures**

## 3.2 The Proposed Methodology

Figure 2 illustrates an overall view of the proposed methodology. The process consists of three main steps. In the first step, the statics of as-is architecture is conceptualized through developing interaction model and process models. In the second step, the dynamics of the enterprise is modeled based on the static models developed in the first step. In the third step, the dynamic model of the enterprise is simulated to observe the behavior of the enterprise over the time. Finally, based on the behavior a decision is made: If the statics of the enterprise requires reconfiguration to improve its dynamic behavior, steps one to three are repeated until the desired behavior is achieved.

In the rest of this section, I delineate each step of the proposed methodology.

1. **Step1**: In this step, I explain the application of i* framework to conceptualize the as-is architecture of the case study:

- The producers are mapped on the concept of actors and their interaction is mapped on the Strategic-Dependency (SD) relationship between them. As it is depicted in Figure 3, two types of strategic-dependencies exist between actors: goal dependency, and resource dependency. For example, actor C is dependent on producer A for the goal "W produced" to be achieved. On the other hand, actor A is dependent on actor C for the resource dependency of "Order for W" which is issued by C.

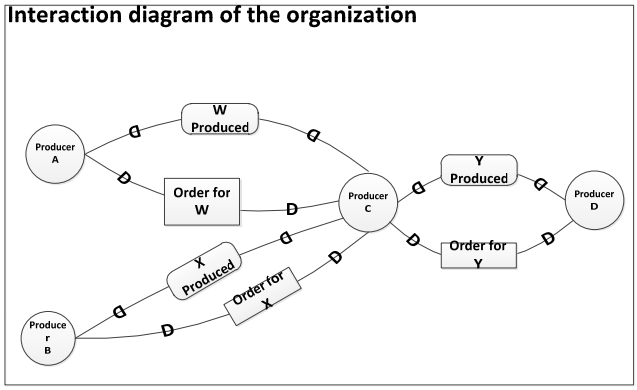


**Figure 3: The conceptualization of the as-is architecture in terms of strategic-dependencies**

**Why is i* appropriate?**

- This form of conceptualization is important from the perspective of a value network in which a group of actors are in interaction and cooperation with each other to achieve a common goal. In other words, because the property of interest is the behavior of the whole system over time, which can be defined as a common goal which is achieved by the interaction of different actors in the system over time, i* serves an appropriate conceptualization framework. More specifically, to reason about global properties of interest (the overall performance of the architecture) in terms of cost or time in the case under study, the different type of interaction between the elements of a system and the part they have in the global property is required to be elicited. This is necessary for the reconfiguration of as-is architecture.
- i* abstracts away the complexity of systems statics by focusing on the different interactions between the elements of the system which are directly related to and affect the behavior of the system.

**[Remark1]** It should be noted that the interaction between actors in i* also include non-functional and task dependencies. However, for the purpose of integration of i* with stock-flow

models, goal dependencies, resource dependencies, and non-functional dependencies are of interest. This is because that in the stock-flow models, the stocks are representative of quantitative and measurable features of the system. In i* models, goal-dependencies and resource dependencies can be directly related to quantifiable measures via various goal break-down and indicators assignment techniques [3]. Moreover, soft-goal dependencies can also be indirectly turned into quantifiable measures via methods such as AHP [2].

**[Remark2]** Having depicted the case study in Figure 3, this question raises that why the interactions related to the provision of products are mapped onto goal-dependencies, while the interaction related to the order settings are mapped onto resource dependencies. This is mainly due to the following reason:

- The product provision dependencies are course-grained interactions. Hence, they should be concretized into fine-grained details of processes and work-products between them to be integrated with stock-flow models. However, order setting interactions capture the informational interactions between the actors which can be directly related to stock and flow models. Although the goal dependencies and the resource dependencies are of different level of granularities, they show two important aspects of interactions between actors which should be captured in one model.

Although i* models depict the abstract interaction between different elements of the system, it lacks the detail to be directly integrated with stock and flow models. Stock flow models provide a process-oriented view of the system (the inflow, stock, outflow view). Hence, process models are appropriate model to connect i* models with stock and flow diagrams. In this paper, I use activity diagrams for the purpose of integration.

2. **Step 2:** In this step, each SD dependency identified in the previous step, is concretized into activities and the artifacts which are transferred between them. Based on the architecture which is depicted in Figure 3, the following steps should be taken:

a. The goals should be concretized into actions versus goal decomposition method, which leads directly to the identification of the corresponded activities.

b. The artifacts (work-products), which are communicated between each activity identified in the above step, are also elicited. The work products of importance are those influencing the measurement of the goal achievement. Correspondingly, I refer to these work-products as material flows; i.e. the materials that are communicated between the activities and their quantities are of importance. According to the case study, the goal dependency of "W Produced" between A, and C, is refined further into the activity of "Produce and deliver W" and the material flow of "Product W". The process model related to the goal dependencies of Figure 3 is depicted in Figure4.

c. The resource dependencies are directly concretized into work-products. Moreover, the processes or activities for which the resources are input or output are also elicited. As depicted in Figure 3, the "Order for Y" work-product has been elicited with the two processes, which produced and consume this information. This mapping leads to the identification of information flows in the process model.

**Why process models?**

Process models capture a static model of the information flows and material flows in the as-is architecture. As it is conveyed from the concept of flows, they are required for the development of a dynamic view of the system. Moreover, the process-oriented view of the activity diagrams identifies the direction of material and information flows which are required for the development of stock-and flow models.

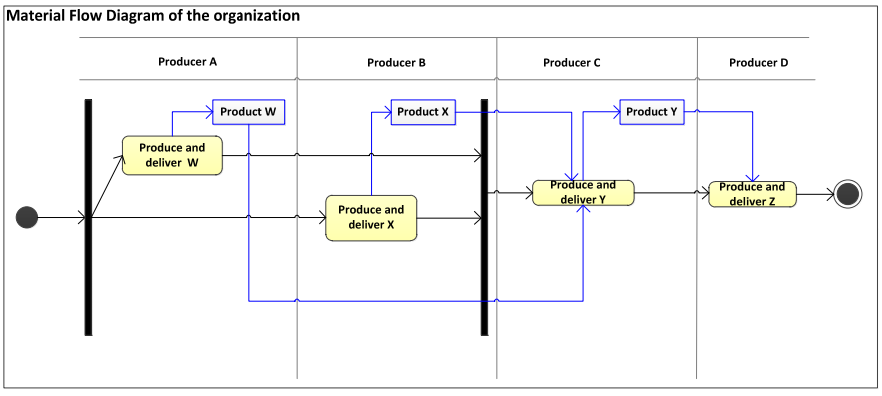


**Figure 4: The identification of material flows in the as-is architecture**

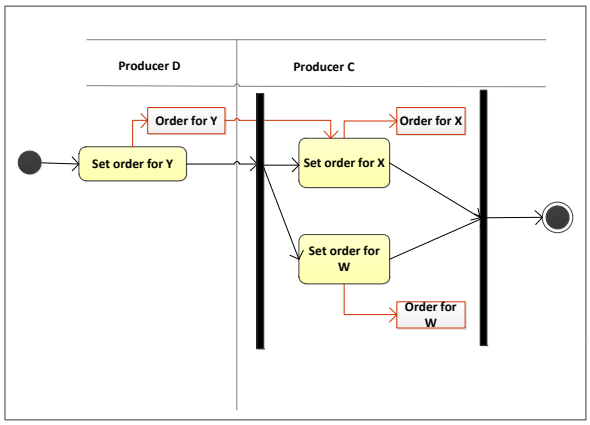


**Figure 5: The identification of information flows in the as-is architecture**

**What do the two process models of Figure 2 and Figure 3 depict?**

They depict the two parallel work-flows which are in run in the communication in parallel with each other. The work-flow of material provision (Figure 4) is initiated by producer A and B, and then continued by producer C, and D. However, the work-flow of information provision (Figure 5), is initiated by producer D, and then continued by producer C, B and A, respectively. The two process models together show the interconnected of the interactions depicted in Figure 5. While producer C produces product Y, at the same time, it may receive the orders for Y from producer D.

**[Remark3]** One issue which is raised in the conceptualization of the as-is architecture, is the level of abstraction and concretization of the activity diagrams developed. As it is depicted in Figure 4, and Figure 5, I have concretized the interaction diagram of Figure 1, just by connecting the activity of "Produce and deliver W" to the goal "W produced". This process itself consists of many sub-processes, product flows and information flows, which are abstracted away. The purpose of the development of the models of as-is architecture is to conceptualize the overall statics and dynamics of the as-is architecture. Hence, at this step, this level of details suffices. In subsequent steps, in which decisions should be made about the reconfiguration of the as-is architecture, there may be a need to develop more concretized versions of process models.

3. **Step 3**: In this step, the properties of interests which affect the behavior of the enterprise over the time are identified in i* models and process models. As stated in the description of case study, I study the cash-level property of producer C over time. For this purpose, two properties of production time and cost should be elicited in interaction diagrams and process models. This will help in modeling the overall performance of the as-is architecture. As illustrated in Figure 4, time and cost properties has been attached to the goal dependencies of interaction diagrams, and subsequently propagated to the processes which realize the goal dependencies in Figure 5. For example, the interaction between producer A and producer C in order to produce product W, takes a duration of 1 months and costs 5000 $. Based on the interaction diagram and the process model the overall performance of the architecture can be reasoned about. Based on the critical path identification, the overall time it takes for the product Y be ready and fed into producer Z is 5 months:

   *Overall time performance = Max {Time for W be produced, Time for X be produced} + Time for Y be produced = 5 months*

   ; and the overall cost performance of product Y is 145000$.

   *Overall cost performance = Cost of W be produced + Cost of X be produced + Cost of Y be produced + Operation cost of C = 145000$*

**Why step 3 is required?**

To evaluate the as-is architecture and decide about the to-be architecture, decision criteria are needed. In order to decide about the reconfiguration of the as-is architecture specified in terms of interactions between actors, the value and performance of each interaction (dependency) is required. Accordingly, the criteria of interest, which define the value of each interaction, are identified and elicited in this step. This serves as a basis to decide where the as-is architecture can be intervened to be reconfigured.

**Why both interaction diagrams and process models are annotated?**

Although the identification of the overall performance, and the specification of the value of the interactions between actors can be done sufficiently on interaction diagrams, however, to decide about the reconfiguration, the details of the interactions which lead to the overall performance of the as-is architecture are required to be traced. This requires the identification of the properties of the processes and activities, which affect the performance of the interactions. Accordingly process models (activity diagrams) are developed. [Due to the simplicity of the case under study, no specific information is added to the interaction diagram in Figure 6B].

**Why moving to step 4?**

Based on the performance criteria, the as-is architecture can be reconfigured. For example, the duration of the interactions or the costs can be reduced. This raises this question that why moving to step 4 or connecting the statics (static view) of the as-is architecture to its dynamics (dynamic view) is required. This is due the fact that the following questions are raised:

- Why do we want to increase the overall performance of the as-is architecture by reducing the overall time or cost of interactions?
- Why do we want to reduce the production time of product A, or its production cost?

To reason about the answer to these questions, having the static view of the architecture is not enough. We therefore require more information about the as-is architecture. One particular way, in which we can reason about these questions is to link the statics of the as-is architecture to its dynamics. The dynamic view of the as-is architecture deals with the behavior of the system over the time. This is while many regulations which are performed in the system, are based on its dynamic performance. Hence, if the statics of the as-is architecture are linked and connected to its dynamics, the influence of the static configuration on the overall behavior of the system during the time can be evaluated and judged about. This also provides insight about how to reason about the reconfiguration of the as-is architecture to form the to-be architecture. Accordingly, in step 4, I explain how, the stock-flow model of the as-is architecture is constructed based on interaction model and process model.

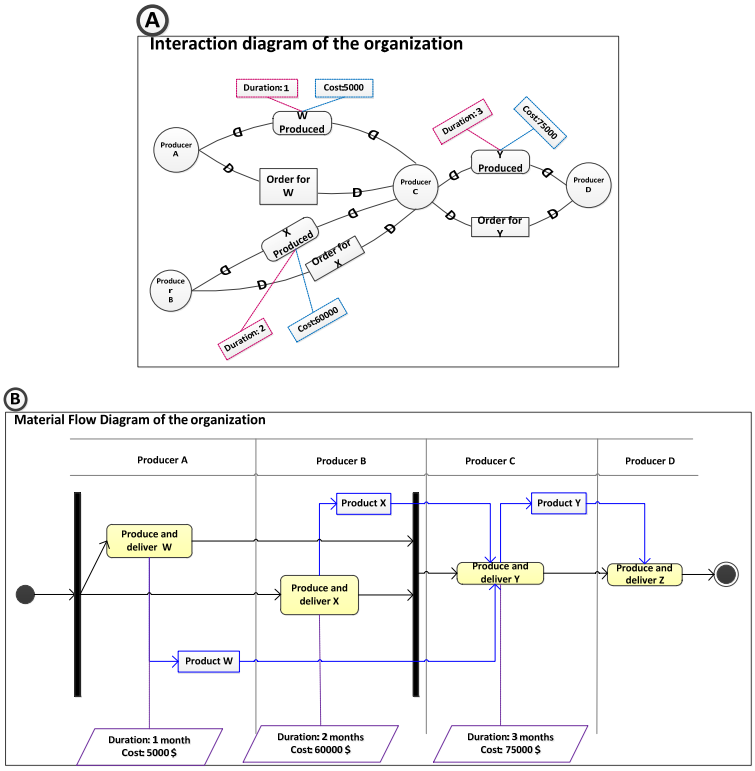


**Figure 6: Identification of the static properties of interest which influence the dynamic behavior of the as-is architecture on A) interaction model B) process model**

4. **Step 4**: To develop stock-flow models, the following steps are taken. [The basis for the development of stock-flow models is the process models developed based on interaction model.]
   - Each actor becomes a separate sector in stock flow model.
   - The material flows and information flows identified as work-products in the process models are mapped on the accumulation (stocks) in the stock-flow models. For this purpose:
     - The information flows identified in the process models are mapped onto stocks and assigned to the sector of the recipient actor. For example, as demonstrated in Figure 5, the information flow of "Order for W" is set by Producer C, and received by producer A. Hence, a stock named "Order for W from C" is assigned to sector "Producer A", in the stock-flow model depicted in Figure 7.
     - The material flows identified in the process models are mapped onto stocks assigned to the sector of the recipient actor. For example, the material flow of "Product W", in Figure 4, is mapped onto a stock named "C's inventory of W", and assigned to the sector "Producer C" in Figure 7.
   - The flow directions for material flows and information flows in the process models are mapped onto the inflows and outflows of stocks through following two steps:
     - As stated earlier, the two workflows of Figure 4 and Figure 5 are in run parallel with each other. Although this parallelism does not influence the statics of the as-is architecture (therefore it is not shown in the process models), however, it influences the dynamic behavior of as-is architecture. Therefore, in the development of the stock-flow models this point should be considered. For this purpose, each stock related

to an information flow is paired with its relevant stock related to a material flow. For example, the stock of "Order for W from C" is paired with "C's inventory of W". Pairing is done via and out flow, and an inflow; i.e. the stock "Order for W from C" has an outflow which is the inflow of the stock "C's inventory of W". Based on this step, the outflow of an information stocks, and the inflow of the material stocks are identified.

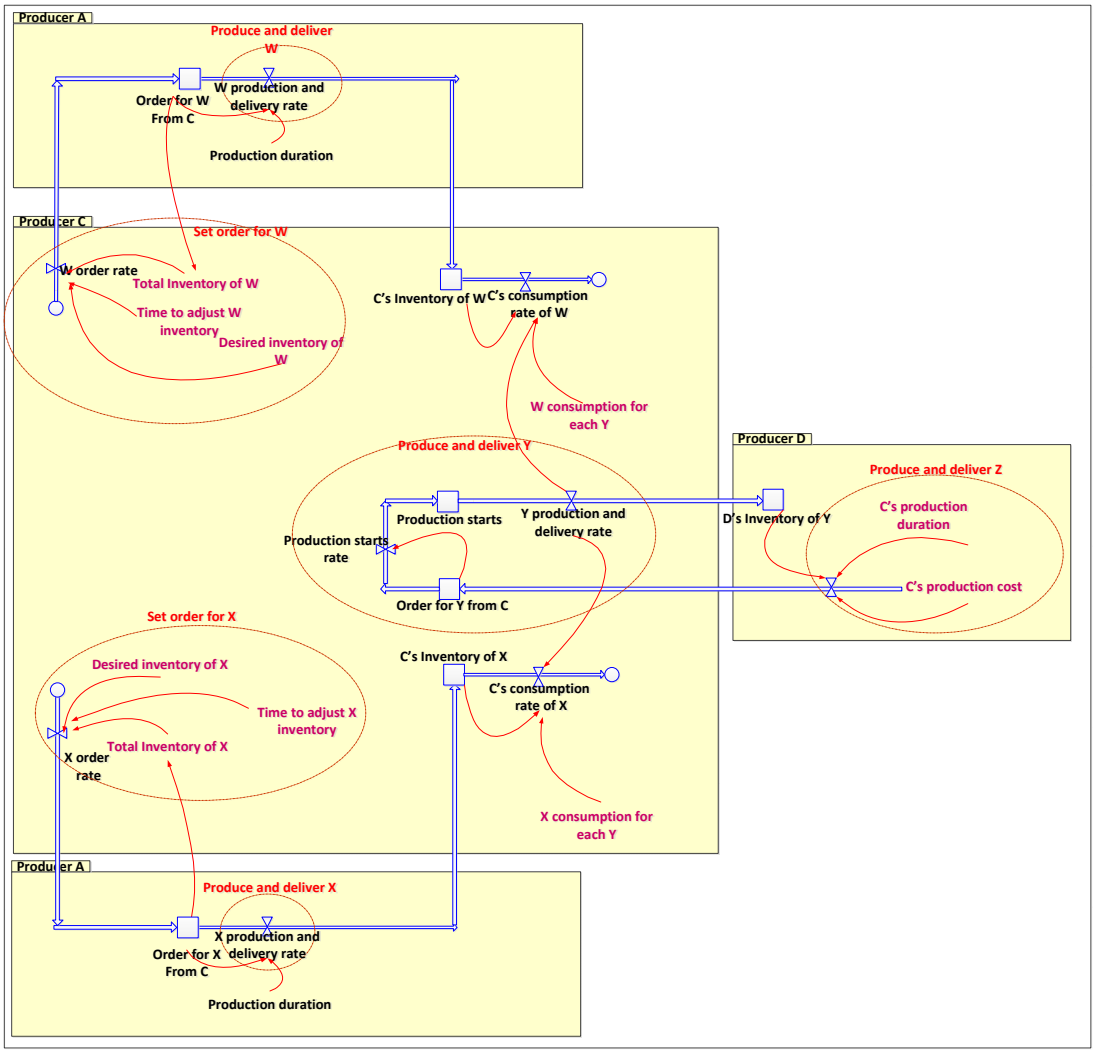


**Figure 7: Stock flow model of the supply chain developed from the process model**

- The inflow of an information stock and the outflow of material stocks are connected to exogenous variables; they are not related to any other stocks. This is due to the fact that the stocks which are connected to the input of the information stocks and the output of the material stocks do not belong to the space of process models (space of design of the as-is architecture), but rather they belong to the space of interaction models ( space of decision making and regulation). Hence, they are connected to exogenous stocks (out-side model stocks), and not endogenous stocks (in-side model stocks).

- The processes in the process models are corresponded to rates (valve) in the stock-flow models. For this purpose the following steps are taken:
  - In each pair of information and material stock, a valve is considered, and the process related to the material flow is mapped on it. For example, in Figure 7, between the "Order for W from C", and "C's inventory of W", a valve is considered which is the map of "Produce and deliver" process in Figure 2.
  - For each outflow from a material stock, and inflow to an information stock, a valve is considered, which is the mapping of the processes in the relevant information and

material process models. For example, in Figure 7, the valve related to the input flow of "Order for W from C" is the process "Set order for W", and the valve related to the output flow from "C's inventory of W" is "Consume W".

- In the previous steps, I explained the construction of the portion of stock-flow models which are related and are mapping of the statics of the as-is architecture. However, dynamic view of the as-is architecture has more information than the statics view. This information is captured in stock-flow models by the element of information feedback. Information feedbacks which are connected to the valves identify the variables which influence the regulation of the inflows and outflows in the architecture. In order words, to regulate the rate of input and output to each stocks, a decision is made based on information feedbacks. These information feedbacks come from three sources:
  - relevant stock related to the valves (rates),
  - other valves in the stock-flow models,
  - auxiliary variables which are neither stocks nor variables.

  According to the dynamic configuration of supply chain explained in the case study, the information feedbacks related to each valve are identified in Figure 5. For example, the variables, which influence the regulation of order rate for W in sector C are desired inventory of W in C, time to adjust W inventory in C, and total inventory of W in C. Moreover, the variable total inventory of W in C is influenced by the previous orders set for W from C.

**[Remark4]** As it is conveyed from step 4, the information feedbacks elicited in Figure 7 are developed based on the dynamic configuration of the supply chain and are derived from neither the interaction model of the supply chain nor its process models.

- In the development of stock-flow models, the last step is to model the cash-level of producer C which is an indicator of the overall behavior of the system over time and is related to the information feedbacks elicited in the stock-flow model. Similar to the previous step, the information feedbacks influencing the cash level are neither derived from the interaction models nor from the process models, and are identified externally (depicted in Figure 6).

5. **Step 5**: Stock-flow models are supported by simulation tools which enables the simulation and prediction of complex systems over time. The open-source tool developed for this purpose is Stella [1]. Correspondingly, in this step, the stock-flow model of the enterprise (Figure 7), and cash-level (Figure 8) are simulated in Stella. The result of this simulation for the property of interest (cash-level) is depicted in Figure 9. As it is depicted, with the current static configuration, the cash level reaches to 850K$ over two years which is far from the desired behavior.

**[Remark5]** For the simulation of stock-flow models, the information feedback variables require quantitative values. In this report, I have set the variables to predefined values, which remain the same for all the simulations. Since the information feedbacks are not derived from the static models, hence setting them to a predefined value and considering the invariable during simulations is equivalent to having a fixed policy for controlling the dynamic behavior of system.

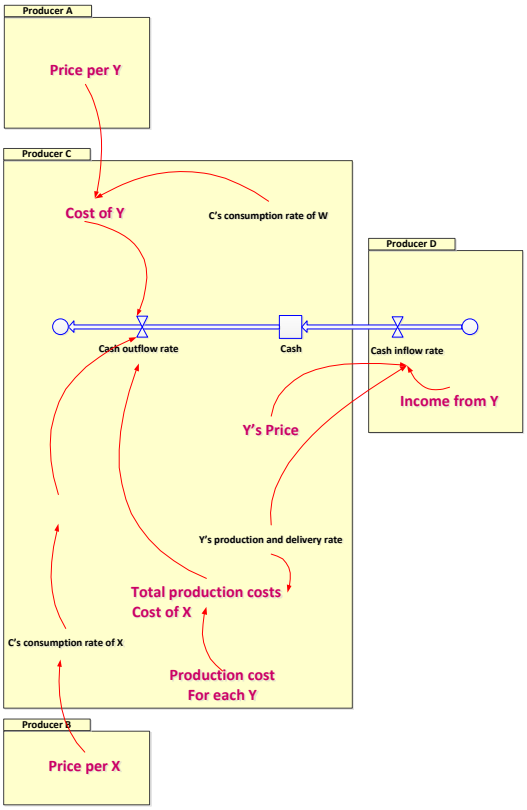


**Figure 8: Stock flow model of cash level (indicator of the behavior of the system)**

The simulation provides insight into how the static configuration of enterprise contributes to the behavior of enterprise over time. Accordingly, when there is a problem in the simulated behavior, it raises this questions: If the dynamic policy of the system does not change, how the statics should be reconfigured to keep up with the desired dynamic behavior.


**Figure 9: The dynamic behavior of enterprise architecture over a two-year period**

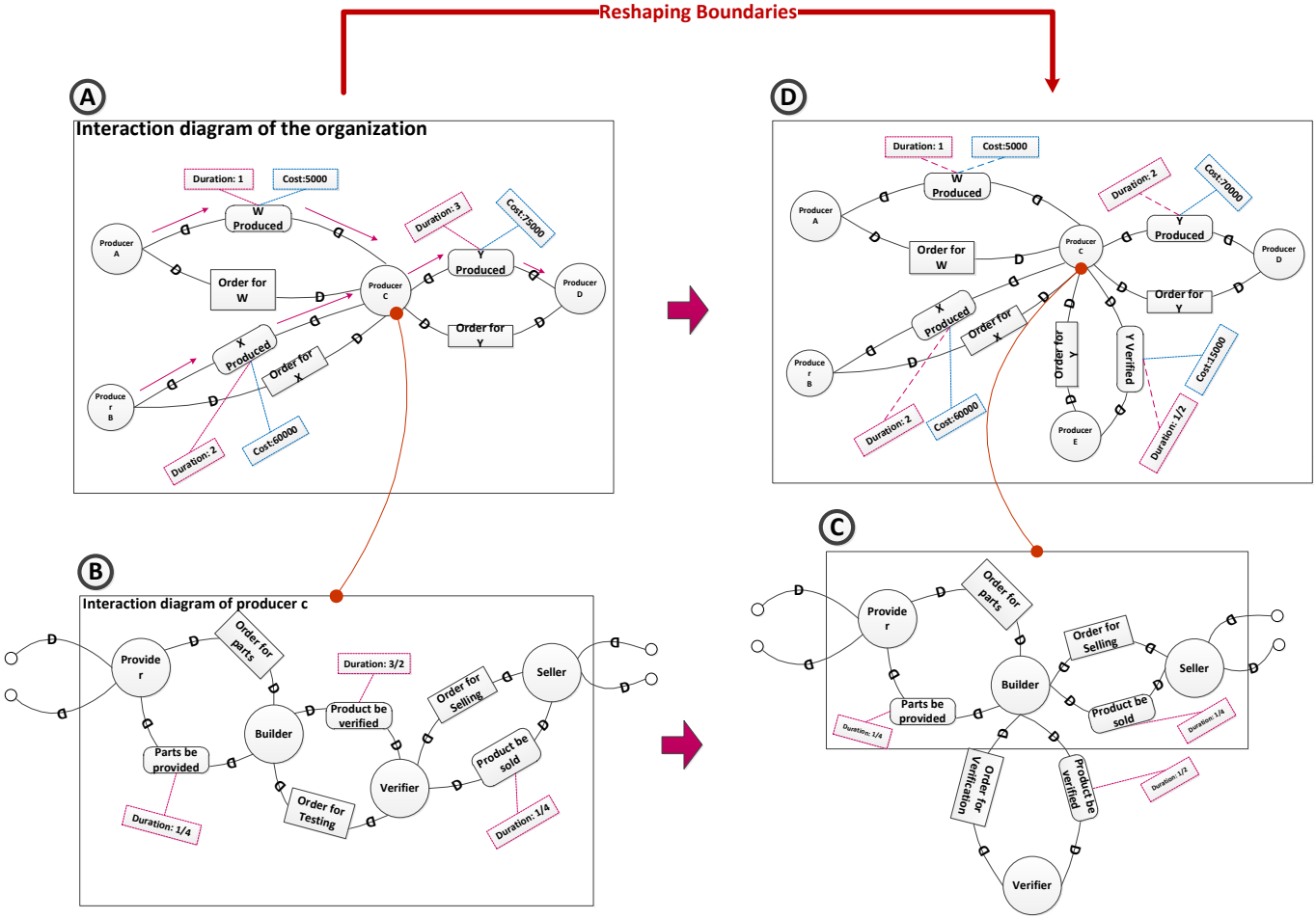


**Figure 10: The steps required for reconfiguring the as-is interaction diagram to the to-be**

**6. Step 6**: As it is observed from Figure 9, the current behavior of the system over time does not meet the desired behavior. Hence, the as-is architecture is required to be reconfigured. For this purpose, step 1 to step 5 should be repeated until a configuration of the as-is architecture is found which meets the desired behavior of the system. Since the cash-level is corresponded to two static criteria of time and cost in the static configuration of as-is architecture, the interaction model should be reconfigured to improve both or either of these criteria.

The overall steps of the reconfiguration of the as-is interaction in the case under study is depicted in Figure 10. To improve the performance of the as-is architecture, the criterion of time has been chosen. As it is conveyed from Figure 10-A, the main time-consuming interaction in the as-is architecture is the goal dependency of "Y produced" which takes three months. Therefore, the realization of this dependency should be reconfigured. For this purpose, the interactions taking place inside the actor of producer C has been elicited and depicted in Figure 8-B. For the goal-dependency of "Y Produced" be realized by producer C, four actors of Provider, Builder, Verifier, and Seller interact with each other inside producer C. The production time of 3 months results from the time performance of the interactions between these actors. Hence, the interactions between these actors should be reconfigured. As depicted in Figure 10-B, the most time-consuming interaction is the interaction between Builder and Verifier for verifying the products which takes 3/2 months. To resolve this issue two possible decision can be made:

a) Continuing with the elicitation of the interaction model taking place inside the Verifier and further examining the realization of its configuration to improve the time performance.

b) Reconfiguring the interactions of the current level of abstraction to improve time performance.

As depicted in Figure 10-C, the second decision has been made; i.e. instead of verifying the products inside Producer C, this interaction and its relevant actor has been excluded from the interaction model of producer C, and has become an independent actor similar to producer A, and producer B. (In other words it has become outsourced). Via outsourcing the time required for verifying product Y reduces to ½ months. Since this solution seems feasible. Therefore, this configuration is considered as a candidate solution. Subsequently step 2 to 5 of the proposed methodology are performed; i.e. the relevant process models and stock-flow models are altered and developed, and finally the

behavior of the new configuration is simulated (Figure 11). The simulation of the behavior is depicted in Figure 9. Since with the candidate solution of reconfiguration the overall behavior of the system reaches to its satisfactory level (the cash level reaches to 2100 over two years), the as-is architecture is reconfigured to the candidate solution which is the to-be architecture (shown in figure 10-D). Otherwise, steps 1 to 5 would have been repeated.

As it is observed in the process of reconfiguring the interaction models from Figure 10-A to Figure 10-D, the boundaries of responsibility of producer C has evolved. In Figure 10-A and B, the interactions taking place within the boundaries of producer C are more, while the interactions of producer C with other actors are less in comparison with their counterparts in Figure 10-C and 8-D.

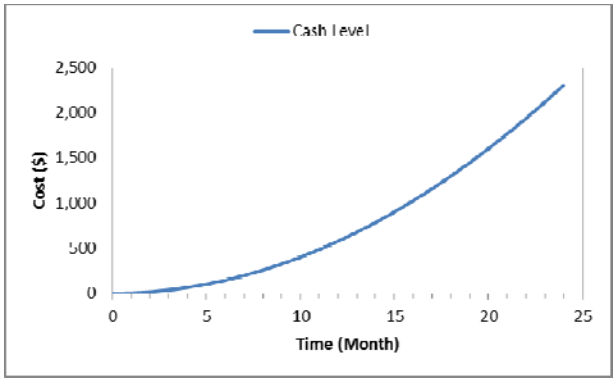


**Figure 11: The dynamic behavior of enterprise architecture with the candidate reconfiguration solution**

## 4. Discussion

In this section, I discuss the limitations and shortcomings of the research reported herein which require further investigation and research.

- In this report, I developed a methodology for the integration of i* strategic-dependency models, process models, and stock-flow models based on a cases study representing the statics and dynamics of a supply chain. Supply chains are the most common examples for the application and analysis of stock-flow models. Therefore, the developed case study covers the main issues raised related the stock-flow models. However, other cases are required to be developed to examine the validity of the proposed methodologies.

- In the proposed methodology, i* models and process models are chosen for modeling the statics, and stock-flow models are chosen for modeling the dynamics of the enterprise. This raises the questions about the appropriate selection of static models, which are most suitable for integrating with the dynamic aspects. Although, throughout this report, the selection of i* models and process models are discussed and justified, however, they are not compared with other static models. Moreover, there exist other conceptual frameworks which conceptualize the dynamism of the systems. This raises the issue of appropriate selection of the dynamic conceptual frameworks which also needs further investigation.

- The proposed methodology for the integration of i* models with the stock-flow models should be validated and further improvements should be explored. More specifically, the proposed methodology has one specific limitation which is important: Although most parts of the stock-flow diagrams are directly developed from the static models, for the development of the dynamic view of systems more information is required than there exist in the static models (The information feedbacks are not derived from the static models). This raises this question if other static models should be involved to completely build the dynamic view of the system from its statics. Moreover, since the information feedbacks are not derived from the statics they are assumed to be pre-defined. However, if the information feedbacks can be developed based on other static models, then reconfiguration the static part of the enterprise may also affect the information feedbacks (This point is not considered in this report).

- From the static models, the strategic dependencies of i* models are employed to model the different types of interactions between the elements of the enterprise architecture. Therefore, the concept of intentionality which exists in i* are not fully exploited. This raises the following question: Does the incorporation of the notion of intentionality of individual actors influence the dynamic behavior of the enterprise over time, or over long run? An interesting issue which exists regarding this question is that although each actor is mapped on a sector in the stock-flow models and is in control of a portion of the dynamism of the system, however, the interconnected of the information feedbacks in the stock flow models may neutralize the influence of the intentionality of each actor over time. This points to one other limitation of the current work: In the proposed methodology, it is assumed that the behavior of the system is decided globally and not locally by individual actors.

- i* is a problem structuring method which appropriately addresses the issue of assignment and reassignment of responsibilities between different structural elements of the as-is architecture. Integrating i* models with the dynamic behavior of the system can be used for the evaluation of structure of the as-is architecture based on its behavior. This issue requires further investigation.

## 5. Conclusion

In this report, I presented a methodology for integrating two static models of enterprise architecture, namely i* models and process models, with one modeling framework addressing the dynamics of systems, namely stock-flow models. The proposed methodology provides one technique for constructing a dynamic model of enterprise from static models, and addresses how the static models can be reconfigured to result into the desired dynamic behavior of enterprise over time. Although in the proposed methodology, stock-flow models are constructed from i* and process models, the construction process cannot be referred to as transformation. This is because that the dynamic view of stock-flow models contains more information and concepts than what is provided in i* models and process models. Accordingly, the mapping process is not complete. The next step of this research is promoting the proposed integration methodology to a transformation methodology.

## 6. References


[1] Sterman, J.D., Business Dynamics, Irwin/McGrow-Hill, 2000.

[2] Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. Social Modeling For Requirements Engineering, MIT Press, 2010.

[3] Barone, D., Yu, E., Won,J., Jiang, L., and Mylopoulos, J., "Enterprise Modeling for Business Intelligence", *In proceedings of IFIP International federation for Information Processing (PoEM 2010)*, LNBIP 68, pp. 31–45, 2010.