Transdisciplinary Lifecycle Analysis of Systems
R. Curran et al. (Eds.)

doi:10.3233/978-1-61499-544-9-357

A Guideline for Adapted System Dynamics Modeling of Rework Cycles in Engineering Design Processes

Elisabeth SCHMIDT[1], Daniel KASPEREK and Maik MAURER
Institute of Product Development, Technische Universität München, Germany

Abstract. A substantial variety of rework cycle system dynamics (SD) models that are capable of simulating the influence of rework on project parameters exists in the literature. Although all of them use the rework cycle concept, these models vary in their structure and quantification as they are adapted to capture specific process features. The difficulty of grasping the variety of diverse rework cycles and the variability in modeling goals are obstacles for modelers in finding the right model. The aim of this article is to provide a guideline that will help developers create adapted rework cycles for engineering design processes. Through literature research of different rework cycle SD models, specific elements are classified in a guideline. With the knowledge of a variety of rework cycles SD models and their capabilities, recommendations for developers can be made on how to build a model that captures the necessary requirements for their research focus.

Keywords. Engineering Design Process, System Dynamics, Rework Cycle

Introduction

Engineering design processes (EDP) are characterized by their dynamic and creative behavior and their unpredictable results. For the planners and managers within companies, it is interesting to learn more about the dynamic process behavior in order to distribute resources appropriately as well as for cost and schedule calculation [1].

The use of SD provides a way to simulate processes and thus enables the modelers to foresee the process behavior. In order to best reproduce the process behavior, modelers include certain features in their models to reflect special process characteristics. This strategy allows for the pursuit of various research objectives and will be referred to below as adapted modeling.

Table 1. Purposes of rework cycles allocated to the references.

Purpose of rework cycle	References
Phase concurrency	[2-13]
Human factors	[4; 12-18]
Staffing	[4; 12; 14; 16-21]
Outsourcing	[17]
Testing	[8; 22]
Tipping point	[21; 23]
Cost and schedule foresight	[24; 25]
Process improvement	[26; 27]

Table 1 gives an overview of the possible purposes of rework cycles that can be found in the SD literature. Developers may include structures to capture impacts of project staffing, phase concurrency, testing processes and more. There are rework cycles

[1] Corresponding Author, E-mail: schmidte@mit.edu.

which pursue more than one of the listed purposes in Table 1 and thus contain several structures to create a simulation of these features. Depending on the selection of purposes, the rework cycle needs to be adapted to enable the simulation of these features. Adaptations of SD models vary in their size, structure and quantification based on their application.

Due to the variety of rework cycles and variability in modeling goals, modelers have difficulties finding the right model for their process. A guideline that is specifically directed to this modeling problem cannot be found within the literature research of this study. Therefore, we propose a guideline that supports modelers in choosing and adapting existing rework cycle concepts for their particular needs.

1. Literature-based guideline for adapting rework cycles

The developed guideline presents different structures that modelers use to technically implement certain behaviors of rework cycles. These structures are referred to in this paper as adaptations. The adaptations are added to a basic rework cycle model in order to generate a model that simulates the considered EDP more accurately. The adaptations are summarized in the adaptation scheme shown in Figure 1.

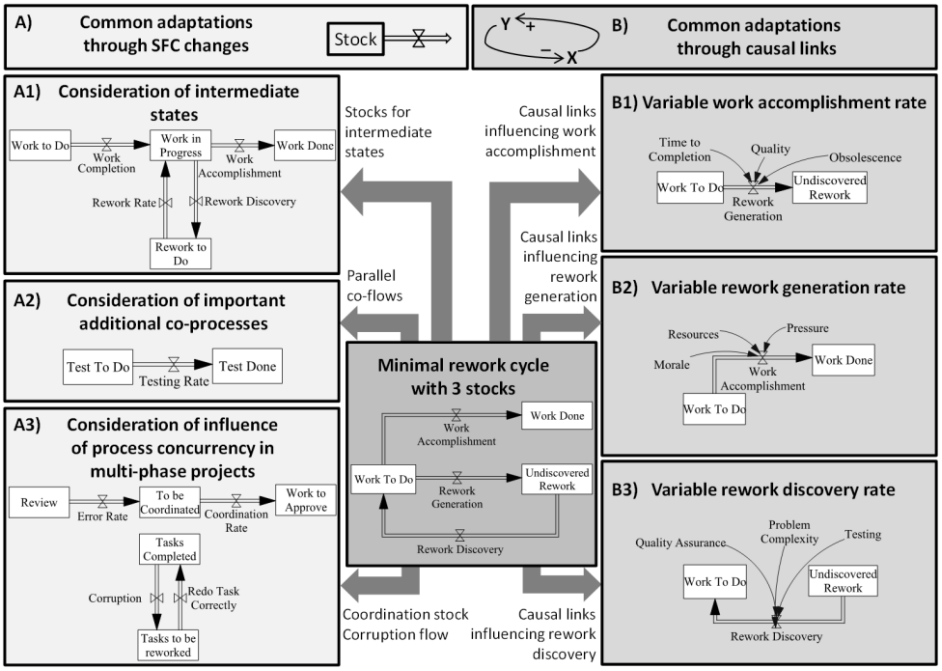

Figure 1. Rework cycle adaptation scheme. Based on the minimal rework cycle there are two ways to adapt the rework cycle – SFC adaptations and causal link adaptations.

The adaptation scheme is built on a basic rework cycle model. In Figure 1, this model is located in the middle. It is characterized by its simple structure that contains the minimum number of independent stocks to model processes with rework [22]. Moreover, the model is considered to be simple because of its constant rates.

The simple model in Figure 1 is often expanded to achieve certain characteristics. The expanding of the model is technically implemented either by means of additional stocks and flows (A) or influencing causal links (B).

Rework cycles can be expanded by adapting the SFC (stock and flow construct) by means of adding stocks and flows (Figure 1, left). The adaptation scheme divides the SFC implementations into three categories:

A1) Consideration of intermediate states

A2) Consideration of important additional co-processes

A3) Consideration of influence of process concurrency in multi-phase projects

Another means for adapting rework cycle SD models is causal link expansions (Figure 1, right). Certain process behaviors can be represented by adding variables and the dependencies between the variables. In most cases, additional variables are used to influence the rates of rework cycles. Using the initial model in the middle of Figure 1, there are three rates that are affected by the values of other factors:

B1) Variable work accomplishment rate

B2) Variable rework generation rate

B3) Variable rework discovery rate

Alongside these six ways of adapting system dynamic models, there are also other ways to model certain features which do not fit into one of the presented categories (e.g. in [20; 26]). However, these adaptations only appear in sporadic cases and thus are not included in the scheme. The SFC and causal link adaptations – which were assessed as appearing frequently within the research of this study – are explained in the following.

1.1. Consideration of intermediate states A1)

In many rework cycles, more states – described by stocks – are considered than the three basic ones illustrated in Figure 1. Along with additional stocks, new flows are also included, which can be useful in modeling different rates of process steps.

An often used intermediate stock is “Work in Progress”. Figure 1 shows in the top left corner (A1) a rework cycle which includes this stock. As a consequence, the rework rate is different from the original completion rate. In some processes this is a benefit, since the assumption of a constant rate for both original and rework would be an improper simplification.

Table 2 lists the references which include intermediate states, similar to the one under A1) in Figure 1. Naming is different in most of the cases. Therefore Table 2 includes a column that lists the names of the additional stocks.

Table 2. Intermediate states considered in rework cycles.

Intermediate states	References
Tasks Completed not Checked, Tasks Approved	[2]
Tasks Pending Test	[22]
Work in Progress	[5]
Tasks to Be Reworked	[7; 8]
Tasks in Testing	[20]
Work in Quality Assurance	[28]
Quality Assurance Backlog	[21]
Known Rework	[12; 24; 29; 30]

The modeler of an SD model is admonished to rethink the structure of the observed EDP and which states need to be captured in the rework cycle. Each crucial state has to be modeled by one stock and distinct process steps by distinct flow rates. The inclusion of the corresponding amount of stocks and flows allows for a more precise modeling of the process. This is especially useful for cases in which the rates of the process steps are significantly different and the use of an average rate for the entire

workflow would be insufficient. Examples for such different rates are resource requiring and non-resource requiring flows [5].

Another advantage of distinct stocks becomes apparent with regard to evaluation purposes. Distinct stocks allow for a separate monitoring of the progress for different process parts. Thus the causes for delays of the project end date can be better located.

These reasons advocate the use of intermediate states in SD models.

1.2. Consideration of important additional co-processes A2)

Some researchers use co-flow structures to consider the dynamics of auxiliary side processes like hiring, training or testing. The inclusion of these co-flow structures allows for a more precise modeling of the observed EDP because the interactions of the side processes and the rework cycles are captured in such models.

As shown in Figure 2, the side processes are modeled with co-flows that influence the rework cycle. In Figure 2, these side processes are staffing and testing. The available staff at a certain point can be calculated as the accumulated difference of the "Hiring" and "Turnover" rate.

In this example the staff level influences the "Work Accomplishment" rate, such that when more employees are available to do work, the rate increases. This influence is graphically represented with an information flow from the stock to the valve.

In the other co-flow of Figure 2 tests are processed from the "Test to Do" stock over a "Testing Rate" to the "Test Done" stock. The number of tests that are done impact the "Rework Discovery" rate.

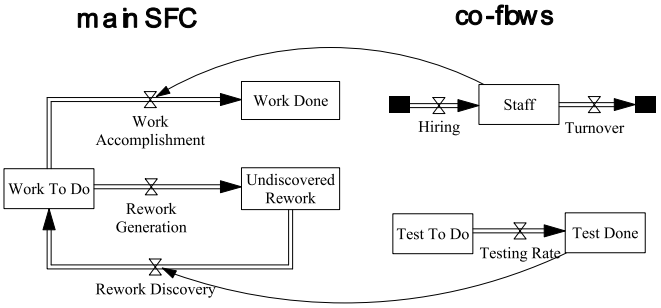

Figure 2. SD models of a rework cycle and co-flows for staffing and testing.

The modeling of the co-processes needs to be included in the SD model if both the main process is affected and the co-processes show a dynamic behavior during the execution of the main process. When comparing the behaviors of SD models with and without the consideration of these side processes, one would observe that the simulated project durations are drifting the further apart the more influence these side processes have on the rates of the main SFC.

For example in projects in which only few new employees are hired to contribute to the work accomplishment, the decrease of the simulated project duration compared to the basic model's duration is lower than the duration decrease of projects with many additional employees. Hence, the benefit of modeling the co-flow correlates with the impact of the hiring side process.

Some developers also integrate co-flows in order to gain additional information during the simulation. In those cases the co-flow does not necessarily affect the rework

cycle, such as the change co-flow in [10]. This co-flow is included to calculate the resulting costs, but that does not impact the rework cycle.

Table 3. Co-processes considered in rework cycles as parallel co-flows.

Co-processes	References
Testing	[8; 22]
Error rectification	[7]
Phase task change	[16]
Hiring	[12-15; 17; 18]
Training	[13; 15; 17; 18]
Change generation	[6; 10]
Change discovery	[6]
Completed work	[17]
Expended effort	[17]
Error generation	[13]

Table 3 lists various co-processes that are included in existing SD models allocated to their authors. This list serves as a guide for future modelers as to which side processes may be included in the model. Depending on the side processes of the observed EDP and its influence on the rework cycle the modeler needs to include co-flows in the SD model.

1.3. Consideration of influence of process concurrency in multi-phase projects A3)

Another important SFC adaptation of the basic rework cycle is consideration of the effects of process concurrency. This feature often appears in models of multi-phase projects. The impact of iteration due to releasing flawed work to subsequent phases is considered in these models. This relationship can also be called coordination.

In the models of [2; 16] coordination is represented by including a "Coordination" stock in the rework cycle. Other than the additional stocks for intermediate states, introduced above, the "coordination" stock is not part of the process chain but generates an additional iteration loop. The additional coordination stock enables flawed tasks from other phases to be accumulated in this stock until they can be coordinated. In the real process, this could happen, for example, in meetings with the responsible employees of the involved phases [2].

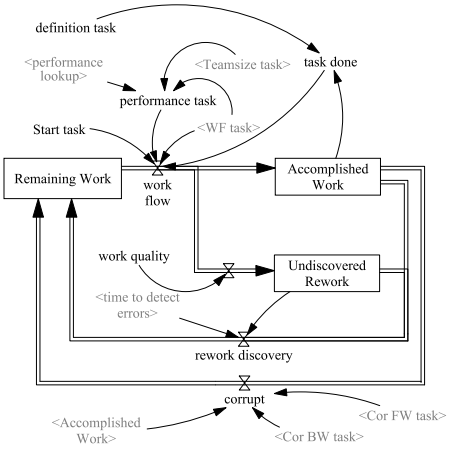

Figure 3. Rework cycle with corruption flow (adapted based on [4]).

The authors [4; 6; 8; 9] dispense with the coordination stock and solely use a corruption flow to model the influence of process concurrency. Figure 3 shows the rework cycle of [4] which iterates flawed work from the "Accomplished Work" stock to the "Remaining Work" stock via a corruption flow. The rate of the flow is calculated with the variables "Cor FW task" and "Cor BW task" which quantify the rework from other process steps that is either flowing to the particular subsequent (forward corruption) or previous (backward corruption) process step.

The inclusion of a coordination or corrupt flow is necessary when modeling multi-phase processes in which the phases are worked on concurrently. In case of such phase overlap the next phase starts before the previous phase has been completed. Therefore flawed tasks of the previous phase may not have been discovered yet and, though, are

released to the next phase. When eventually the flaw is discovered, the work unit needs to be sent back to the phase in which the flaw has been generated.

For this reason the basic rework cycle with its one rework discovery flow is not sufficient because it only accounts for phase internal flaws. It necessitates a second iteration loop to consider the rework of tasks that have been discovered in a different phase.

Table 4. SFC implementation of coordination in multi-phase projects.

SFC adaptations	References
Coordination stock & flow	[2; 16]
Corruption flow	[4; 6; 8; 9]

An overview of existing SFC adaptations for modeling the influence of phase concurrency is provided in Table 4.

1.4. Variable work accomplishment rate B1)

The work accomplishment during EDPs can be influenced by various factors. In SD models, these influences are modeled by causal links which affect the work accomplishment rate. For this reason, the adaptation scheme allocates the modeling of variable work accomplishment to the causal link adaptations. The influencing variables are listed and categorized in Table 5.

The company characteristic *quality* is a constant factor that defines the percentage of flawless work and thus the portion of work that moves from the "Work to Do" stock to the "Work Accomplished" stock.

The management lever *time to completion* is a factor that arises from the project schedule. The allocated *resources* directly influence the work accomplishment, whereas *pressure, overtime* and *organizational changes* define productivity and therefore indirectly affect the accomplishment of work.

Similarly, the human factors influence the productivity. Different *skill levels* and *morale* are characteristics of individuals that impact work completion. Furthermore, the performance of a team varies with the time due to the development of *synergies* and with *group size*.

Table 5. Influences on work accomplishment and associated references.

Category	Factor	References
Company characteristic	*Quality*	[2; 4; 8; 12; 17; 29; 30]
Management levers	*Pressure*	[5; 12; 30]
	Resources	[4; 5; 7; 19; 21; 22; 28-30]
	Overtime	[5; 12; 30]
	Organizational changes	[12]
	Time to completion	[17; 30]
Human factors	*Skill level*	[15; 17; 30]
	Morale	[12; 30]
	Synergy	[12]
	Group size	[4; 12]
Process factors	*Available work*	[2; 4; 5; 7; 8; 10; 13; 28; 30]
	Completed work in previous phase	[8]
	Undiscovered rework in previous phase	[10]

Within the category process factors, start conditions and iteration effects are differentiated. Start conditions represent the prerequisites for a phase or task to start. For example, work completion cannot start unless there is *work available*. Other factors, such as the amount of *undiscovered rework in a previous phase* continuously affect the work accomplishment throughout the whole process.

The necessity of each factor listed in Table 5 in an SD model is different. The factor *quality* has to be included in each model because it defines the share of flawed work units which create the rework cycle in the first place.

Other factors listed in Table 5 are optional and the benefit of their consideration in causal links of SD models depends on their influence on the rates. This influence can be significant. As stated earlier, *pressure, overtime* and *organizational changes* as well as the human factors influence productivity. This resulting productivity is found to be able to vary by a factor of two [12]. This productivity factor multiplied with the work accomplishment rate can change the process duration significantly. In these cases the adapted work accomplishment rate is better suited than a constant rate like in the basic model.

Values for the quantification of the listed factors can be found in the referenced literature in Table 5.

1.5. Variable rework generation rate B2)

Rework generation within an EDP in most cases cannot be described with a constant rate. Therefore many SD model developers include additional variables to demonstrate the varying behavior of rework generation. Table 6 brings together various factors on the generation of rework.

Table 6 shows that rework generation is dependent on the *work completion.* The reason for this relationship is that the rework generation rate is often calculated as the product of the work completion or accomplishment rate and 100% minus the *quality*. Another company characteristic is *target design maturity*. This variable is used in [5] in order to compare it with the project factor *believed design maturity* in order to trigger iteration in the rework cycle as well as the start of a subsequent phase.

Table 6. Influences on rework generation and associated references.

Category	Factor	References
Company characteristic	*Quality*	[2; 4; 5; 7; 8; 10; 12; 17; 22; 29; 30]
	Work completion	[4; 7; 8; 10; 12; 17; 22; 29; 30]
	Target design maturity	[5]
Project factor	*Believed design maturity*	[5]
Management lever	*Time to completion*	[17]
Iteration effect	*Undiscovered rework in previous phase*	[10]
Other	*Obsolescence*	[12]

The value of the variable *time to completion* terminates the process in the example of [17], and thus the rework generation corresponding to the work accomplishment ends. The model of [10] captures the fact that the amount of *undiscovered rework in the previous phase* increases the error generation in the next phase.

Another factor that can initiate rework is *obsolescence*, which means that already accomplished work becomes obsolete and thus worthless [12].

As explained in section 0 the benefit of adding these factors to the causal links influencing the rework generation rate is case-dependent. One example that proves the importance of an adapted rework generation rate through causal links can be found in the study of [10]. Through optimization they found that *undiscovered rework in the previous* phase can increase the rework generation in the following phase by up to 50%. With such a high dependency, a neglect of the factor like in the basic model would cause ineligible shortened project durations in simulations compared to the adapted model.

1.6. Variable rework discorvery rate B3)

The discovery of errors in tasks that have been worked on is usually modeled as being variable. Many authors include dependencies on other variables in their model to achieve a better approximation of the process they want to observe. Factors that can be found in the literature are listed in Table 7 and allocated to different categories.

Table 7. Influences on rework discovery and associated references.

Category	Factor	References
Company characteristics	*Probability of discovering errors*	[2; 8; 22]
	Undiscovered rework	[4; 8; 10]
	Quality	[2; 8; 22]
	Quality assurance / testing	[2; 8; 9; 21; 22]
Project factors	*Problem complexity*	[5]
	Perceived progress	[12; 17]
Management levers	*Pressure*	[21]
	Resources	[28]
Iteration effects	*Dependence on previous phase*	[2; 4; 7; 8]
	Rework completion	[8]
	Quality assurance / testing	[2; 8]
	Progress	[10]
Other	*Time*	[4; 5; 10]

Company characteristics are, for example, the *probability of discovering errors* and the amount of *undiscovered rework*. The higher these variables are, the more rework is discovered within a certain time unit. The higher the *probability of discovering errors* is, the longer the project duration but also the better the quality of the completed work. The company might also conduct test or *quality assurance* processes which also influence error finding.

The management lever *pressure* is modeled so as to increase the fraction of rework and thus the discovery rate in the rework cycle of [21].

The iteration effects can originate from elements in previous phases or of subsequent phases. A multiplicative relation between the rework rate of the previous phase and the corruption rate which is combined with the rework discovery rate is described in [4; 8]. Impacts from subsequent phases can be defined, for example, by the *quality assurance* of subsequent phases [8] or the work *progress* in these phases [10]. In the example of [10] the *progress* of the subsequent phase reduces the time constant for rework discovery in the previous phase with up to 50%. For such remarkable dependencies the simulated project duration is shorter for adapted rework cycles than for the basic model that does not account for the time constant reduction for rework discovery.

2. Reflection and discussion

The development of a guideline for the adaptation of rework cycles in SD provides future SD modelers with the necessary information and consolidates existing literature. The guideline details the major components of SD models and compares various rework cycles. Suggestions are made on when the adaptation of the rework cycle is more useful than the basic model.

One point that needs to be addressed is the initial evaluation of the guideline. A validation will require modelers to apply the guideline, preferably on a practical process. Moreover, the process which should be simulated by rework cycles would

have to be complex enough so that each feature presented in the guideline would be applied.

Therefore, one proposal for subsequent research is the success evaluation of the guideline. This requires practical EDPs as modeling subjects as well as resources such as SD modelers who apply the guideline during the modeling and give feedback.

3. Conclusion

The aim of this study is to create a guideline for the modeling of rework cycles in SD, adapted to simulate certain features of specific EDPs. The need for this guideline was derived from the desired support for SD modelers. Another driver for this research effort was the lack of a comparable guideline in the reviewed literature.

During the literature research, 25 rework cycles of various authors were observed regarding their purpose and structure. The rework cycles were analyzed in regards to how they simulate certain characteristics of the EDP and to how they differ from the simplest version of a rework cycle. Adaptations are applied to adjust the model to simulate the considered EDP appropriately.

Two kinds of adaptations were identified: SFC adaptations and causal link adaptations. SFC adaptations consist of the SD elements stock and flow and are mainly included in order to model intermediate states, parallel co-processes and iteration loops to realize coordination within the rework cycle.

Causal link adaptations on the other hand, consist of information flows and variables and capture the impact of stocks or variables on rates. The three basic rates work accomplishment, rework generation and rework discovery were observed concerning their influences. For each rate, influencing factors modeled in existing rework cycles were gathered and listed in the corresponding tables.

In a next step, the guideline was discussed, and further research needed for the evaluation was identified.

Acknowledgement

We thank the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) for funding this project as part of the collaborative research centre ‘Sonderforschungsbereich 768 – Managing cycles in innovation processes – Integrated development of product-service systems based on technical products’.

References

[1] M. Kreimeyer and U. Lindemann, Complexity Metrics in Engineering Design: Managing the Structure of Design Processes, Springer, Berlin, 2011.

[2] D.N. Ford and J.D. Sterman, Dynamic modeling of product development processes, *System Dynamics Review*, Vol. 14, 1998, No. 1, pp. 31-68.

[3] D.N. Ford and J.D. Sterman, Overcoming the 90% syndrome: Iteration management in concurrent development projects, *Concurrent Engineering,*Vol. 11, 2003, No. 3, pp. 177-186.

[4] D. Kasperek, M. Lindinger, S. Maisenbacher, and M. Maurer, A structure-based System Dynamics Approach for Assessing Engineering Design Processes, in: *2014 International System Dynamics Conference, System Dynamics Society*, Delft, Netherlands, 2014.

[5] H.N. Le, *A Transformation-Based Model Integration Framework to Support Iteration Management in Engineering Design*, PhD Thesis, University of Cambridge, Cambridge, UK 2012.
[6] S. Lee and I.S. Lim, Degree of Overlapping Design Activities in Vehicle Development : A System Dynamics Approach*, Asian Journal on Quality,* Vol. 8, 2007, No. 2, pp. 128-144.
[7] J. Lin, Overlapping in Distributed Product Development, in: *2006 International System Dynamics Conference*, System Dynamics Society, Nijmegen, The Netherlands, 2006.
[8] J. Lin, K.H. Chai, Y.S. Wong, and A.C. Brombacher, A dynamic model for managing overlapped iterative product development, *European Journal of Operational Research,* Vol. 185, 2008, No. 1, pp. 378-392.
[9] F. Nasirzadeh, M. Khanzadi, A. Afshar, and S. Howick, Modeling quality management in construction projects, *International Journal of Civil Engineering*, Vol. 11, 2013, No. 1, pp. 14-22.
[10] K. Parvan, H. Rahmandad, and A. Haghani, Empirical Study of Design-Construction Feedbacks in Building Construction Projects, in: *2013 International System Dynamics Conference*, System Dynamics Society, Cambridge, MA, 2013.
[11] A. Powell, K. Mander, and D. Brown, Strategies for lifecycle concurrency and iteration–A system dynamics approach, *Journal of Systems and Software,* Vol. 46, 1999, No. 2, pp. 151-161.
[12] K. Reichelt and J. Lyneis, The dynamics of project performance: benchmarking the drivers of cost and schedule overrun, *European Management Journal,* Vol. 17, 1999, No. 2, pp. 135-150.
[13] S. Ruutu, P. Ylén, and M. Laine, Simulation of a distributed design project, in: *2011 International System Dynamics Conference*, System Dynamics Society, Washington, DC, 2011.
[14] T. Haslett and S. Sankaran, Applying Multi-Methodological System Theory to Project Management, in: *53rd Annual Meeting of the ISSS*, Brisbane, Australia, 2009.
[15] J.I.M. Hernández, J.R.O. Olaso, and A.G. López, Technology Assessment in Software Development Projects Using a System Dynamics Approach: A Case of Application Frameworks, in F.P.G. Márquez and B. Lev (eds): *Engineering Management*, InTech, Rijeka, 2013, pp. 119-142.
[16] T. Laverghetta and A. Brown, Dynamics of naval ship design: a systems approach, *Naval Engineers Journal*, Vol. 111, 1999, No. 3, pp. 307-323.
[17] S. Lisse, Applying system dynamics for outsourcing services in design-build projects, *Journal of Project, Program & Portfolio Management,* Vol. 4, 2013, No. 2, pp. 20-36.
[18] J. Williford and A. Chang, Modeling the FedEx IT division: a system dynamics approach to strategic IT planning, *Journal of Systems and Software*, Vol. 46, 1999, No. 2, pp. 203-211.
[19] L.J. Black and N.P. Repenning, Why firefighting is never enough: preserving high‐quality product development, *System Dynamics Review*, Vol. 17, 2001, No. 1, pp. 33-62.
[20] N.P. Repenning, A dynamic model of resource allocation in multi‐project research and development systems, *System Dynamics Review*, Vol. 16, 2000, No. 3, pp. 173-212.
[21] T. Taylor and D.N. Ford, Tipping point failure and robustness in single development projects, *System Dynamics Review*, Vol. 22, 2006, No. 1, pp. 51-71.
[22] H. Rahmandad and K. Hu, Modeling the rework cycle: capturing multiple defects per task, *System Dynamics Review*, Vol. 26, 2010, No. 4, pp. 291-315.
[23] H. Rahmandad, Dynamics of platform-based product development, in: *2005 International System Dynamics Conference, S.D. Society*, System Dynamics Society, Boston, MA, 2005.
[24] K. Cooper and G. Lee, Managing the dynamics of projects and changes at Fluor, in: *2009 International System Dynamics Conference*, System Dynamics Society, Albuquerque, NM, 2009.
[25] J.M. Lyneis, K.G. Cooper, and S.A. Els, Strategic management of complex projects: a case study using system dynamics, *System Dynamics Review*, Vol. 17, 2001, No. 3, pp. 237-260.
[26] G. D'Avino, P. Dondo, and V. Zezza, Reducing ambiguity and uncertainty during new product development: a system dynamics based approach, in: *Technology Management: A Unifying Discipline for Melting the Boundaries*, IEEE, Portland, OR, 2005.
[27] N.P. Repenning and J.D. Sterman, Capability traps and self-confirming attribution errors in the dynamics of process improvement, *Administrative Science Quarterly,* Vol. 47, 2002, No. 2, pp. 265-295.
[28] N.R. Joglekar and D.N. Ford, Product development resource allocation with foresight, *European Journal of Operational Research*, Vol. 160, 2005, No. 1, pp. 72-87.
[29] K.G. Cooper, Naval ship production: A claim settled and a framework built, *Interfaces*, Vol. 10, 1980, No. 6, pp. 20-36.
[30] K.G. Cooper, J.M. Lyneis, and B.J. Bryant, Learning to learn, from past to future, *International Journal of Project Management*, Vol. 20, 2002, No. 3, pp. 213-219.