A Co-evolution Oriented Change Analysis Framework in Product Development Project

Xin ZHANG * **Marc ZOLGHADRI** ** **Patrice LECLAIRE** **
Philippe GIRARD *

* *CNRS-IMS UMR-5218*
University of Bordeaux
351 Cours de la Libération, 33405 Talence, France
(e-mail: firstname.family@ims-bordeaux.fr).
** *LISMMA EA2336*
Supméca-Institut supérieur de mécanique de Paris
3 Rue Fernand Hainaut, 93400 Saint-Ouen, France
(e-mail: firstname.familyname@supmeca.fr)

Abstract: Nowadays, companies are facing more and more challenges during Product Development (PD) project. On one hand, the PD project should design and develop the product to satisfy the requirements as far as possible under time, cost and quality constraints. On the other hand, the PD project is expected to deal with any possible modification in an agile and flexible way. Therefore, how to manage changes under various restrictions becomes an essential issue during a PD project. The aim of this paper is to provide some exploratory results concerning modeling change propagations during PD projects to support companies to improve their change management performance. We first propose a Co-evolution Oriented Change Analysis (COCA) framework. The framework enables us to simultaneously model product management, project management, and partnership management knowledge areas as well as the interrelations between them. In the framework, we propose a product evolution model manifesting itself with a serial of states reflecting how the information/data is aggregated along the project process. Relying on the product evolution model as well as the aggregation of information/data from the multiple knowledge areas, we suggest a method of building up a change propagation network. Within the network, we identify and summarize a set of change propagation patterns. Through analyzing the characteristics of the patterns, we suggest some guidelines as a theoretical contribution to change management.

1. INTRODUCTION

The growing complexity of business context brings more challenges to Product Development (PD) project. Generally, a PD project is defined as *an endeavour of activities beginning with the perception of a market opportunity and ending in the production, sale, and delivery of a product* [Ulrich and Eppinger, 2007]. On one hand, the PD project is expected to deliver the product with less time cost to satisfy the requirements as far as possible, in order to reduce time-to-market of the final product. On the other hand, the PD project has to deal with potential changes to retain the flexibility and agility in aim of keeping competitiveness of the products in the market.

During a PD project, changes reveal multiple senses. Changes could bring the opportunity to a company for developing innovations. However, changes could also enhance the risk of failing to release the final product within the constraints (lead time, requirements, cost, etc.) and lead in some unexpected or unforeseen situations. Along the PD project, changes may occur at any time. Given the occurrence of a change, the later it is coped with, the more the cost would be paid according to the Rule of Ten proposed by Clark and Fujimoto [1991]. Considering this, the concerned system, product parts, features or activities during product development are proposed to be frozen prior to a serial of predetermined freeze points to reduce the risk of rework [Eger et al., 2005].

Also, one change can cause the occurrence of other changes. In other words, change can propagate through the potential relations between the systems, product parts, features and/or activities involved in the product development. Eckert et al. [2004] pointed out that the link between the composition of a system (such as a complex product) can reflect the relations of changes. Considering the complexity of PD project and the multiple involved aspects, the changes occurring on the involved elements and their inter-relations compose a network where the nodes represent the elements and their relations are modelled by the edges. We specify this network as change propagation network and the edges as potential change propagation channels, i.e., the relation between the two involved elements that could transfer the impact of the initial change occurring in the upstream element to the downstream element. Therefore when a change occurs on a

given element, it could be coped with locally or transferred to other element(s) or both.

Change propagation would involve multiple knowledge areas of PD project, i.e., the product and constituent components, project activities, and involved partners. In other words, a change occurring in one system belonging to one knowledge area would cause other change(s) in other area(s). For instance, a modification to the design parameter of one product component would force the concerned supplier to re-produce the components, and then the project would be delayed because of the extra time cost of re-production. The impact of the original change could either be amplified or reduced through the propagation. Thus, changes are expected to be perceived, analyzed and coped with or even controlled as soon as possible.

In this paper, we firstly propose Co-evolution Oriented Change Analysis (COCA) framework, with which we are enabled to simultaneously model (1) the product management, (2) project management, and (3) partnership management knowledge areas of PD project as well as (4) their interrelations. In the COCA framework, we introduce a product evolution model facilitating to recognize how the information/data derived from the multiple knowledge areas is aggregated. Relying on the product evolution model, we also suggest a method of building up a network in which change propagations are investigated. In further, a set of change propagation patterns are identified and summarized, and they provide a theoretical contribution to change management in identifying the mechanism of change propagation and eliciting some strategies or methods to cope with.

Hereafter in section 2, the related literature is reviewed and summarized. Then in section 3 we propose the COCA framework that models product design and development process with a serial of product states and suggests a simultaneous consideration in the multiple knowledge areas. In section 4, we propose a set of change propagation patterns relying on the COCA framework. The conclusion and perspective is located in the last section.

2. RELATED WORK

This section identifies and summarizes available literature of change management in PD projects. The literature review mainly concerns two parts. The first part examines the research of identifying and analyzing change propagations. The second part introduces the relevant research supporting our work during modeling the change propagations.

2.1 Conception of Change and Change Propagation

In general understanding, we describe change as *an act or process through which something becomes different*. In practice, the phenomena of change are correspondingly embodied according to the specific field of perceiving them. Although most of the reviewed literature mentioned the concept of changes, they only regard the concept as brief related work introduction or state of art to clarify the scenario for their work [Arnaud et al., 2002, Gareis, 2010, Bröchner and Badenfelt, 2011]. While other researches provide some contributions to the conception of changes. Eckert et al. [2005] indicated design change can be interpreted as the alterations to design process for modifying an existing design or recognizing shortcomings in a partially completed design. As lying the heart of almost all the design processes, change is identified as the improvement or even the innovation of design processes.

Based on the understanding of change, change propagation is regarded as a phenomenon that a change causes other one(s). The fact that a change rarely occurs alone and multiple changes can have interacting effects with each other [Eckert et al., 2004] is validated frequently in the real business context, which presents that the network of changes with various connections is more practical and veracious representation of change propagation. According to [Eckert et al., 2004], individual change chains are not sufficient to illustrate the practical multiple changes having interacting effects on other (sub-)systems, therefore the concept of complex change networks was introduced to display the connections between (sub-)systems or parts. For example, the connections between the collaborating departments within a company as the channels enable engineering changes to propagate [Do et al., 2008]; the dependencies between the design parts of a final product can transfer the consequence of changes [Ouertani and Gzara, 2008].

2.2 Change Propagation Analysis

Rouibah and Caskey [2003] proposed a parameter-based methodology to analyze change propagation in a collaborative scenario. According to the product structure, the parameters linked to elements of product as well as to persons can compose a network, which can also inform the relevant persons (or roles) whenever the status of the value of a particular parameter changes. In this way, the possible change propagation between the changed and the neighboring parameters in the parameter network can be discovered and notified to the co-ordinators and collaborators. Depending on the parameter network, the change propagation might be tracked more or less easily to its conclusion. Within the change networks studied by Eckert et al. [2004], four types of change propagation behaviors are summarized through their investigation, i.e., constants, absorbers, carriers and multipliers.

In the research proposed by Rutka et al. [2006], a method for change propagation analysis based on a dependency model considering three main aspects was suggested: (1) information that represents several viewpoints or domains of the engineering system, (2) dependency information that describes the links or relationships between two items, (3) the evolvement of overall design representation and the corresponding engineering organization along the product lifecycle.

Inspired by the reviewed literature, the interactions between the systems provide the channels through which changes can propagate. The involved systems as well as their interactions implies a potential network where we can analyze changes and change propagations. Based on this idea, , we propose the modeling approach that enables us to build up the network of change propagations and to identify the change patterns.

3. THE COCA FRAMEWORK

In this paper, the framework named Co-evolution Oriented Change Analysis (COCA) is proposed to analyze changes and change propagations by considering the three knowledge areas (i.e., product management, project management, and partnership management) simultaneously (see Figure 1). It enables us to:

- Model product design and development process by indicating how product evolves considering the multiple knowledge areas simultaneously;
- Acquire the information/data derived from the multiple knowledge areas. The information/data can be aggregated to reflect the product functions.
- Identify the dependencies between the information/ data belonging to the same/different knowledge area(s);
- Identify the potential change propagation channels, and analyze the mechanism of change propagations.

Within the COCA framework, the first knowledge area we identify is *project management*. It involves arranging the product design and development activities with a set of milestones and restricting the activities carried out under the constraints from time, quality and cost. We investigate the generic product design and development process (also called the generic process for short in this paper) proposed by Ulrich and Eppinger [2007] as the starting point to model a PD project. The six phases of the generic product design and development process imply a serial of critical activities (i.e., (1) planning, (2) concept development, (3) system-level design, (4) detail design, (5) testing and refinement, and (6) production ramp-up) that are executed in turn to achieve the prescribed milestones. This generic process reflects our consideration in the project management issue. The second knowledge area is *product management*. It mainly highlights the evolution of product from customer needs to design solution. We propose a product evolution model which reflects the evolving states of product during the PD project. Each state captures the highlighted form of the product at the particular moment. The product evolution model aligns to the first five phases of the generic process. The third knowledge area is *partnership management*. It enables us to consider the partners who participate in the PD project in terms of their involved activities and their roles. We categorize the partners into two types, i.e., *end product suppliers* and *enabling product suppliers*. The end product suppliers are those supplying the *end product(s)* to the company, and the enabling product suppliers are those simply providing the *enabling product(s)* to the focal company. The end products and the enabling products originate from the EIA-632 standard [Electronic-Industries-Alliance, 1999]. The end product refers to *the portion of a system (i.e., final product) that performs the operational functions and is delivered to the company*. The enabling product is *the item that provides the means of enabling the end product(s) to get into service, keep in service or terminate from service.*

3.1 Product Evolution Model

Along the generic process, the product evolves from the statement of initial needs to the design solution for the further manufacturing process. We identify four critical states reflecting the phased forms of the evolving product. The four states refer to *Needs*, *Requirements*, *Logical solution* and *Physical solution* (see Figure 1). During the phases of achieving the states, four *deliverables* are composed and maintained (see the bottom part of Figure 1). The deliverables are *Needs Definition*, *Requirement Definition*, *Logical Solution* and *Physical Solution*. Each deliverable is a set of generated documents and indicates:

(1) the results obtained through executing the activities along the product evolution (illustrated by the alignment arrows in Figure 1), i.e., what the product currently consists of and how the product is made up given the information/data obtained from the multiple knowledge areas;
(2) the specified objectives of designing and developing the product, and the means used to determine whether the objectives are achieved.

As the alignment to the generic process indicating, the "***needs***" state is forming from the "planning" phase and is reached during the "concept development" phase. During achieving this state, the expectations, needs from acquirer and stakeholders are transformed to specifications. These information as well as some other derived technical requirements are contained in the deliverable "needs definition". Certain outputs obtained through the two phases (i.e., planning, concept development phases) like the project mission statement, the market opportunity are also integrated into the deliverable.

The "***requirements***" state starts to form during the "concept development" phase and is reached during the "system-level design" phase. Within this period, the feasibility of the potential product concepts are investigated and the industrial design concepts are developed. The outcome of achieving this state is "requirements definition" deliverable that contains some system technical requirements, and the design constraints concerning the performance and function of design solution. The constraints concerning production and the supply chain strategy derived from the two phases (i.e., concept development, system-level design phases) as well as the requirements of enabling products are also included in the deliverable.

The "***logical solution***" state begins to form during the "system-level design" phase, and is reached by the end of this phase. During forming this state, the capability, behaviour, and structure of the product are defined. The product architectures, some derived technical requirements and a list of key suppliers are generated and recorded in the "logical solution" deliverable.

The "***physical solution***" state is forming when the "detail design" phase begins and is achieved by the end of the "testing and refinement" phase. Through achieving this state, the component geometry is defined, and the tolerances of all the concerned parameters are identified and assigned. Also the plans of validating and verifying the design solution are composed. The make-buy analysis is executed according to the determined supply chain strategy. As the outcome, the "physical solution" deliverable records the geometry specification of the subsystems and the design parameters. The design solution of the end

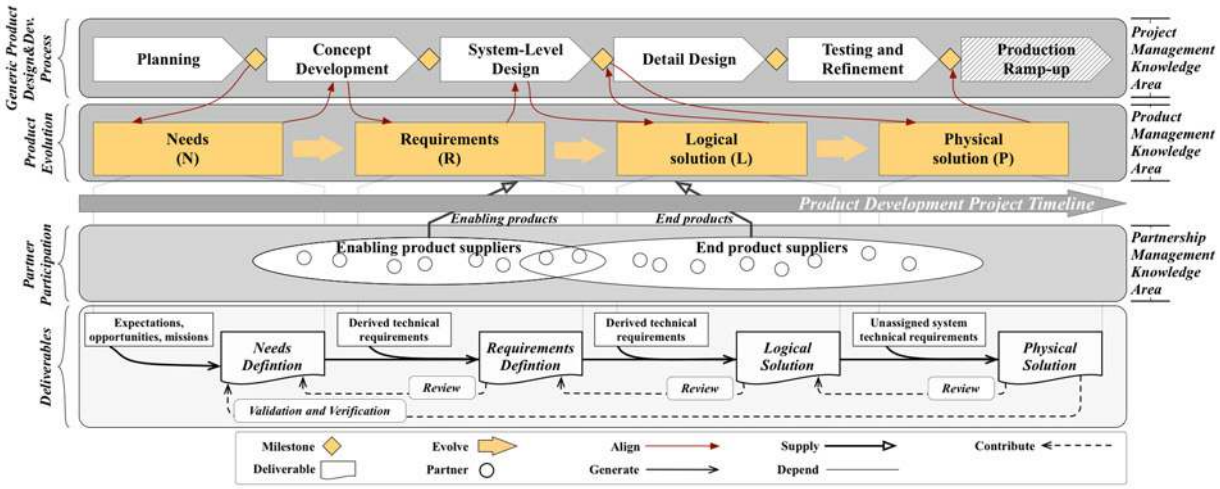

Fig. 1. Co-evolution Oriented Change Analysis (COCA) framework

product(s) as well as the alternative ones are also included in this deliverable.

During the product evolution process, we think of the product in both functional and physical terms. The *functional systems* of the product are "the individual operations and transformations that contribute to the overall performance of the product", and the *physical systems* are "the parts, components, and subassemblies that ultimately implement the product's functions" [Ulrich and Eppinger, 2007]. During the product design, the functional systems describes the effects exposed by the physical systems [Wie et al., 2005].

The overall functional system is identified by analyzing the customer expectations/market opportunity/project mission. Through the product evolution process, the overall functional system is more and more specified, branching into the functional *building blocks* of the smaller granularity. In this paper, the building blocks refer to the intermediate systems obtained through decomposing the system, and they can be decomposed in further till into the end elements (i.e., the ones of the minimum granularity can not be decomposed any further). Along this decomposition, the physical systems evolve into the smaller physical building blocks as the implementations corresponding to the functional systems/building blocks.

Based on the above product evolution model, we take use of product evolution process as the main clue to aggregate the information/data from the other two knowledge areas and then present our simultaneous consideration. We suggest a paradigm to compose the potential aggregations:

To achieve one of the PD project goals in which a *product function* (product) is identified and then implemented by one *module/component*, one or more *activities* are executed under the *constraints* (project), and one or more *partners* would participate in some activities to perform their *responsibilities*(partner).

By the above paradigm, we specify the interrelations between the three knowledge areas and then are enable to model the project process with the information/data of the knowledge areas simultaneously.

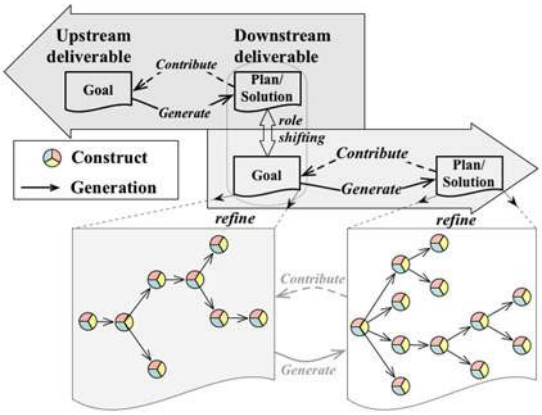

Fig. 2. Mutual relations between deliverables and constructs

In this section, we introduced the product evolution model aligned to the generic process, along which the suppliers contribute their effort with the end products and/or the enabling products. Taking the product evolution process as the main clue, we depicted our simultaneous consideration in the multiple knowledge areas to model a PD project by the COCA framework. In the following section, we will demonstrate the procedure of acquiring the information/data from the three knowledge areas.

3.2 Mutual Relations between Deliverables and Constructs

Corresponding to the product evolution states, each two adjacent deliverables (denoted as upstream deliverable and downstream deliverable respectively) maintain the mutual relations as followings (also see Figure 2).

- Generation: Given the goal in the upstream deliverable, the corresponding plan(s)/solution(s) is/are created or produced by the downstream deliverable.
- Contribution: The plan/solution in the downstream deliverable contributes in achieving the corresponding goal created in the upstream deliverable through supplying the produced effort.

The mutual relations between the deliverables are not only implying the transformations between the states

but could be also embodied by the relationships between the functional and/or the physical systems (or building blocks) corresponding to the particular determined design methodology in the PD project. In this paper, we investigate the axiomatic design methodology [Suh, 1990] and the functional reasoning methodology [Umeda and Tomiyama, 1997] to educe and identify the potential generation relations between the functional and/or the physical systems (or building blocks). According to the two methodologies, the potential generation relations between the functional/physical systems (or building blocks) are identified and listed in Table 1.

Table 1. Generation relations between functional/physical systems

Design methodology	Involved systems
Axiomatic design	**one** functional system generates **one or more** physical system(s)
	one physical system generates **one or more** functional system(s)
Funcitonal reasoning	**one** functional system generates **multiple** sub-functional systems
	one functional system generates **one or more** physical system(s)
	multiple sub-physical systems generate **one** physical system

As indicated in Table 1, the generation relations between the functional and/or the physical systems (or building blocks) only reflect the process involving product management knowledge area. Then we extend to aggregate the information/data from the other knowledge areas with the functional/physical systems according to the paradigm through the simultaneous consideration (see section 3.1). In this way, corresponding to each functional/physical system, we create an entity named as *construct* that is specified by the information/data from the three knowledge areas. Each construct is perceived and determined by the physical/functional system (or building block). So the generation relations can also be identified between the constructs (see Figure 2).

The generation relation between the deliverables/constructs not only implies the processes of product evolution but also could transfer the impact of occurred changes as a *change propagation channel*. The change occurring in the upstream deliverable could cause another change in the adjacent downstream deliverable through their generation relation, and it is the same to the constructs. For example, the customer needs (described as "a bike particularly for indoor cycling") can be regarded as the goal specified in "need definition" deliverable. It would be answered by the design constraints in the further plan/solution, i.e., "requirements definition" deliverable, for example "a narrow size of bicycle tyre". If there comes a change that the customer turns to expect a mountain bike as final product, then the size of bicycle tyre could be changed accordingly.

An example Here we demonstrate a small example [1] to help the readers understand the above content, in

[1] The final product in the example originates from a "*Heckler All Mountain Bicycle of 2004 collection*" produced by *Santa Cruz Bicycles company* whose website is "http://www.santacruzbicycles.com/home/"

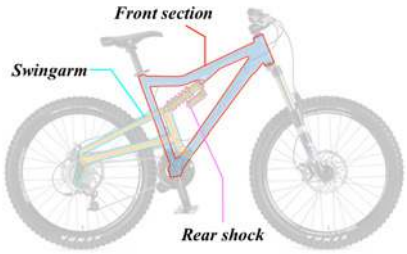

Fig. 3. Final product: rear suspension system

which the final product is "the rear suspension system in a mountain bike". As Figure 3 illustrated, the final product consists of three parts: (1) front section, (2) swingarm, and (3) rear shock. At the earlier phase of the project (i.e., the product evolves as "needs" state), we initially describe one of the product functions (denoted as "$f_{suspension}$") as "absorbing the vibrations from the ground". Along the product evolution process, the mentioned product function (i.e., $f_{suspension}$) is refined into sub-functions and those functions of smaller granularity can be identified at later phase as: (1) "connecting to the front section and the rear shock" (of the swingarm, denoted as "$f^1_{swingarm}$"), (2) "keeping stiffness" (of the swingarm, denoted as "$f^2_{swingarm}$"), (3) "connecting to the front section and swingarm and the front section" (of the rear shock, denoted as "$f^1_{rear\_shock}$"), (4) "reacting to the movement led by the swingarm" (of the rear shock, denoted as "$f^2_{rear\_shock}$"), (5) "connecting to the swingarm and the rear shock" (of the front section, denoted as "$f^1_{front\_section}$"), (6) "holding the rider off the ground" (of the front section, denoted as "$f^2_{front\_section}$").

Referring to the constructs, their information/data from the knowledge areas does not only imply the generation relations with others but also manifest itself as the interrelated measure items determining the response, the characteristics and/or the behaviour of the systems/building blocks issued by the three knowledge areas. These measure items are called as *direct parameters* [Andreasen, 1987].

Moreover, the constructs belonging to the different deliverables and the generation relations between them compose a change propagation network. In this paper, we name this network as *inter-deliverable change propagation network* with respect to the product evolution process. In the following section, we then investigate the dependencies between the direct parameters of the constructs, and the dependencies imply another change propagation channels.

3.3 Dependencies between Constructs

Along the product evolution process of a PD project, on one hand a deliverable can be refined with a set of constructs and each of the constructs can be particularly specified by the functional/physical systems/building blocks. On the other hand, the focal company executes the activities with the direct parameters contributed by the multiple knowledge areas (i.e., product management, project management, partnership management) to acquire the achievements (i.e., educing new direct parameters) of the PD project. To explain that, we still turn to the example mentioned in section 3.2.1. In the early phase,

$f_{suspension}$ is identified through analyzing the customer needs. Considering that the specifications to the needs is still abstract , we can only identify the direct parameters characterizing the systems of larger granularity, such that a parameter of "composition of the final product" is a set of "front section", "rear shock", and "swingarm" parts. Later on, more direct parameters will be educed along the product evolution process to characterize the systems obtained through decomposing the ones of large granularity till the systems that can not be decomposed any more. After identifying that the final product consists of the three parts, we can identify the direct parameters characterizing "front section", "rear shock", and "swingarm" parts from the document of specifications. For example, one direct parameter "length of rear shock" characterizes "rear shock", and it is concerned to express the function of $f^1_{rear\_shock}$. Meanwhile, another direct parameter "spring preload" of "rear shock" is also identified, and it is concerned to express another function of $f^2_{rear\_shock}$.

Referring to the direct parameters characterizing the building blocks (or systems) of the three knowledge areas, we discover there exist various *dependencies* between them. In our research, a dependency between two direct parameters is defined as the effect of the change in one direct parameter's value on another according to the definition proposed in [Andrew and Juite, 1995]. In Table 2, we present the six dependence cases of direct parameters with some representative research contributions.

Table 2. Classification of dependencies

Dependency	Methods and example references
Project∼ Project	Six modelling "primitives"[Kathleen and David, 1999]; Activity dependencies[Browning et al., 2006].
Product∼ Product	Dual-domain analyses of product issues [Mike and R, 2007]; Dependency at creation/modification, consistency, redundancy [Ouertani and Gzara, 2008].
Partner∼ Partner	Team-Based Design Structure Matrix [Tyson, 2001]; Partners dependency modeling [Zouggar et al., 2009].
Product∼ Project	Domain Mapping Matrix [Mike and R, 2007]; Generalized Bill-Of-Materials and Operations [Zolghadri et al., 2010].
Product∼ Partner	Dependencies between multiple domains [Danilovic and Börjesson, 2001]; Incidence matrix mapping activity to design parameters[Zouggar et al., 2009].
Project∼ Partner	Hidden dependency between partners [Ulrich and Eppinger, 2007]; Generalized Bill-Of-Materials and Operations[Zolghadri et al., 2010].

In our research, all the dependencies between constructs are categorized as mono-directional and bi-directional dependencies. The classification of dependencies enable us to analyze change propagation channels with considering the directions of change propagation. Referring to the mono-directional dependency between constructs, the change propagation can only occur from the upstream construct to the downstream one. While the bi-directional dependency allows the change propagation occurring in either construct and then influence the other one. Taking a flashlight as example, which is simply composed with shell as main body, battery for supplying power, a reflector for concreting light and a small bulb for lighting. Concerning the energy relation between the bulb and the battery, we can find there exists some mono-directional dependency between them, i.e., there is an effect of the change in batterys ampere parameter on the bulbs brightness.

Within one deliverable, the constructs refined from them and their dependencies compose a second change propagation network. We name this network as the *intra-deliverable change propagation network*.

In this section, we mentioned two classes of change propagation networks (inter-deliverable and intra-deliverable networks) corresponding to the relationships between the constructs (i.e., mutual relations, dependencies). In section 4, we will investigate the two networks and highlight some meaningful substructures of the networks to suggest a set of change propagation patterns.

4. CHANGE PROPAGATIONS PATTERNS

We characterize change propagation with three indicators under COCA framework. They are Trajectory, Involved deliverable(s) and Explored constructs.

Trajectory: It indicates the routing of change propagations. Based on the relations between the constructs and their belonging deliverable(s), we identify three directions of change propagation perceived from one construct: (1) forward direction which means due to generation relation, current change causes another change occurring in other construct of the downstream deliverable; (2) feedback direction which means due to contribution relation, current change causes another change occurring in other construct of the upstream deliverable; (3) inward direction which means due to dependency, current change causes another change in other construct of the same deliverable.

Involved deliverable(s): If the change propagation only concerns the constructs within the same deliverable, then there is at least one involved deliverable. Otherwise, more than one deliverable is involved in the given change propagation. The former condition is classified as intra-deliverable change propagation, while the latter is inter-deliverable change propagation.

Explored constructs: Along the change propagation, a number of constructs are explored due to the effect of change transferred in accordance with the trajectory. Among the constructs, a particular one where the change propagation starts is identified as initial construct and denoted as CT_i, i.e., the start point of change propagation. There also exist(s) some construct(s) where the effect of change would not be transferred further, i.e., the change propagation is terminated. These construct(s) is/are identified as final construct(s) and denoted as CT_f. The rest constructs along the change propagation are identified as intermediate constructs with the denoted CT_m.

The mutual relations enable us to track and trace constructs in accordance with PD project timeline, therefore the generation relations enable us to discover and predict some potential changes in later phase(s) caused by propagation. Meanwhile, the contribution relations enable us to investigate some initial change(s) occurring in earlier

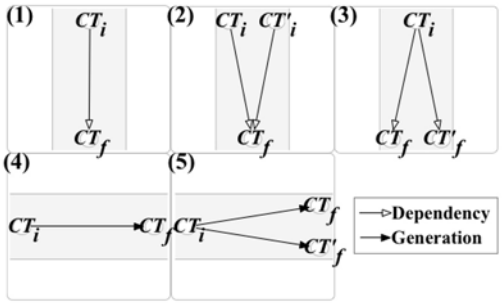

Fig. 4. Intra-deliverable change propagation (1, 2, 3) and inter-deliverable change propagation patterns (4, 5)

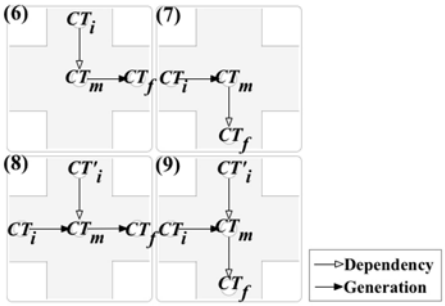

Fig. 5. Hybrid change propagation patterns (6, 7, 8, 9)

phase(s) that cause the current change by propagation. Through analyzing the dependencies between the constructs within the same construct, the couplings of parameters from different constructs can be identified. Given the change propagations through the dependencies as channels, the affected constructs are perceived and identified, with which the consequence of change propagation can be evaluated. Taking both the mutual relations and the dependencies into account, we combine them into more complex change propagation channels. Within the complex channels, not only the change propagations through either the mutual relations or the dependencies should be paid attention to, but the ones going through both the mutual relations and the dependencies should also be highlighted. The latter change propagations imply a set of particular constructs that require more cost of coping with.

We suggest a set of patterns by categorizing the change propagation channels, which cover intra-deliverable change propagation (i.e., dependencies as change propagation channels), inter-deliverable change propagation (i.e., mutual relations as change propagation channels), and hybrid channel change propagation (i.e., dependencies and mutual relations as change propagation channels). Figure 4 illustrates the first two types of change propagations by showing the initial constructs (denoted as CT_i), the final constructs (denoted as CT_f) and the dependencies/relations between them. Patterns (1) and (4) of Figure 4 illustrats the basic principle of change propagations through dependencies and relations respectively. Patterns (2), (3), (5) are the change propagation patterns composed with patterns (1) and (4).

In practice, change propagations often go through both dependencies and mutual relations. In Figure 5, we present the change propagation patterns illustrating the scenarios of hybrid channels. The change propagation patterns consist of initial constructs (denoted as CT_i), the intermediate constructs (denoted as CT_m), the final constructs (denoted as CT_f) and the dependencies/relations between them. Patterns (6) and (7) are the basic situations of hybrid channel change propagation patterns, while patterns shows the more complex situations. All the perceived change propagations under COCA framework can be treated and converted into a combination of the above nine patterns.

We propose a set of suggestions to adopt our methodology towards better change management. Our suggestions are based on the attributes proposed by Fricke et al. [2000] that indicate the tendency of better change management, which includes:

- Less: it aims to reduce and/or eliminate the avoidable changes especially in the later phase of project.
- Earlier: it derives from the consideration in Rule of Ten [Clark and Fujimoto, 1991] and aims to detect emerging changes earlier.
- Effective: it provides analysis of effective efforts and benefits a change could bring.
- Efficient: it states that the resources (such as time, cost) employed for coping with changes should be used efficiently.
- Better: it emphasizes the importance of improving the maturity of managing changes through learning from the past change process.

Corresponding to the above five attributes, the following suggestions are given:

(1) Range: change propagation should be controlled within a smaller range, i.e., the fewer deliverables and constructs involved, the better. Given a change propagation channel, the final and intermediate constructs in it should be equipped with the solution of coping with changes in order to eliminate the further propagated changes.
(2) Timing: along the project timeline, the constructs (i.e., initial construct and intermediate construct) in which changes occur in earlier deliverable should be prepared with some solution of coping with change to prevent the potential change propagation and limit the further cost.
(3) Critical construct: An intermediate construct that receives both inter-deliverable and intra-deliverable change propagation is considered as a critical construct (see CT_m in pattern 8 and 9 of Figure 5). Within the given change propagation, those critical constructs require more efforts in order to handle the difficulty and cost of coping with change.
(4) Multiple mapping: given the multiple mapping in change propagation, the construct which either receives or transform multiple dependencies/relations should be given with the priority of coping with changes. According to the patterns stating multiple mapping, the constructs connecting with others through multiple mapping channels should be given the priority of coping with changes.
(5) Change propagation pattern: through our methodology, the perceived change propagations can be treated and converted into the combination of the patterns.

5. CONCLUSION

In this paper, we introduced Co-evolution Oriented Change Analysis (COCA) framework to provide some exploratory results. First, the framework that reflects product evolution process and suggests a simultaneous consideration in the multiple knowledge areas (i.e., product, project and partnership areas). Second, we proposed the method of modeling the aggregation of information/data of PD project based on the framework. With the method, the change propagation networks are identified that support to analyze change propagations within/between the multiple knowledge areas. Third, we identified and summarized a set of change propagation patterns through investigating and analyzing the networks. Relying on the characteristics of the patterns, we suggested several guidelines towards better change management when employing the COCA framework.

Limited by space, there are some details in the framework not being mentioned, for example, how to represent the building blocks, the constructs. In addition, the current contribution provides a theoretical result without explanation of how to apply it into real case.

In further, we will keep our effort to analyze the mechanism of change propagations and develop the improvement guidelines.

REFERENCES

Hein L Andreasen, MM. IFS (Publications) Ltd, 1987.

Kusiak Andrew and Wang Juite. Dependency Analysis in Constraint Negotiation. *IEEE Transations of system, man, and cyberetics*, 25:1301–1313, September 1995.

Riviere Arnaud, DaCunha Catherine, and Tollenaere Michel. Performances in engineering changes management. In *IDMME 2002*, pages 1–10, October 2002.

Jan Bröchner and Ulrika Badenfelt. Changes and change management in construction and IT projects. *Automation in Construction*, 20(7):767–775, February 2011.

Tyson R. Browning, Ernst Fricke, and Herbert Negele. Key concepts in modeling product development processes. *Systems Engineering*, 9(2):104–128, 2006.

Kim B. Clark and Takahiro Fujimoto. *Product Development Performance: Strategy, Organization and Management in the World Auto Industry*. Harvard Business School Press, Boston, Massachusetts, USA, 1991.

Mike Danilovic and Håkan Börjesson. Managing the Multiproject Environment. In *The Third Dependence Structure Matrix (DSM) International Workshop*, pages 1–17, 2001.

N. Do, I. J. Choi, and M. Song. Propagation of engineering changes to multiple product data views using history of product structure changes. *International Journal of Computer Integrated Manufacturing*, 21(1):19–32, January 2008.

Claudia Eckert, P John Clarkson, and Winfried Zanker. Change and customisation in complex engineering domains. *Research in Engineering Design*, 15(1):1–21, March 2004.

Claudia Eckert, John Clarkson, and chris Earl. Predicatability of change in engineering: a complexity view. In *ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, pages 1–10, September 2005.

Tido Eger, Claudia Eckert, and P John Clarkson. The Role of Design Freeze in Product Development. In *International Conference on Engineering Design ICED 05*, pages 1–11, August 2005.

Electronic-Industries-Alliance. EIA-632 Standard, January 1999.

Ernst Fricke, Bernd Gebhard, Herbert Negele, and Eduard Igenbergs. Coping with Changes: Causes, Findings, and Strategies. *System Engineering*, 3:169–179, 2000.

Roland Gareis. Changes of organizations by projects. *International Journal of Project Management*, 28(4): 314–327, May 2010.

Carley M Kathleen and Krackhardt David. A Typology for C^2 Measures . In *The 1999 International Symposium on Command and Control Research and Technology*, pages 1–12, 1999.

Danilovic Mike and Browning Tyson R. Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, pages 300–314, 2007.

M. Z. Ouertani and L. Gzara. Tracking product specification dependencies in collaborative design for conflict management. *Computer-Aided Design*, 40(7):828–837, July 2008.

Kamel Rouibah and Kevin R. Caskey. Change management in concurrent engineering from a parameter perspective. *Computers in Industry*, 50:15–34, 2003.

A. Rutka, Marin D. Guenov, Y. Lemmens, T. Schmidt-Schäffer, P. Coleman, and A. Rivière. Methods for engineering change propagation analysis. In *25th International congress of the aeronautical sciences*, pages 1–8, 2006.

Nam.P. Suh. *The Principles of Desing*. Oxford Series on Advanced Manufacturing Series. Oxford University Press, Incorporated, 1990. ISBN 9780195043457. URL `http://books.google.fr/books?id=Z5fF5qQPQ9sC`.

Browning R. Tyson. Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, 48:292–306, August 2001.

Karl T. Ulrich and Steven D. Eppinger. *Product Design and Development*. Mc Graw-Hill, fourth edition, 2007.

Yasushi Umeda and Tetsuo Tomiyama. Functional Reasoning in Design. *IEEE Expert*, 12(2):42–48, April 1997.

Michael Van Wie, Cari R. Bryant, Matt R. Bohm, Daniel A. Mcadams, and Robert B. Stone. A model of function-based representations. *AIE EDAM*, 19(02), August 2005.

Marc Zolghadri, Claude Baron, and Philippe Girard. Modelling mutual dependencies between products architecture and network of partners. *International Journal of Product Development*, 10:62–86, October 2010.

S. Zouggar, M. Zolghadri, and P. Girard. Modelling Product and Partners Network Architectures to Identify Hidden Dependencies. In *Proceedings of the 19th CIRP Design Conference*, pages 32–39, 2009.