

Managing Dynamic New Product Development Processes

Seventeenth Annual International Symposium of the
International Council on Systems Engineering (INCOSE)
24 - 28 July 2007

Arie Karniel
ariel@eng.tau.ac.il

Yoram Reich
yoram@eng.tau.ac.il
School of Mechanical Engineering, Tel Aviv University, Tel Aviv, Israel

Copyright © 2007 by Arie Karniel, Yoram Reich. Published and used by INCOSE with permission.

Abstract. New Product Development (NPD) processes are considered most challenging, involving major risks due to unknown or unforeseen obstacles, in terms of technology and business risks. The actual process activities which depend on the evolving product knowledge could be determined only during the process. Thus, process planning is inherently dynamic and requires adaptation to product knowledge changes as well as other changes. Current Workflow tools can support *ad-hoc* changes, but do not support the planning of process dynamics and the execution of such dynamic process changes as they unfold.

The current article presents a system framework for managing dynamic process planning changes resulting from changes in customer requirements, product structure, product parametric dependencies and constraints, as well as *ad-hoc* changes. The proposed framework comprises: process planning, incorporating the Design Structure Matrix (DSM) method; business rules for interpreting the DSM-based plan to process plan; dynamic process plan changes; and implementation of changes into Run Time process simulation.

1 Introduction

A Product Development Process (PDP) is a collection of related activities targeted to convert a new idea, concept, or market opportunity, into a marketed product. PDPs are highly complex and dynamic. They are complex because they involve multiple disciplines contributing to the development of complex multidisciplinary products (e.g. mechatronics), with limited resources, shortened development time, and increased quality and regulatory demands. They are dynamic, because they evolve continually due to various reasons including market changes, technological changes, and organizational changes (Westfechtel, 1999; Fricke *et al.*, 2000; Otto and Wood, 2001).

Amongst all PDPs, New Product Development (NPD) processes, present additional complication to product developers: developers' knowledge of the new product being designed is initially poor and only increases during the development process. Given this, we cannot hope to define the scope of work needed for a new product, nor could we plan how to manage it *a-priori* (Westfechtel, 1999). Therefore, change management and its derivative, design iterations, become major concern in NPD processes and in particular, in product design activities (Fricke *et al.*, 2000; Yassine and Braha, 2003). Iterations are considered as major source of increased product development lead-time and cost (Browning and Eppinger, 2002; Eppinger *et al.*, 1997).

In spite of the complexities of NPD processes, product developers have to act but unfortunately, without much decision support tools. Shane and Ulrich (2004) surveyed 31 Development Process management articles published in *Management Science*, and state that "...most of this work must be characterized as 'insight models,' with almost none of it resulting in new decision support tools." (p. 136). Furthermore, the existing workflow management tools, execute fixed workflow process models, not supporting changes in process scheme (Weske *et al.*, 2004; Heller and Westfechtel, 2003), and not supporting activity decomposition and recursive activity execution (Madhusudan, 2005).

Planning the process and performing project activities in an appropriate sequence is critical to minimizing rework due to information dependencies. Introduced by Steward (1981), the Design Structure Matrix (DSM) is utilized to capture the required product knowledge for process planning. Yet, the DSM does not represent any information regarding the logic of the linkages between activities, and the presented information is insufficient for implementing process simulations (Smith and Eppinger, 1997; Clarkson *at al.*, 2000; Yassine, 2006).

When we want to integrate process planning (via DSM) and execution (through workflow tools), the limitations of both are exacerbated. In the first step of process planning, DSM representations of product data give rise to many process interpretations (Karniel and Reich, 2006a). When these interpretations, defined as *Business Rules*, are formulated and organized, they show how to translate DSM-based plan to a concurrent process plan model, the *Process Scheme*. Formalizing these business rules, including the case of self-iterations (Karniel and Reich, 2006c), allows constructive verification of the process scheme. The verification criteria are based on Work-Flow (WF) nets (Aalst, 1998); and include additional requirements which are specific to design processes.

Dehnert and Aalst (2004) emphasized the difference between business model descriptions and the unambiguous workflow specification required for their implementation at the process execution step. The Dynamic PDP (DPDP) approach presented in this article consolidates process planning and re-planning subject to changes, dynamic process implementation, process execution, and simulation. In particular, we focus on a method for aiding decision making at run time when NPD process scheme has to change due to evolving product knowledge.

While we focus on NPD processes, we take a broad interpretation of the term "product." A product could be a consumer product, a large infrastructure system, a service, an enterprise, or any system that is being designed or modified to address new needs. For the larger, more complex systems, the capabilities offered by the proposed framework are ever more critical.

In what follows, section 2 briefly surveys the DSM method, process verification literature, the basic cycle from planning to implementation, and adaptive processes literature. Section 3 describes the proposed framework, which is illustrated in an example case study in Section 4. Conclusions are drawn in section 5.

2 Literature Survey

Before starting, it is important to define the terminology used in this article. The term *Activity* is used to describe a logical step within a process. Activities have pre- and post-conditions; additionally, they may accept inputs and share outcomes during their execution (Cho and Eppinger, 2005). The term *Process Scheme*, is used to describe the process structure, i.e., the model of linkages between activities, their precedence, concurrency, and logic relations, which is used for plan implementation (Karniel and Reich, 2006a). The term *Run Time process*, is used to describe the process progress according to the *Process Scheme*. Due to the uncertain and iterative nature of the process, numerous valid Run Time processes could be generated (Karniel and

Reich, 2006c). The term *Product knowledge* describes a limited set of product related knowledge being reflected in the *process scheme*; thus, changes in the former may drive *process scheme* changes.

DSM. The commonly used GANTT and Pert charts are inadequate for planning design processes, as they do not effectively model design activities dependencies and process iterations (Steward, 1981). The DSM representation can be used for various applications (Browning, 2001). In relation to process planning, the DSM provides the means to identify serial, parallel and iterative design flows; and model and manipulate iterations and multidirectional information flows (Eppinger *et al.*, 1994).

DSM is a square matrix that uses off-diagonal entries to signify the dependency of one element upon another. When modeling processes, the lower diagonal portion represents a precedent activity relationship; i.e., a marking in cell $\{j,k\}$ (row j , column k , $j>k$), indicates that activity k should follow activity j . For each activity, its row shows its inputs and its column shows its outputs. The upper diagonal portion of the DSM matrix denotes an iterative process (a link to an upstream activity). The diagonal values are typically left empty, or used for activity properties such as duration or self-iteration probabilities (Karniel and Reich, 2006c).

The outcome of a particular activity can directly impact a previously performed activity, which should be performed again, i.e., rework. Serial activities are linked by forward link, and indicate a serial process. Parallel or concurrent activities have no relation linkages. Coupled activities have forward and feedback links, and they form an activity loop (cycle).

Process verification. Process correctness verification is profoundly discussed in the Workflow literature (Rinderle *et al.*, 2004a; Sadiq *et al.*, 2004). An established formal languages model for process specification is Petri net (Reising and Rozenberg, 1998). Petri net is a bipartite directed graph with two node types called places and transitions, connected by directed Edges (arcs). Places can contain Tokens. A transition may 'fire', i.e., transfer tokens from its input places to its output places according to 'firing rules'. Process activities are modeled by transitions; places correspond to conditions. Marking, representing the process state, is the distribution of tokens over the places. Petri nets can be used as diagrammatic tool for modeling and analyzing distributed system behavior including sequential, conditional, parallel and iterative routing (Aalst and Hee, 2002).

WF nets (workflow nets) (Aalst, 1998) are Petri nets, used for specifying the control of workflow processes with defined starting and ending. Using the formal definitions of a WF net, one can set the required conditions for proper process specification (*Process Scheme*), using a correctness criteria named '*Soundness*'. WF net properties are (Aalst, 1998): 1) A WF net has starting place and final place. 2) All the activities and places are on a path from the starting place to the final place (i.e., no activities or conditions that do not contribute to process progress).

Proper process, defined according to the soundness criteria has the following properties: 1) From every process state, which is reachable from the initial state (token at the starting place); there is a 'firing sequence' that leads to termination state (token at the final place). The process should terminate eventually. 2) Once the terminal state was reached there are no activities that did not perform yet; formally, there are no tokens in places other than the final place (i.e. the termination state). 3) There are no inactive activities which could not execute due to unmet pre-conditions.

Translating from DSM to Process Scheme. Karniel and Reich (2006b) presented a translation procedure with the following stages: (1) Defining design activities. (2) For each design activity,

defining the parameters that influence other design activities. (3) For each parameter, assigning the influence value it may have on changes of design and setting the influence direction. (4) Summing the values for each influence direction between the design activities (Impact DSM). (5) Assigning probability values (preliminary probability DSM) and scaling them; thus, creating a Probability DSM (Yassine, 2006). The above procedure steps resemble the procedure in (Sered and Reich, 2006). (6) The Probability DSM was reordered using optimal Sequencing and Clustering algorithm (Karniel *et al.*, 2005). Yet, other reordering algorithms could be utilized (e.g., Meier *et al.*, 2006). (7) Merging Design-blocks (grouped activities) and calculating probabilities. (8) Translating the DSM into a Design Process Matrix (DPM); i.e., explicitly adding logic activities (process Begin, and End activities; Input logic and Output logic per each design activity; Design-block Begin and End activities). Adding logic is done according to *Business Rules* that are set according to business requirements. (9) Converting the DPM into an equivalent Current process scheme (C- Process). (10) Using the process scheme for creating a Run Time process (RT-process).

Process Dynamics classification. A process case (i.e., the development of a certain product) is dynamically evolving at run time according to the process scheme. We can make the following dynamic process changes classification: Class I – Constant process scheme dynamics: The dynamics will include the actual decisions being made and resource allocation; in iterative design processes, the activity content may change in each iteration, leading to different types of activity outcome at each iteration (O'Donovan *et al.*, 2003). Class II – Case Handling: It is actually a sub category of class I, since the process is actually pre-defined to a detailed level. Yet from the user perspective, the process scheme details are elaborated and changed at run time (Aalst *et al.*, 2005). Class III – *ad-hoc* changes: manually changing the process scheme in unpredicted (and typically uncontrolled) manner (Dustdar *et al.*, 2005). Class IV – Dynamic change of the Process Scheme: The changes manifest in the process scheme itself, e.g., adding or deleting activities at each iteration (Rinderle *et al.*, 2004b); or changes of the activity order at different iterations based on changes in knowledge (Clarkson *et al.*, 2000; Lévárdy and Browning, 2005).

Adaptive processes. In NPD processes, the product related knowledge is partial at the beginning, and evolving during the process. Since process scheme changes (activities and their links) are anticipated, this above cycle of planning, modeling, verification, execution and simulation should be repeated. Such Process Management Cycle approach is required for every process which is both iterative and changing due to evolving knowledge (Rinderle *et al.*, 2004a; Lévárdy and Browning, 2005). Any constant scheme approach, which can be successfully employed under other conditions, fails to support the requirements of NPD processes (and other less challenging design processes). Rinderle *et al.* (2004a) indicated that “current adaptive WfMS do not allow propagating WF type changes to individually modified WF instances”, which is important for the adequate support of long-running workflows, or in our case, for unique processes.

A change of administrative processes should typically apply to all the process cases, which may be in different process stages (Rinderle *et al.*, 2004a). A survey of adaptive workflow changes for administrative processes appeared in (Klein and Dellarocas, 2000). In Reichert *et al.* (2005), a combined approach for managing multiple process cases was described: *Process Migration* (i.e., replacement of process segments) was performed for most process cases; if the changes could not be implemented, the cases were *Flashed* through the changed process

segments; *Ad-hoc* changes followed process *Adaptation*, i.e., each case had its individual process instance segments which differed from the general process.

However, design processes, and particularly NPD processes are typically unique; thus, the change is local to one case only, and requires *Build* approach; building the process at runtime in correspondence to its particular situation and knowledge. The latter approach is used in this article.

The AHEAD system (Westfechtel, 1999; Heller and Westfechtel; 2003) was specifically designed to support design processes providing flexibility and decision support capabilities. The process is modeled by Task net that evolves during execution; it is product dependent; it enables simultaneous engineering, and feedback; tasks are arranged in a hierarchy; and reactivation of a task creates new task version, allowing traceability of changed processes. However, process planning and modeling are done manually by the user. The implementation concepts used in AHEAD are the foundation of controlling the process scheme dynamics in the current work. The additional layer of the current work is the automation of the process planning, modeling and simulation based on product knowledge.

3 Dynamic Product Development Process (DPDP) framework

A simple linearized process development model typically includes process stages as depicted in Figure 1. Concentrating on the phases from Concept to Production, and utilizing a more practical modelling approach that includes parallel activities, decision points (e.g., Stage Gates model; Cooper, 2001), and iterations, might yield for example, a high level process model depicted in Figure 2. The Specification activity starts in parallel to the Conceptual Design. Decisions regarding iterations of activities can be made in Preliminary Design Review (PDR) or Critical Design Review (CDR). The decision taken may be continuing the project, iterate, or project cancellation. The process described specifically indicates Begin and End activities that are required for process implementation (Karniel and Reich 2006b).



Figure 1. Development process model.

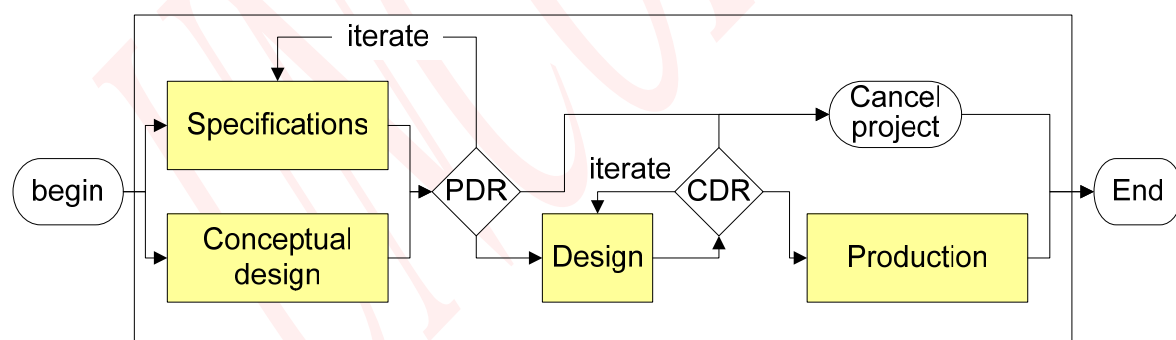


Figure 2. High level process model.

At the high level process definition, the process model might be considered relatively invariable and may apply to many products. We shall refer to such predefined process models as

Process Skeleton, as their actual activity content is defined according to the product. Typically, industrial development organizations have different high level development process skeletons, which represent the organization best practices. Existing Workflow tools can manage such skeleton processes with some *ad-hoc* modifications or more elaborated pre-defined sub processes being added at run time, i.e., Case Handling.

Managing the product-specific details activities, at sub process level, based on the product knowledge (e.g., modeled by the DSM method), is beyond the capabilities of current tools. Furthermore, as in NPD, the product structure is not completely defined at the conceptual design stage, the DSM is not static. As the DSM changes according to the product knowledge, the process scheme based on the DSM should also change. Thus, the above procedure of DSM translation that reflects one cycle from planning to implementation should iterate each time a change in knowledge occurs.

The process in Figure 3 describes the framework controlling mechanism of the dynamically changing design process with dynamic process scheme changes. There are two types of entries: *external* entries, which are not aligned with process propagation and might change at any time; and *internal* entries, which are dependent on the actual progress of the process, and include the process status and the product knowledge which is developed during the process. The external entries are: skeleton processes (pre-defined best practices); *Ad-hoc* changes; product requirements; and resource constraints. The internal feedback entries are: the process status, which is assumed to be defined within the system; and product knowledge which is manually entered by developers and is the sources of DSM calculations. The framework includes two main blocks: the *process generator* that generates the dynamic process schemes, and a workflow engine that executes the run time process according to the scheme. The workflow engine is assumed to be equivalent to current workflow engines, with resources optimization capabilities. The current work is focused on the process generator and its interaction with operating process engine.

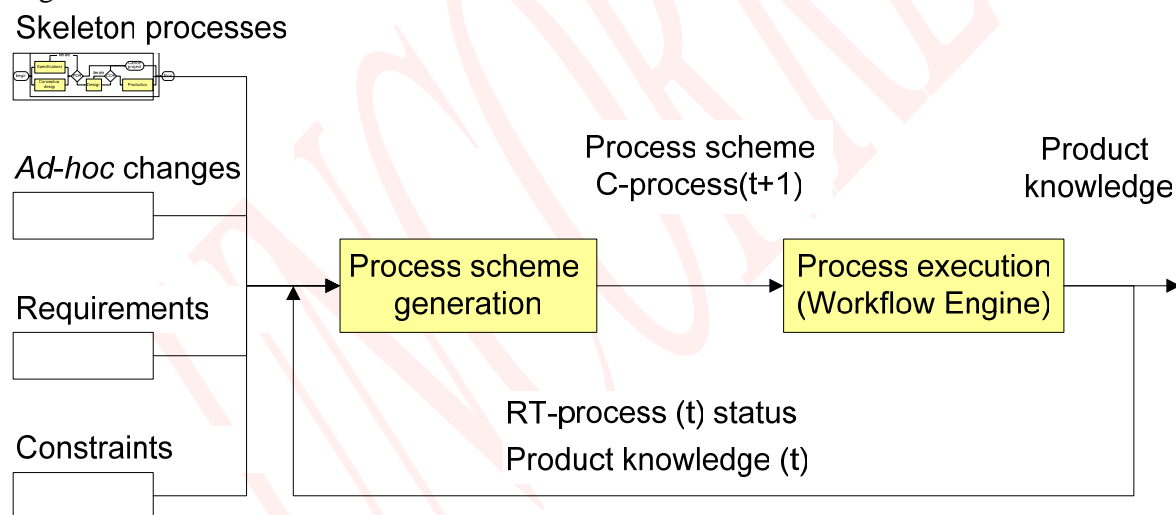


Figure 3. DPDP framework.

Figure 4, depicts the iterative meta-process procedure of the process generator. The diagram presents activities (procedures or processes) followed by results (information or model), and manual entries of information (e.g., product knowledge, business rules). Current input information at time t is utilized for planning the process for the next time step $t+1$. The information created at the next time step $t+1$ becomes the information used for the next meta-

process calculation. Manual entries of product requirements and *ad-hoc* process changes are not implemented yet, but the infrastructure for their implementation exists.

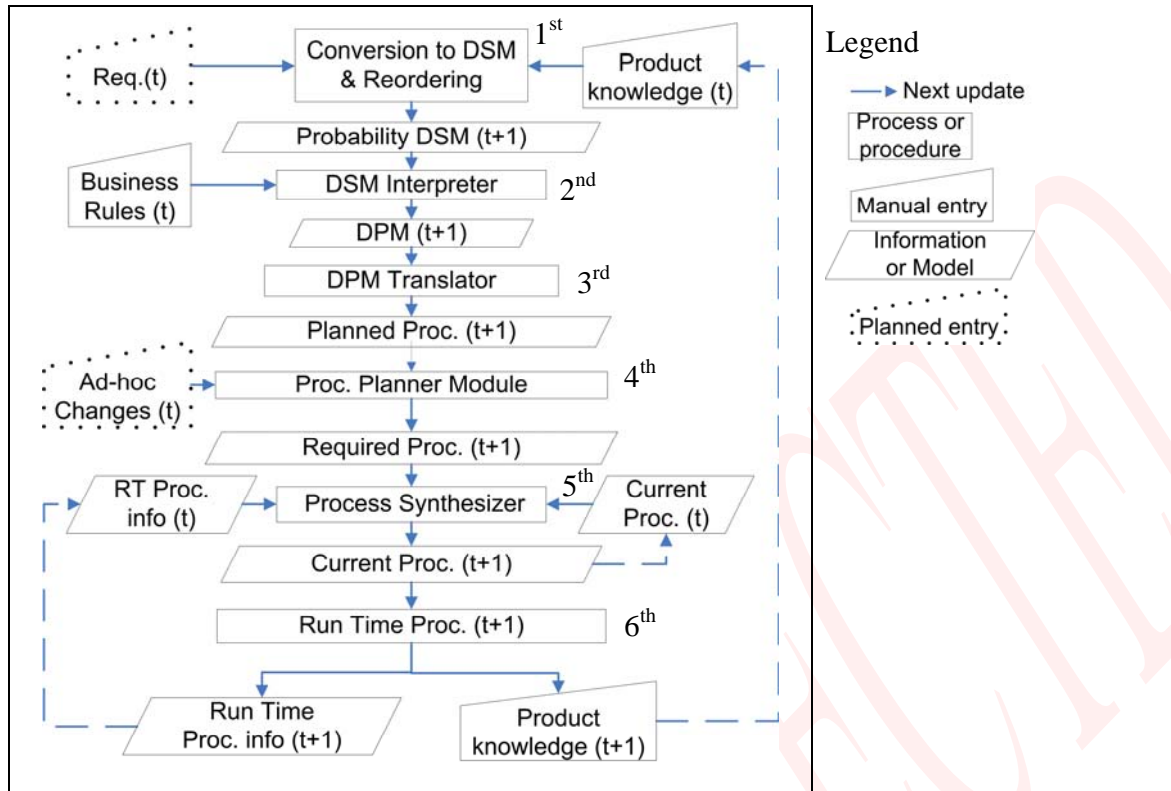


Figure 4. Meta-Process model.

In the first step, the product knowledge is transformed into DSM. Product knowledge includes the following: product structure translated to design activities at the required detail level (typically, the work of one resource, e.g., designer, applied to a product component, e.g., sub-assembly, will be considered as one activity); and influence of changes in the design created by one activity on design work performed by other activities. Methods of acquiring such information are described in (Sered and Reich, 2006; Yassin, 2006). It is assumed that such information can be captured and managed by Product Data Management (PDM)/Product Lifecycle Management (PLC) systems. The transformation sub-steps (1) to (6), previously described (in section 2), were detailed in several articles (e.g. Karniel and Reich 2006b). Sub-step (7) is demonstrated in next section example.

The second step – interpretation of the ordered DSM into DPM (sub-step 8) – includes the usage of business rules that define the process logic that is not described by the DSM. These business rules might be set manually, or be changed during the process; thus, the rules defined at time t apply to the planning of process scheme for time $t+1$.

In step three, the DPM is translated into the Planned Process (sub-step 9). The Planned Process is an optimization of the process plan subject to the given information and assuming the process starts now. This assumption was used in (Karniel and Reich 2006b), and is the typical assumption used in DSM-based simulations. Most workflow tools also assume one time planning. The process scheme is then used for creating the Run time process.

The novelty of our new approach is the concepts of process planning updates that lead to updated process scheme at each time step. In step four, updating the process scheme requires integration of the *planned process*, based on the current product knowledge (the process plan as if the process starts from scratch); and any *ad-hoc* changes to the process (e.g. changes to the skeleton process, or adding a new activity such as prototyping to reduce development risk), yielding the *required process*. The *required process* is still an abstraction refraining from the actual process status.

In step five, the Process Synthesizer utilizes the *required process* (planned for $t+1$), the *current process scheme* $C\text{-Process}(t)$ and the actual status of the run time process $RT\text{-process}(t)$, and calculates the *current process scheme* for the next time step $C\text{-Process}(t+1)$. Differences between the *required process*($t+1$) and the current process, $C\text{-Process}(t)$ might not be immediately implemented to $C\text{-Process}(t+1)$ since the run time process should confirm with correctness criteria. Thus, the implementation of the *required process* might be postponed to a later time step. The rules of application are beyond the scope of this article. The correctness criteria being applied are:

- The *Soundness* criteria, which were defined for WF nets.
- Executing each design activity at least once (Karniel and Reich 2006b).
- Proper termination of design activities.

Implementation Rule (IR): Termination of design activity may be achieved by the following Business Rule (BR) options:

- Immediate termination – the activity is abandoned.
- Termination by a milestone.

The second option indicates completion of meaningful deliverables of the applicable design activity (e.g., completing a drawing of an existing design to comply with drawing standards, or completing a finite elements analysis for estimating a parameter). In this case, additional rules should be set for estimating activity duration within a simulation. Additionally, the implementation of a specific process scheme may be postponed. For example, if the activity is the only source of another activity (i.e., the latter cannot start if the former did not complete), the former activity must complete with a desired (partial) results, otherwise the next activity may not start (have a deadlock problem).

The resulting process scheme is the outcome of the process scheme generator (Figure 3), and becomes the input of the workflow engine. It should be noted that unlike the typical existing workflow engines, this workflow engine manages process with dynamically changing process schemes.

The RT-process, being managed by the workflow engine, has two outcomes: its status information and the product knowledge gained. If no changes have occurred, the process scheme at time $t+1$ ($C\text{-Process}(t+1)$), may be identical to the current process $C\text{-process}(t)$, thus the run time process just follows the current process scheme.

4 DPDP Example

The example product is a simplified model of laser direct imaging (LDI) plate/image-setter for the prepress market industry; presented in Sered and Reich (2006). This imaging system is based on external drum technology applying direct exposure of high laser power on plate panels and films to convert digital input into finished panels ready to be used in standard press-printing process. A metal drum cylinder revolves about its axis allowing a fast motion imaging axis. A laser exposure head, adjacent to the drum perimeter, driven by a smooth and precise mechanism,

parallel to the drum axis, allows a slow motion imaging axis, see Figure 5(a). The flexible plate/film is held around the drum by a registration system using edge clamps and device for punching edge holes. Load/unload system draws a plate from an automatic cassette unit or a manual tray. After the imaging process terminates, it unloads the plate to a special fitting in the back of the machine. A computerized control system manages the process.

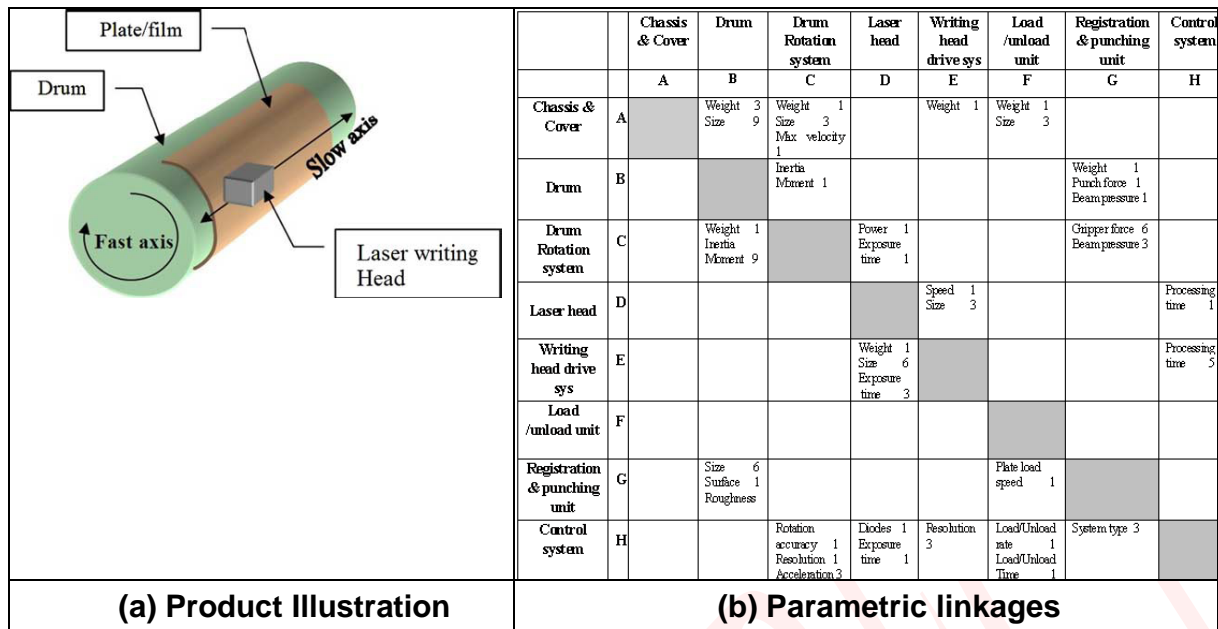


Figure 5. External drum LDI plate / image-setter technology.

The product knowledge is represented in Figure 5(b). The matrix Rows (Columns) are the product components (sub systems). In each cell there is a list of parameters and figures. The figure indicates the influence of a change of component (column) on another component (row); for example: a change in the design of component B (Drum), has influence value 9 on a required design change in component C (Drum registration system) due to inertia moment parameter, and influence value of 1 due to change in weight. The influence figures are directed, i.e., the influence may not be symmetric. Summing the figures in each cell yields a Numeric DSM in Figure 6(a). The DSM is linearly converted to Probability DSM, thus the maximal value 12 converts to maximal probability 0.52 (Yassine, 2006); and reordered, Figure 6(b). In the current example, only activity F has self-iteration probability. Other activities have zero self-iteration probability (diagonal values).

Using a one-to-one correspondence of components and design activities, the DSM structure indicates that design activity of component F (Load/Unload unit) is not affected by changes of any other subsystem. Design activity A is influenced by many other activities due to components influence, and therefore should wait until the other activities complete. All other activities are coupled, forming an activity loop. The optimization procedure (Karniel *et al.*, 2005) performs clustering and sequencing (including tearing), thus can separate the activities further into two closely interconnected activity groups (BGC and DHE), which become Design-blocks. The activities within a Design-block have parallel start and an aligned completion (i.e., all activities perform in parallel and considered as one unit). The Design-blocks are coupled and follow coupled parallel start rule; yet, their completion is not aligned.

	A	B	C	D	E	F	G	H		F	B	G	C	D	H	E	A		F	BGC	DHE	A
A	0	12	5		1	4			F	0.22								F	0.22			
B		0	1					3	B		0	0.13	0.04					BGC	0.04	0.74	0.09	
C		4	0	2				9	G	0.04	0.30	0	0.00					DHE	0.09	0.32	0.72	
D				0	4			1	C		0.17	0.39	0	0.09				A	0.17	0.626	0.04	0.00
E				10	0			5	D					0	0.04	0.17						
F						5			H	0.09		0.13	0.22	0.09	0	0.13						
G		7	0			1	0		E					0.43	0.22	0						
H			5	2	3	2	3	0	A	0.17	0.52	0.00	0.22			0.04	0					
(a) Numeric DSM									(b) Ordered Probability DSM									(c) Merged Design-blocks DSM				

Figure 6. DSM conversions and calculations.

The interrelation probabilities assigned to activities within a Design-block are assigned to the whole Design-block, using the following calculation:

$$Pd = 1 - \prod(1 - Pi)$$

Where Pd is the combined probability cell of a Design-block, and Pi are the merged probability cells. The probability cells which are on forward direction (sub diagonal) to design activities within the Design-block are merged to the probability of forward link to the merged Design block (e.g., the probabilities from activities in BGC Design-block, 0.13, and 0.22 to activities in DHE are merged to 0.32; probability of a forward link between the two Design-blocks). Feedback probabilities are merged in the same manner. The probabilities within the Design-block (which are not on the diagonal) are all considered as feedback links and are merged to the self-iteration probability of the Design-block, e.g., the Design-block BGC has a self-iteration probability of 0.74. The calculations outcome is depicted in Figure 6(c).

The implementation of a Design-block in DPM is done through the use of common Begin and End activities that are directly linked to, and from, each of the activities within the Design-block. Self-iterations are described in the DPM as feedback link from Output logic to the Input logic, or in case of a Design-block, from the Design-block End activity (De) to the Begin activity (Db). Using the merged presentation (all activities within a design block are presented as one activity), the resulting DPM is depicted in Figure 7(a). The abbreviations used: Input Logic (IL); Output Logic (OL); Design-block Begin activity (Db); and Design-block End activity (De). Each of the design activities within the Design-block has its own logic activities (IL, OL) in a similar manner to the activities F and A.

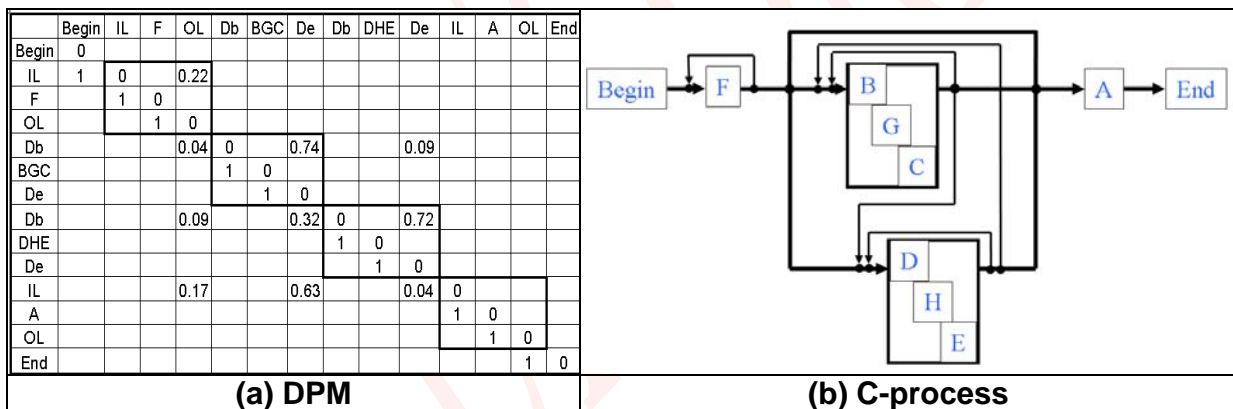


Figure 7. Translating DPM to process.

The DPM is translated into a process plan, Figure 7(b). Forward links are marked by thick lines. Only the process Begin and End logic activities are presented. Using the translation rules, in (Karniel and Reich 2006c) the following input logic and output logic apply. The logic activities Begin, IL and Db always send signals to all next activities (Split-And). The logic activities OL and De send signals to all the forward links on the first completion of the design activity (or design block), but send a signal according to the assigned probabilities on iteration of the activity (or design block). In case of iteration, a signal might be sent to the End activity. Activating OL and De is by waiting for all forward links. IL and Db logic is waiting for all forward links at the first execution of the design activity or design block, wait for any feedback link, or wait for additional signal of a forward link on iteration. The Planned process becomes the *required process* if no *ad-hoc* changes are added.

At the beginning of the design process, there are no run time status restrictions, and the required process becomes the current process. The run time process is executed according to the current process, and may perform iterations. In Figure 8(b) there are two iterations of activity F based on the Current process in Figure 7 (b).

As additional product knowledge is gained, the planned process may change; for example, two influence value estimations were changed (see figure 6(a)): the influence value estimation of H (control system) on E (writing head driving system) changed to 3 (instead of 5), i.e., estimating less influence of the controller processing time over the design of the writing head; and adding an influence estimation of the process time of H on G (Registration unit) with value 1. These changes will create other design blocks structure in the DSM (DE and BGCH), which is reflected in another *planned process*, depicted in Figure 8(a).

The implementation of such change depends on the Run Time status. For example, if the design blocks did not start (i.e., F is still executing), the change can be immediately set. Otherwise, if H activity has completed at least once, the following activities may start; thus, a change may apply immediately. However, if the change occurred during the execution of the Design block (DHE) containing activity H, then the implementation of the *required process* depends on the business rule being applied; for example, may the activity be interrupted (and restarted later) or should it reach a milestone?

The RT-process depicted in Figure 8(b) demonstrates the case of process scheme change (from Figure 7(b) to Figure 8(a)), being made during the second execution of H activity. The RT-process includes iteration of F and the first execution of the BGC and DHE Design-blocks. Then, DHE Design block iterates and the change in C-process (Figure 8(a)) occurs during the iteration of DHE, i.e., on the second execution of activity H. BCG design-block did not perform self iteration, but due to the iteration of H, and the applicable C-process scheme, the activities should iterate; and H completes its execution as part of BGCH Design-block. The remaining activities DE form the other Design-block. In this case, the Design-block duration (not presented graphically) may have changed, e.g., if H duration is long relative to other activities, the duration of the remaining DE Design-block might be reduced. As all the coupled activities completed their iterations, design activity A can start; once it completes the design process is completed.

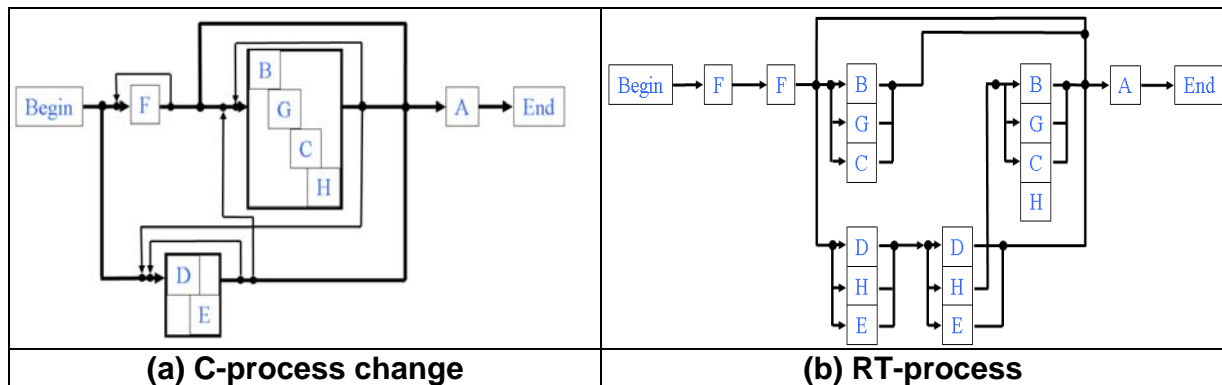


Figure 8. Implementing dynamic scheme change.

5 Discussion

The proposed approach requires the analysis of numerous cases and the detailing of logical steps to address them. While this is laborious, it is assumed that the set of rules driving the logic is sufficient to address most design process cases. This assumption should be further investigated.

Using logic activities assigned to the design activities adds an overhead, but allows separating the process-logic implementation layer from the process planning; such layering supports additional dynamics, as the logic being applied to logic activities may change during the process. The example being used is the change of logic in the first execution of a design activity versus the logic in following iterations. Other sources of logic changes may be changes in business rules according to simulation results; e.g., missing a project milestone may result in applying more parallel activities (with more resources).

The proposed system can be utilized as decision making aid, using its simulation capabilities. For example, consider a case where the supplier of the engine for the drum rotation system (C), which is a specially designed component, can no longer supply the engine. There are several optional counter measures including: trying to develop and manufacture an engine with the same specifications by another supplier, developing a new engine, or buy an existing engine by another supplier and make the appropriate changes in the product. Each option may have different impact on the product quality (not addressed here) and on the process effort and timing. They could all be simulated to provide some insight to decision makers.

For time estimations the activity duration is calculated. Process progress options are time dependent as there are behavioral differences if an activity that should iterate has completed; or is still executing. Calculations of activity duration can be done using the approach described for overlapping activities (Cho and Eppinger, 2005), which is extended to the Design-block case.

6 Conclusions

Executing a process is the only way to get things done. In a dynamic and turbulent world, almost anything changes. We presented a framework for managing dynamic processes including their dynamic scheme. Such framework allows monitoring dynamic processes as they unfold including the traceability between product knowledge changes and process changes; it allows simulating different changes thus providing a means for extracting process management knowledge; it serves as a complete record of the process that can be studied on-line in order to improve it further.

There is no limit to the type of process that could be modeled. While we concentrated on NPD processes being important for driving organizational success, these processes could be reorganization processes, mergers processes, or other related to small or large systems.

Several novel concepts were presented in relation to the framework, which overcome the limitations of current methods and tools. An integrated automated system is proposed performing: automated planning and re-planning of the process based on the evolving product data; automatic planned process scheme implementation based on (dynamically) applying business rules; integration of *ad-hoc* changes to form required process; calculation of the current process based on the run time process status; and finally implementation of process at run time.

7 Reference List

- Aalst W.M.P. van der, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
- Aalst W.M.P. van der, Hee K.M. van, *Workflow Management: Models, Methods, and Systems*, MIT Press, Cambridge, MA. 2002.
- Aalst, W.M.P. van der, Weske M., Grünbauer D., "Case handling: a new paradigm for business process support", *Data & Knowledge Engineering* 53(2): 129-162, 2005.
- Browning T.R., "Applying the Design Structure Matrix system to decomposition and integration problems: A review and new directions", *IEEE Trans. Eng. Manage.*, 48:292-306, 2001.
- Browning T.R., Eppinger S.D., "Modeling impacts of process architecture on cost and schedule risk in product development", *IEEE Trans. Eng. Manage.*, 49(4):428-442, 2002.
- Cho S.H., Eppinger S.D., "A simulation-based process model for managing complex design projects", *IEEE Trans. Eng. Manage.*, 52(3): 316-328, 2005.
- Clarkson P.J., Melo A.F., Eckert C.M., "Visualization of Routes in Design Process Planning", *In Proceedings of the Fourth International Conference on Information Visualisation (IV2000)*, IEEE Computer Society, pp. 155-164, London, 2000.
- Coates G., Duffy A.H.B., Whitfield I., and Hills W., An integrated agent-oriented approach to real-time operational design coordination, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17: 287–311, Cambridge University Press, 2003.
- Cooper R.G., *Winning at New Products*, 3rd ed. Cambridge: Perseus Publishing, 2001.
- Dehnert J., and Aalst W.M.P. van der, Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289-332, 2004.
- Dustdar S., Hoffmann T., and Aalst W.M.P. van der., "Mining of ad-hoc Business Processes with TeamLog", *Data and Knowledge Engineering*, 55:129–158, 2005.
- Eppinger SD, Whitney DE, Smith R, Gebala D, "A model-based method for organizing tasks in product development", *Res. Eng. Design*, 6(1):1-13, 1994.
- Eppinger SD, Nukala MV, Whitney DE, "Generalized models of design iterations using signal flow graph", *Research in Engineering Design*, 9:112-123, 1997.
- Fricke E., Gebhard B., Negele H., and Igenbergs E., Coping with changes: Causes, findings, and strategies, *Systems Engineering*, 3(4):169-179, 2000.
- Heller M., Westfechtel B., "Dynamic project and workflow management for design processes in chemical engineering", *Proceedings 8th International Conference on Process Systems Engineering (PSE 2003)*, Kuming, China, 2003.
- Karniel A., Belsky Y., Reich Y., "Decomposing the problem of constrained surface fitting in reverse engineering", *Computer-Aided Design*, 37:399-417, 2005.
- Karniel A., Reich Y., "From DSM based planning to design process simulation: A review of

- process scheme verification issues”, submitted, 2006a.
- Karniel A., Reich Y., “A coherent interpretation of DSM plan for PDP simulation”, submitted, 2006b.
- Karniel A., Reich Y., “Simulating design processes with self-iteration activities based on DSM planning”, submitted, 2006c.
- Klein M., Dellarocas C.A., “Knowledge-based approach to handling exceptions in workflow systems”, *Journal of Computer-Supported Collaborative Work*. 9(3/4):399-412, 2000.
- Lévárdy V., Browning T.R., “Adaptive test process – designing a project plan that adapts to the state of a project”, *Fifteenth Annual International Symposium of the International Council on Systems Engineering (INCOSE)*, 2005.
- Madhusudan T., “An agent-based approach for coordinating product design workflows”, *Computers in Industry*, 56:235–259, 2005.
- Meier C., Yassine A., Browning T., “Design Process Sequencing with Competent Genetic Algorithms”, *PDRL working paper # PDL-2006-03*, 2006.
- O’Donovan B.D., Clarkson P. J, and Eckert C.M., “Signposting: Modeling Uncertainty in Design Processes”, *International Conference On Engineering Design (ICED 03)*, Stockholm, 2003.
- Otto K.N, Wood K.L, *Product Design*, Prentice-Hall, Inc, New Jersey, 2001.
- Reichert M., Rinderle S., Kreher U., Dadam P., “Adaptive Process Management with ADEPT2”, *21st International Conference on Data Engineering (ICDE’05)*, pp. 1113-1114, 2005.
- Reising W., Rozenberg G., editors, “Lectures on Petri Nets I: Basic Models”, *Lecture notes in Computer Science*, vol 1491, 1998.
- Rinderle S., Reichert, M., Dadam, P., “Correctness Criteria for Dynamic Changes in Workflow Systems - A Survey”, *Data and Knowledge Engineering*, 50(1):9-34, 2004a.
- Rinderle, S., Reichert, M., Dadam, P., “Flexible Support of Team Processes by Adaptive Workflow Systems”, *Distributed and Parallel Databases*, 16(1): 91 – 116, July 2004b.
- Sadiq S., Orłowska M.E., Sadiq W., Foulger C., “Data flow and validation in workflow modelling”, *Proceedings of the International Conference in Research and Practice in Information Technology, ADC’2004*, Dunedin, New Zealand, 2004.
- Sered Y., Reich Y., “Standardization and modularization driven by minimizing overall process effort”, *Computer-Aided Design*, 38(5):405-416, 2006.
- Shane S.A., Ulrich K.T., “Technological Innovation, Product Development, and Entrepreneurship in Management Science”, *Management Science*, 50(2):133-144, 2004.
- Smith R.P., Eppinger S.D., “Identifying controlling features of engineering design iteration”, *Management Science*, 43(3):276-293, 1997.
- Steward D.V., *Systems Analysis and Management: Structure, Strategy, and Design*, Petrocelli Books, NJ, 1981.
- Weske M., Aalst, W.M.P. van der, Verbeek H.M.W., Advances in business process management, *Data and Knowledge Engineering* 50(1):1-8, 2004.
- Westfechtel B., Models, and tools for managing development processes, *Lecture Notes in Computer Science 1646*, Springer, Berlin, 1999.
- Yassine, A., “Investigating product development process reliability and robustness using simulation”, *Journal of Engineering Design*, in press, 2006.
- Yassine A., Braha D., “Complex Concurrent Engineering and the Design Structure Matrix Method”, *Concurrent Engineering Research and Applications*, 11(3):165-176, 2003.

8 BIOGRAPHY



Arie Karniel is a PhD student in the Faculty of Engineering, Tel Aviv University, Israel. He received his BSc and MSc degrees in Mechanical Engineering from the Technion, Israel, in 1989 and 1991 respectively. He received his MBA, information systems, from Tel Aviv University in 1997. Arie has practiced engineering design, product management and system engineering in aerospace, machatronics, and software industries. His research interests include: design processes, collaborative design, reverse engineering, workflow management, product life cycle management, and engineering knowledge management.



Yoram Reich is an Associate Professor in the Faculty of Engineering, Tel Aviv University, Israel. He received his BSc (Summa Cum Laude) and MSc (Magna Cum Laude) in Mechanical Engineering from Tel Aviv University in 1980 and 1984, respectively. Before obtaining the PhD degree in Civil Engineering from Carnegie Mellon University in 1991, he practiced engineering design for over 7 years in the audio, structures, and marine industries. Yoram Reich has authored or co-authored over 100 papers and is a member of the editorial boards of the journals *Advanced Engineering Informatics*, *International Journal of Mass customization*, and *Research in Engineering Design*. His research focuses on several interrelated topics: product design, conceptual design, collaborative design, machine learning, and knowledge acquisition techniques for engineering applications,

9 Contact Information

Yoram Reich, School of Mechanical Engineering, Tel Aviv University, Tel Aviv, Israel. e-mail: yoram@eng.tau.ac.il; phone: +972-3-640-7385; fax: +972-3-640-77617