CHANGE PROPAGATION MANAGEMENT BY ACTIVE BATCHING

Oh, Gyesik; Hong, Yoo S.
Seoul National University, Republic of Korea (South Korea)

Abstract

Incremental design causes changes to an existing product by modifying or adding sub-systems. In the perspective of development process management, changes prolong duration and raise cost with increasing design change jobs. The change management of a complex product is challenging since changes are stochastically propagated to multiple sub-systems. The present study proposes the novel change management method, active batching, to accommodate the complex and uncertain characteristics of change propagation. Active batching composes multiple probable change requests as a batch, based on the prediction of change propagation. It saves cost by eliminating multiple setups and redundant execution. It also avoids unnecessary waiting to compose a batch, which accelerates development process. Numerical study on software development project validates that active batching moves Pareto frontier line of project performance into the direction of less cost and short duration.

Keywords: Design process, Project management, Uncertainty, Complexity, Active batching

Contact:
Gyesik Oh
Seoul National University
Industrial Engineering
Republic of Korea (South Korea)
gushigi4@snu.ac.kr

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 4: Design Methods and Tools, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

In the fast-changing market, incremental design is widely adopted to develop new products under limited duration and budget. Firms are required to develop new products periodically to keep the pace of demand and industry clockspeed. Customers expect new products with higher performance and quality (Ollinger and Stahovich, 2004). Also, competitors release products with added or enhanced attributes (McMahon, 1994). Incremental design is beneficial for companies aiming to create new generation products with tight development timeline. Since incremental design utilizes existing product architecture and modules, it takes less time and cost to develop new products with enhanced reliability (Eckert et al., 2009).
Incremental design involves engineering changes and their propagation in the perspective of product development management. Incremental design enhances the utility of a new product by upgrading existing attributes or including new attributes (Koh et al., 2009). To reinforce or add product attributes, existing sub-systems are modified or new sub-systems are supplemented (Wyatt et al., 2009). Since there exists interdependency among sub-systems, modification or addition of a sub-system might lead to the changes of other sub-systems (Clarkson et al., 2004). Although the sub-system is not directed related to the enhanced or added attribute, it might be modified because of propagated changes from other sub-systems.
Change management is critical for development performance, cost and duration. In complex product structure, the change of a certain sub-system might be propagated to multiple sub-systems within a product like avalanche (Eckert et al., 2004). Engineering changes generate additional design jobs to adjust existing sub-systems on change requests (Ahmad et al., 2010). Therefore, expenditure and duration of development project increase as engineering changes surge. However, it is difficult to control engineering changes since they are propagated uncertainly (Clarkson et al., 2004). The stochastic propagation hinders the exact prediction of change request arrivals on each activity. The unsettled development process is the obstacle to planning and controlling a development project for managers.
The present study proposes the new change management method, *active batching*, in the incremental product development project. Since changes are propagated complexly, a sub-system might receive multiple change requests from other sub-systems. Existing change management models accumulate change requests and execute them together to retain further propagation and prevent redundant work (Terwiesch and Loch, 1999; Loch and Terweisch, 1999; Du et al., 2015). However, those studies cope with propagation uncertainty passively by waiting for probable change requests with the criteria as the given time interval or the given number of changes (Nadia et al., 2006; Ahmad et al., 2010) or pausing the execution of change requests until propagation sources are inactivated (Maier et al., 2015). Active batching determines the batch, based on the prediction of upcoming possible change requests. To support decision, the present study provides the model which estimates the consequences of batching decision on duration and cost. In the practical perspective, the active batching provides managers the flexible planning and control method against changes in incremental development project. Compared to existing management methods, active batching moves Pareto frontier line of project performance into the direction of less cost and short duration.

2 CHANGE PROPAGATION MANAGEMENT

Change propagation management alleviates the indirect impact of incremental design on development process. For the given scope of incremental design, direct impact on development process is easily assessed with the aforementioned analysis by clarifying corresponding activities. However, indirect impact in incremental design is difficult to assess and manage. Indirect impact refers to design change jobs caused by propagation from directly impacted corresponding activities. The proper change management method prevents unnecessary design change jobs by controlling complex change propagation. Since change is propagated stochastically in the complex manner, change propagation management is challenging. The performance of development process, cost and duration, depends on change propagation management methods.
Change propagation management is complicated due to complexity and uncertainty of change propagation. In complex product architecture, change requests are generated by multiple sub-systems and transferred via multiple sub-systems as well (Ahmad et al., 2010). Therefore, it is not trivial for a certain sub-system to track the propagation paths from multiple change sources. Also, the change arrival is uncertain since change is propagated in the stochastic manner (Clarkson et al., 2004). The change management

plan based on the assumption of possible change requests might be invalid if the anticipated change request does not arrive.

Accumulation of change requests has been introduced as the management method to save cost. A development activity consists of setup and execution. The setup is required for engineers to be mentally ready for change requests as “diving deep into the problem” (Ha and Porteus, 1995; Terwiesch and Loch, 1999). By piling up change requests, an activity can deal with multiple change requests as a single job. Therefore, the setup is curtailed from multiple times to one, compared to the case that each change request is handled separately. Also the amount of execution can be reduced by eliminating redundant modification (Wynn et al., 2014). If the corresponding design space required to be changed for multiple change requests is overlapped, the simultaneous modification of intersection saves cost. Generally, amassing change requests prolongs project duration since an activity delays execution to compose the batch. However, under the complex change propagation, accumulation accelerates development process by resolving congestion (Loch and Terwiesch, 1999). When too many change requests arrive at a certain sub-system, overall development process is delayed since the design activity of the sub-system design becomes the bottleneck. Accumulation mitigates the bottleneck effect by reducing the amount of work and required execution time.

Existing change management methods can be categorized as how they respond to the uncertainty of upcoming change requests as Figure 1. Under the strategy of pending, an activity waits for the end of all precedent activities which might propagate changes to itself (Maier et al., 2015). Pending is the most risk-averse strategy since it delays performing the change job until change propagation risk perishes. By minimizing propagation risk, the amount of cost is minimized whereas the project duration is prolonged. Prompt execution adopts change requests immediately and does not consider the uncertainty of further change requests (Maier et al., 2015). Prompt execution represents the non-management method as the comparison of other management methods. It accelerates project process since there is no waiting for probable forthcoming changes. However, project cost increases since it does not take advantage of cost saving in the accumulation of change requests. Passive batching somewhat considers forthcoming probable propagation by amassing change requests for the given time interval (Loch and Terwiesch, 1999; Nadia et al., 2006; Ahmad et al., 2010). The cost and duration of project implemented with passive batching are between those of prompt execution and pending.

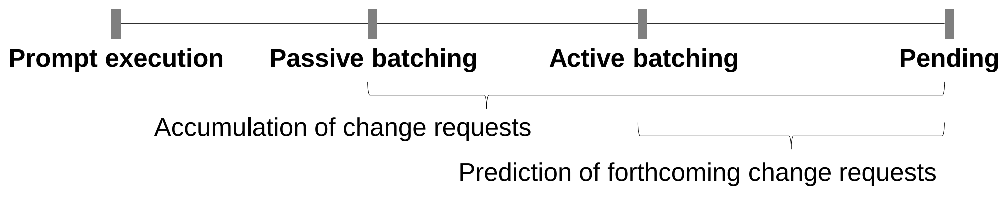
Prompt execution
Passive batching
Active batching
Pending
Accumulation of change requests
Prediction of forthcoming change requests

Figure 1. The comparison of change management methods

3 ACTIVE BATCHING

The present study proposes active batching which determines the set of uncertain upcoming change requests as the batch. Both active batching and passive batching pile up change requests. Whereas passive batching waits for uncertain change requests for the fixed time interval or the given batch size, active batching composes the batch based on the anticipation of probable forthcoming change requests. Although pending makes a decision with the prediction of change propagation, it postpones the execution until all change sources are inactivated. The simple project consisting of three activities in Figure 2 illustrates how active batching works. In Figure 2, activity *k* receives the change request from activity *i* at t_i. When the change request arrives, activity *k* needs to determine the batch scope since another change request from activity *j* might be transferred at t_j. If activity *k* determines to compose a batch including the change request from activity *j*, it waits until t_j, when change propagation becomes certain.

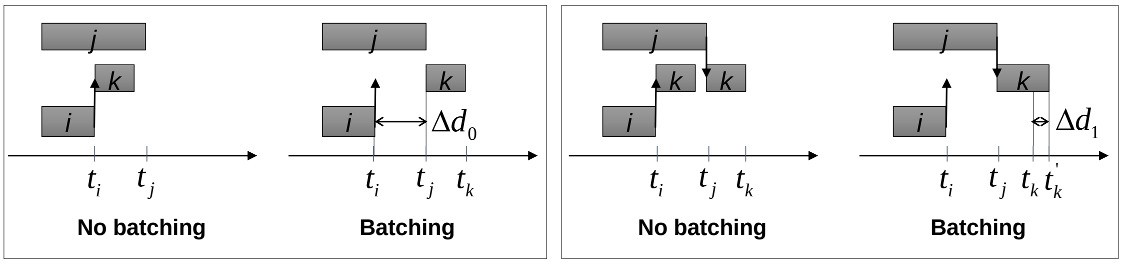

Figure 2. The impact of batching decision on development process

Like other change management methods, active batching saves cost but prolongs project duration in general cases. Active batching saves cost only when the change from activity *j* is propagated by averting redundant setup and execution of activity *k*. The cost of further propagation from activity *k* is also reduced. The amount of expected cost saving is assessed as

$$E[\Delta C] = p(c_k = 1 | c_j = 1) \times (\Delta sc + \Delta ec), \tag{1}$$

where $p(c_k = 1 | c_j = 1)$ denotes the probability of change occurrence of activity *k*, $c_k = 1$, under the condition of change occurrence of activity *j*, $c_j = 1$. Δsc and Δec indicate the amount of expected cost saving for setup and execution, respectively. When the change is not transferred from activity *j*, the end of activity *k* is delayed as the amount of postponement, Δd_0. In the case of occurrence, activity *k* under active batching finishes late as the amount of Δd_1 since it conducts the larger modification job from two change sources. When the change request from activity *j* to activity *k* arrives earlier than the end of activity *k*, activity *k* is congested with multiple change requests. In the case of congestion, active batching reduces project duration, which results in the minus value of Δd_1. The expected delayed duration due to active batching is assessed as

$$E[\Delta D] = p(c_k = 1 | c_j = 1) \times \Delta d_0 + p(c_k = 0 | c_j = 1) \times \Delta d_1. \tag{2}$$

The active batching decision includes the tradeoff between duration and cost of the project. Aggressive batching decision saves cost but prolongs duration. The present study proposes the coefficient of relative value on duration, β, as the criteria of active batching decision. If Inequality (3) is satisfied, the upcoming change is included in the batch since the expected value of cost saving is higher than that of activity delay.

$$\beta \times E[\Delta D] < E[\Delta C]. \tag{3}$$

As β increases, an upcoming change request is unlikely to be included in the batch since the project manager puts emphasis on short project duration rather than low development cost.

Active batching decision is made in three steps as Figure 3. In the figure, c_j^x denotes the *x*th change job of activity *j*. When a change request arrives at $t(b_i(1))$, activity *i* determines the scope of a batch. To forecast upcoming change requests, *definite changes* are searched as the sources of change propagation to activity *i*. Definite changes of the focal activity are defined as the occurred changes of precedent activities which can propagate changes to the focal activity. Intermediate changes propagated from definite changes are identified with information including arrival time, occurrence probability and impact. Finally, the batch set is determined among batch candidates, forthcoming change requests, by considering tradeoff between development delay and cost saving. $B_i(m)$ denotes activity *i*'s batch set consisting of *m* batch candidates.

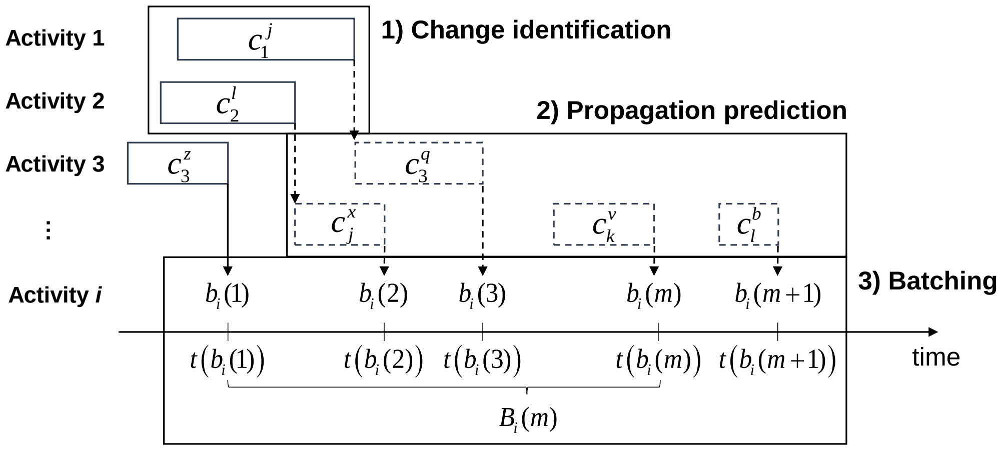

Figure 3. Active batching decision

Two models are provided to support active batching decision. The first one is propagation prediction model which estimates the essential information of upcoming requests. Since change propagation is complex, propagation paths from sources to batch candidates are difficult to be tracked. The second one is batching assessment model assessing the impact of batching decision in terms of duration and cost. Since multiple change requests are transferred to an activity, the sophisticated model is needed to assess the combined change impact from multiple sources of sub-systems.

The present study utilizes pair-wise activity dependency information for establishing two aforementioned change models. Design structure matrix is widely used for representing dependency information between development activities (Browning, 2015). Design structure matrix contains change impact and likelihood between the each pair of activities as Figure 4. In Figure 4, *CL(i,j)* refers to the probability that the change of activity *j* causes the change of activity *i*. If the change is occurred, activity *i* needs to conduct modification work as the portion of *CI(i,j)* to the amount of nominal work. The simplest method to obtain impact and likelihood is to interview experts (Browning and Eppinger, 2002; Clarkson et al., 2004). If the company has sufficient information of change orders in product data management system, dependency information can be gathered from the system (Giffin et al., 2009; Gokpinar et al., 2010).

	i	*j*	
i		0.6	$CL(i,j)$
j	0.4		

Change likelihood

	i	*j*	
i		0.3	$CI(i,j)$
j	0.7		

Change impact

Figure 4. Design structure matrix

3.1 Propagation prediction model

Propagation prediction model tracks change propagation from a change source, a definite change, to a batch candidate. Since change propagation is complex, a change might be propagated via multiple intermediate changes until a batch candidate. Figure 5 illustrates the example of change propagation from the definite change to a batch candidate. The *x*th change of activity *s*, c_s^x, is the change source. The change might be propagated to an intermediate change, c_n^y, and finally to a batch candidate, $b_i(m)$. $b_i(m)$ denotes the *m*th batch candidate of activity *i*.

$$\begin{tikzcd}\boxed{c_s^x} \arrow[r, "CL(n,s)", "CI(n,s)"'] & \boxed{c_n^y} \arrow[r, "CL(n,s)\times CL(i,n)", "CI(n,s)\times CI(i,n)"'] & \boxed{b_i(m)}\end{tikzcd}$$

Figure 5. Propagation prediction

The likelihood and impact of a batch candidate are estimated as the multiplication of likelihoods and impacts along the change path. Since change propagation between c_s^x to c_n^y and c_n^y to $b_i(m)$ are independent, the realization likelihood of $b_i(m)$ is the product of *CL(n,s)* and *CL(i,n)*. Therefore, realization likelihood of a batch candidate $b_i(m)$ with the given change source c_s^x can be assessed as

$$p\left(b_i(m)\right)=\prod_{l=1}^{q-1} CL(a_{l+1},a_l), \quad (a_1=s, a_q=i), \tag{4}$$

where a_l denotes the activity number of the *l*th change along the change path and *q* indicates the total number of changes in the path. The first changed activity is the source s and the final one is the batch activity *i*. The impact of change is mitigated along the propagation path since the amount of change job is proportional to the impact of direct precedent change job in propagation (Kang and Hong, 2009). If the precedent activity n modified its design as the portion of *CI(n,s)*, its successive activity *i* needs to execute the design job as the portion of *CI(n,s)×CI(i,n)*. The amount of a change job of activity *i* is proportional to the changed portion of precedent activity n as *CI(n,s)*. In this manner, the mitigated impact of a batch candidate is calculated as

$$MI\left(b_i(m)\right)=\prod_{l=1}^{q-1} CI(a_{l+1},a_l), \quad (a_1=s, a_q=i). \tag{5}$$

3.2 Batching assessment model

Batching assessment model estimates the impact and realization likelihood of a batch. The realization likelihood and impact of change propagation from multiple sources to an activity might be inferred from the change history of previous product development projects. However, it takes too much effort and time to infer the information since the amount of data is very large. Also data is stored in different systems, which makes practitioners difficult to access, mine and analyze (Giffin et al., 2009; Gokpinar, et al., 2010). Therefore, the present paper utilizes pair-wise dependency information to estimate the impact and realization likelihood of a batch with propagation prediction model as the alternative of inference from the change history.

The realization likelihood of a batch is assessed with noisy-OR model. Based on the assumption of causal independence among parent causes to a child effect, noisy-OR model estimates the realization likelihood of effect when multiple causes exist (Hackerman and Breese, 1996). Since multiple change sources cause changes to a batch activity in the present study, noisy-OR model is applicable under the assumption of causal independence (Lee and Hong, 2015). In noisy-OR model, the batch is executed when at least one batch candidate is realized. Therefore, realization likelihood of the batch of activity *i* B_i is estimated as

$$RL\left(B_i(m)\right)=1-\prod_{j=1}^{m}\left(1-p\left(b_i(j)\right)\right) \tag{6}$$

where *m* denotes the number of batch candidates.

The expected impact of multiple change sources is obtained by considering all possible propagation cases. Since the realization of each batch candidate is assumed as independent, all cases need to be considered except the all-non-propagation case. The expected impact of a batch is assessed as summating the products of probability and impact for all possible cases, expressed as

$$EI\left(B_i\right)=\sum_{\forall(G,G')}\left(\prod_{b_i(j)\in G} p\left(b_i(j)\right)\right)\left(\prod_{b_i(k)\in G'}\left(1-p\left(b_i(k)\right)\right)\right)I(i,G) \tag{7}$$

G indicates the set of realized batch candidates. The elements in *G'* do not occur. $I(i,G)$ denotes the combined change impact of set *G* on activity *i*.

In estimating the expected impact of a batch, it is important to assess the combined change impact, $I(i,G)$, from multiple change sources. Wynn (2014) proposed information space to logically assess the impact of change caused by multiple precedent activities. The information space of activity i can be represented as Venn diagram in Figure 6. Among the design space of activity i S_i, a certain space $S_{(i,k)}$ is influenced by precedent activity k. When multiple precedent activities cause the change to activity i, the impact is the union of influenced spaces.

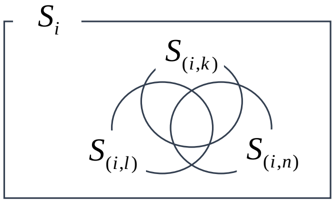

Figure 6. Illustration of design space

The present study proposes the approximation method to assess the combined change impact. The amount of combined change impact possesses the lower and upper bound. When the largest influenced space includes all other influenced spaces, the area of combined change impact is equivalent to the largest influenced space.

$$I_L(i,G) = \frac{\max\limits_{v} \left| S_{(i,v)} \right|}{\left| S_i \right|} \tag{8}$$

v indicates a precedent change of a batch candidate in the batch set. $\left| S_i \right|$ denotes the size of information space of activity i. For example, the size of a design activity can be estimated as the number of design parameters determined by the activity. On the other hand, the size of combined impact area is the summation of all influenced spaces when all influenced spaces are disjoint.

$$I_U(i,G) = \max\left[\frac{\sum\limits_{v} \left| S_{(i,v)} \right|}{\left| S_i \right|}, 1 \right] \tag{9}$$

The combined change impact ranges from the lower bound and the upper bound as

$$I(i,G) = I_U(i,G) - \alpha\left(I_U(i,G) - I_L(i,G)\right), \alpha \in [0,1] \tag{10}$$

where α denotes the coefficient of saved redundant execution. As the coefficient increases, the combined impact decreases to the lower bound.

4 NUMERICAL STUDY

In this chapter, active batching is compared to existing project management methods, prompt execution, passive management and pension. Management methods are implemented on software development project investigated by Fu et al. (2012). The project consists of 12 sub-systems and corresponding 12 activities under one-to-one mapping between a module and a development activity. The information of change propagation, change likelihood and change impact, is represented in Figure 7. The cost and duration of each activity is illustrated as Table 1. The portion of setup duration and cost are assumed as 0.2 for all activities. For example, setup takes 1 day for design activity whose nominal duration is 5 days.

Table 1. Cost and duration of design activities (Fu et al., 2012)

Activity number	1	2	3	4	5	6	7	8	9	10	11	12
Cost	8	9	6	11	45	9	8	20	5	14	28	9
Duration	8	10	4	8	15	8	10	16	5	10	15	8

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0.5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.6	0.5	0	0	0	0	0	0
3	0	0	0	0.8	0	0	0.6	0.3	0	0.3	0	0
4	0	0	0.6	0	0	0	0.2	0.8	0	0	0	0
5	0	0.6	0	0	0	0.4	0	0	0	0	0	0
6	0	0	0.4	0	0	0	0.5	0	0.8	0.4	0	0
7	0	0	0.3	0	0	0	0	0.2	0	0	0.7	0.3
8	0	0	0.2	0	0	0	0.3	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0.2	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0.2	0	0.7
12	0	0	0	0	0	0	0	0	0	0	0	0

Change likelihood

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0.5	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0.5	0.5	0	0	0	0	0	0
3	0	0	0	0.8	0	0	0.4	0.4	0	0.2	0	0
4	0	0	0.4	0	0	0	0.1	0.2	0	0	0	0
5	0	0.5	0	0	0	0.5	0	0	0	0	0	0
6	0	0	0.1	0	0	0	0.2	0	0.7	0.3	0	0
7	0	0	0.2	0	0	0	0	0.4	0	0	0.8	0.4
8	0	0	0.3	0	0	0	0.3	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0.3	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0.2	0	0.6
12	0	0	0	0	0	0	0	0	0	0	0	0

Change impact

Figure 7. Change information of software development project (Fu et al., 2012)

For the given modified module set of incremental designs, the cost and duration of different management methods are compared. Figure 8 illustrates development process of incremental design with different management methods. The changed modules and corresponding activities are set as {2, 3, 5, 6, 8, 10}. Prompt execution provides the shortest development duration. However, it results in the highest cost since the amount of design work is the largest. Design jobs relentlessly continue since change propagation is not controlled. Pension uses the smallest development resource, whereas it results in the longest project duration. The results of passive batching reside between prompt execution and pension by accumulating engineering changes. For example, activity 3 executes redesign at time 10 by accumulating changes from activity 10 and 4 since batch interval is set as 5, $i = 5$. Active batching determines the commencement of design based on the update of change propagation information. For example, activity 6 begins at the end of activity 10. Originally, batch set was supposed to be composed of change requests from activity 3 and 10. Since change request from activity 10 is not realized, activity 6 begins its work by adopting the change request from activity 3.

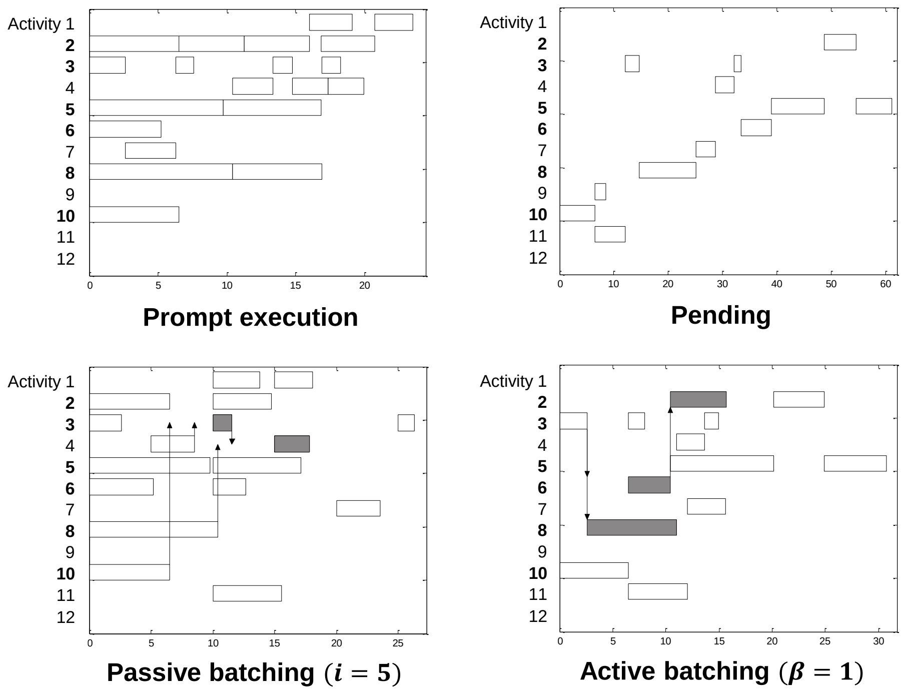

Figure 8. Development process with different change management methods

Active batching moves the Pareto frontier line of passive batching into the direction of less cost and short duration. Figure 9 plots the averaged results of development projects. Since change propagation occurs stochastically, the averaged value is earned by Monte Carlo simulation. For each management method, simulation runs 10,000 times. For the same management method, the results depend on managerial coefficients. For active batching, low importance on the coefficient of value on development duration β increases cost and decreases duration. Wide batch interval of passive batching *i* leads to similar results. However, some managerial coefficient results in dominated performance as Figure 9.

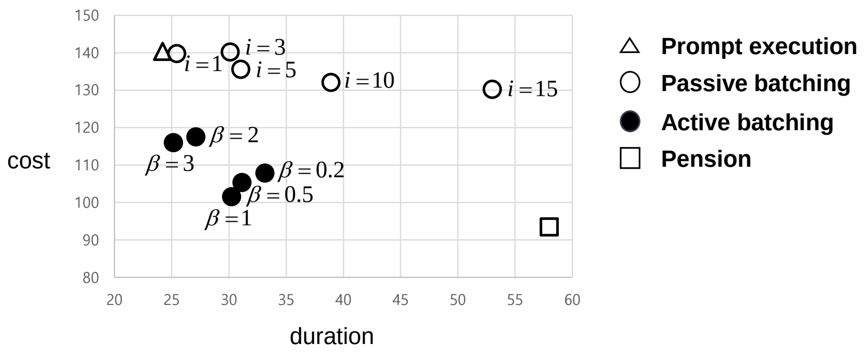

Figure 9. Project performance with different change management methods

5 CONCLUSION

The present paper presents the risk management method for incremental design. Incremental design is suitable for companies releasing products periodically since incremental design maintains product architecture and most modules. Companies are able to develop a new-generation product within short duration. Although incremental design reduces development burden, modification and addition of modules are required to enhance product utility. The change of modules in complex product leads propagation of engineering changes in development process. Change propagation in development process generates additional change jobs, which results in the prolonged duration and increasing cost. However, the prediction of change propagation is the overwhelming task since changes are propagated stochastically in the complex manner. Therefore, change management in incremental design has been the critical issue. The present paper proposes the novel change management method, active batching. The existing change management method, passive batching, acknowledges the existence of multiple change requests on the activity. It controls propagation by batching change requests with the criteria as the given time interval or the given batch size. Active batching determines the scope of batch based on the prediction of probable upcoming change requests. In the theoretic perspective, two models named divergence model and convergence model were developed to track change propagation and assess the impact of batching decision. Practically, active batching provides better performance than passive batching in terms of both duration and cost. Project managers are able to take the advantage of flexible decision with updated propagation information.

REFERENCES

Ahmad, N., Wynn, D. C. and Clarkson, P. J. (2010), “The impact of packaging interdependent change requests on project lead time”, *The 12th International DSM Conference*, Cambridge, UK, pp. 293-298.

Browning, T. R. and Eppinger, S. D. (2002), “Modeling impacts of process architecture on cost and schedule risk in product development”, *Engineering Management, IEEE Transactions on*, Vol. 49, No. 4, pp. 428-442.

Browning, T. R. (2015), “Design structure matrix extensions and innovations: A survey and new opportunities”, *Engineering Management, IEEE Transactions on*, Vol. 63, No. 1, pp. 27-52.

Clarkson, P. J., Simons, C. and Eckert, C. (2004), “Predicting change propagation in complex design”, *Journal of Mechanical Design*, Vol. 126 No. 5, pp. 788-797.

Du, J, El-Gafy, M. and Zhao, D. (2015), “Optimization of change order management process with object-oriented discrete event simulation: Case study”, *Journal of Construction Engineering and Management*, p. 05015018.

Eckert, C., Clarkson, P. J., and Zanker, W. (2004), “Change and customization in complex engineering domains”, *Research in Engineering Design*, Vol. 15 No. 1, pp. 1-21.

Eckert, C., Wyatt, D. and Clarkson, P. J. (2009), “The elusive act of synthesis: Creativity in the conceptual design of complex engineering products”, *The 7th ACM Conference on Creativity and cognition*, Berkeley, USA, pp. 265-274.

Fu, Y., Li, M and Chen, F. (2012), "Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix", *International Journal of Project Management*, Vol. 30 No. 3, pp. 363-373.

Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C. and Clarkson, P. J. (2009), "Change propagation analysis in complex technical systems", *Journal of Mechanical Design*, Vol. 131 No.8, p. 081001.

Gokpinar, B., Hopp, W. J. and Iravani, S. M. R. (2010), "The impact of misalignment of organizational structure and product architecture on quality in complex product development", *Management Science,* Vol. 56 No. 3, pp. 468-484.

Ha, A. Y. and Porteus, E. L. (1995), "Optimal timing of a reviews in concurrent design for manufacturability", *Management Science*, Vol. 41 No. 9, pp. 1431-1447.

Hackerman, D. and Breese, J. S. (1996), "Causal independence for probability assessment and inference using Bayesian networks", *System, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, Vol. 26 No. 6, pp. 826-831.

Kang, C. M. and Hong, Y. S. (2009), "Evaluation of acceleration effect of dynamic sequencing of design process in a multiproject environment", *Journal of Mechanical Design*, Vol. 131 No. 2, p. 021008.

Koh, E. C. Y., Keller, R., Eckert, C. M. and Clarkson, P. J. (2009), "Change propagation modelling to support the selection of solutions in incremental change", *The 2nd International Conference on Research into Design,* Bangalore, India, pp. 199-206.

Lee, J. and Hong, Y. S. (2015), "Design freeze sequencing using Bayesian network framework", *Industrial Management & Data Systems*, Vol. 115 No. 7, pp. 1204-1224.

Loch, C. H. and Terwiesch, C. (1999), "Accelerating the process of engineering change orders: Capacity and congestion effects", *Journal of Product Innovation Management*, Vol. 16 No. 2, pp. 145-159.

Maier, C., Browning, T. R., Yassine, A. A. and Walter, U. (2015), "The cost of speed: Work policies for crashing and overlapping in product development projects", *Engineering Management, IEEE Transactions on*, Vol. 62 No. 2, pp. 237-255.

McMahon, C. A. (1994), "Observations on modes of incremental change in design", *Journal of Engineering Design*, Vol. 5 No. 3, pp. 195-209.

Nadia, B., Gregory, G. and Vince, T. (2006), "Engineering change request management in a new product development process", *European Journal of Innovation Management*, Vol. 9 No. 1, pp. 5-19.

Ollinger, G. A. and Stahovich, T. F. (2004), "RedesignIT – A model-based tool for managing design changes", *Journal of Mechanical Design*, Vol. 126 No. 2, pp. 208-216.

Terwiesch, C. and Loch, C. H. (1999), "Managing the process of engineering change orders: The case of the climate control system in automobile development", *Journal of Product Innovation Management*, Vol. 16 No. 2, pp. 160-172.

Wyatt, D. F., Eckert, C. M. and Clarkson, P. J. (2009), "Design of product architecture in incrementally developed complex products", *ICED 09,* California, USA.

Wynn, D. C., Caldwell, N. H. M. and Clarkson, P. J. (2014), "Predicting change propagation in complex design workflows", *Journal of Mechanical Design*, Vol. 136 No. 8, p. 081009.