A framework for managing engineering change propagation

Krishna Reddi
Syracuse University

Young Moon
Syracuse University, ybmoon@syr.edu

A framework for managing engineering change propagation

Krishna R. Reddi and Young B. Moon*

Department of Mechanical and Aerospace Engineering
Institute for Manufacturing Enterprises
Syracuse University
149 Link Hall, USA
E-mail: krreddi@syr.edu
E-mail: ybmoon@syr.edu
*Corresponding author

Abstract: Engineering Changes (ECs) are facts of life for any company developing and introducing new products, despite a commonly held notion that they are distractions from normal operation. Companies can become more innovative by utilising ideas from the ECs or by learning how to handle the ECs. This paper presents a framework to manage the ECs effectively, particularly the issue of EC propagation. An EC seldom confines itself to a single change, but triggers other changes in different components. The framework is designed to identify the affected components automatically, capture the required knowledge during the design phase of the product life cycle, and use it during the Engineering Change Management (ECM) process.

Keywords: engineering change management; ECM; change propagation; innovation; learning; object-oriented concepts.

Reference to this paper should be made as follows: Reddi, K.R. and Moon, Y.B. (2009) ‘A framework for managing engineering change propagation’, *Int. J. Innovation and Learning*, Vol. 6, No. 5, pp.461–476.

Biographical notes: Krishna R. Reddi is a doctoral student in the Department of Mechanical and Aerospace Engineering at Syracuse University, USA. He holds a Bachelor of Engineering from Andhra University, India and a Master’s in Engineering in Mechanical Engineering with Specialization in CAD/CAM from Osmania University, India. His research interests include agent-based modelling, engineering change management, product life cycle management and enterprise resources planning.

Young B. Moon is on the Faculty of Mechanical and Aerospace Engineering at Syracuse University, USA. He is the Director of the Institute for Manufacturing Enterprises. He holds a PhD degree from Purdue University, USA and is a Professional Engineer (PE), a Certified Fellow in Production and Inventory Management (CFPIM) and a Certified Manufacturing Engineer (CMfgE). His professional interests include enterprise systems, product realisation processes and systems, product life cycle management, and modelling and simulation.

1 Introduction

Competition is driving companies for better quality and performance of their products. As a result, companies are constantly striving to improve their products with greater and continuous involvement of the customers. Engineering Changes (ECs) are necessary to accommodate any customer's requests or improve the products even after the product is released to the market. According to Cooper, the key success and profitability factors for new product development are the companies' capacity to differentiate their products and solve customer problems (Cooper and Edgett, 2008):

> "Developing and delivering new products that are differentiated, solve major customer problems, and offer a compelling value proposition to the customer or user is the number one key to New Product Development success and profitability."

An EC can be considered as "an alteration in the approved configuration of a product related item" (US Military Standard 480B, 1988). An item can be a document or a physical component of the product structure (Riviere *et al.*, 2002). ECs occur in many forms including dimensions, fits, forms, functions and materials of products or their components. After the product design is finalised and released for production (Huang and Mak, 1999; Huang *et al.*, 2003), these changes may be aimed at:

- solving a problem that was not detected in the product development process
- enhancing the functionality of the product
- improving the performance of the product.

Whatever may be the reason, any EC should help to improve a company's competitiveness in the marketplace.

An EC is said to be a necessary evil. It is necessary in the sense that it increases the quality and performance of the product, while it is evil in the sense that it costs the company in terms of extra expense and time. An EC helps in improving the level of customer satisfaction with the product, since many of these changes are initiated by the customer as new requirements. The companies also initiate modified specifications or manufacturing changes as they see that their customers are better served with the EC (Bhuiyan *et al.*, 2006). It is also introduced to increase the functionality of the product in the latter stages of the product life cycle or to replace an obsolete technology. While these are the desirable effects of an EC, ECs also introduce undesirable effects such as longer product leadtime, as well as extra cost in terms of personnel and material scrap. In addition, ECs introduce numerous document changes to the product data, which require a significant effort to handle.

Properly planned and managed ECs are great assets to any organisation as they enable the organisation to match the technological innovation of competitors and, thus, maintain a competitive advantage. Poor management of ECs leads to poor performance of a corporation due to expensive or unnecessary purchases, high scrap expenses, production delays, loss of market share, slow market responsiveness, *etc.* (Diprima, 1982). Optimal management of ECs can be achieved by balancing between efficiency and effectiveness of a company's Engineering Change Management (ECM) procedure. Companies that emphasise more on effectiveness tend to incorporate numerous checkpoints to ensure all the tasks are done in a proper manner, while the others that emphasise more on efficiency keep the ECM tasks to a minimum level or adopt Enterprise System solutions (Moon, 2007) so that the ECs can be processed rapidly (Huang and Mak, 1999).

ECM is a complex process. The complexity in implementing an EC can be understood from the fact that it generally demands involvement of more than one functional area or even a whole organisation. In order to address the complexity, companies need to adopt a well-defined procedure to manage an EC, starting with their EC proposal to ultimate implementation. Typically, ECM committees are formed to deal with problems incurred with ECs. An EC coordinator is appointed to manage and coordinate all the EC-related activities throughout its life cycle.

An important issue in managing an EC is to identify how other components are affected by the EC. The dependencies of different components of a given product may be direct or indirect with respect to an EC. When a component is affected by a change in another component that is directly caused by the EC, the dependency is called indirect. The dependency relationships can become too complex to be handled by a person with single technical perspective. It requires multidisciplinary knowledge to ensure identification of all the dependencies between the components of the product. In order to capture and document all the necessary dependencies, numerous people from different disciplines should be involved from the design phase. Such an involvement makes the identification of affected components complete during EC processing. Identification process can be automated by capturing the dependencies in a retrievable format (Browning, 2001).

This paper presents a framework to automate the identification of affected parts due to an EC and to document how the components are affected. It lists all the components that are affected by an EC along with their features. The attribute-component and component-component relationships are captured during the design phase and are used to identify the affected parts. When products are complex, it is almost impossible to identify all the dependencies after the product is designed, manufactured and released into the market. The presented framework adopts concurrent engineering concept, which is well understood and accepted methodology in designing products in industry. It is easier and practical to capture all the possible dependencies between the attributes and the components within a product during the design phase. Once the dependency data are captured and stored in a retrievable format, the identification process can be automated.

The rest of the paper is structured as follows. This introduction is followed by a literature review on ECM, particularly focusing on the issue of EC propagation. Section 3 presents the developed framework. In Section 4, a case study illustrates how the presented framework is used. The paper ends with conclusions and discussion on future work.

2 Literature review

Relatively little work have been published on ECM as compared to other topics of research in Product Life cycle Management (PLM). In this section, literatures that deal with ECM relevant to the proposed framework are reviewed.

Browning (2001) in his review on the use of Design Structure Matrix (DSM) to system decomposition and integrated problems mentioned DSM as a popular representation and analysis tool for system modelling. A DSM used to display relationship between components of a system in a compact, visual and advantageous format can also provide an indication of how change may propagate through a product. Intelligent system decomposition and integration analysis is facilitated by the use of DSMs. The system here can be a product, process or an organisation.

Eckert *et al.* (2004) based on a study conducted in Westland Helicopters, identified two types of changes: the emergent changes and the initiated changes. This particular study was based on the interviews conducted with the company's employees. The interviewed designers commented that they typically expected up to four follow up changes arising from each initiating change. The key to successful change management lies therefore in understanding the state of design and the connectivity between parts of a design. The source of change, interdependencies between parts and systems, types of propagation behaviour, consequences of change on product quality, cost, and time to market, and the state of tolerance margins on key parameters need to be taken under consideration for successful change management. They proved that by capturing the design knowledge and experience, in the form of experienced designers in the company, an automatic tool to identify the EC propagation can be developed. This work has further led to the development of a computer support tool by Clarkson *et al.* (2004) to identify the risk of a change.

Clarkson *et al.* (2004) reported an analysis of change propagation based on the case study in Westland Helicopters. They also developed mathematical models based on likelihood, impact and risk DSMs, to predict the risk of change propagation in terms of likelihood and impact of change. This paper concentrated on capturing past experience about the propagation of change between systems in terms of the likelihood of their occurrence and the impact such changes would have. The chief managers during the interviews mentioned that designers frequently failed to realise how their work will influence the others, change flow occurred resulting in changes up to four components/systems from the initial change, and unexpected changes ranged from 5% to 50%.

Yang *et al.* (2005) and Rouibah and Caskey (2003) introduced a concept to manage ECs and engineering workflow, utilising a parametric network. Here, the word 'parameter' refers to a critical attribute such as the product performance, geometry, *etc.* This parameter-based approach links an engineering workflow to product data, and so can propagate ECs and engineering tasks across company borders. The parameters are identified and assigned to a particular company in a network of companies to manage and respond to the changes in more efficient manner. Though this method is effective for less complex products the number of parameters increases drastically and so does the complexity of the network. And the parameters selection is another cumbersome task in case of a complex product.

Keller *et al.* (2005) proposed a Change Prediction Method (CPM) tool to visualise the change propagation and how multiple views are used in the context. The tool uses enhanced information visualisation techniques such as multiple views and fisheye techniques for displaying the desired information in the context of change management. The CPM gave a good indication of future change likelihood without the need for detailed knowledge of the product development process. The resulting visualisations due to the tool offered new possibilities for designers in industry to analyse and view change propagation data so that they are not overwhelmed by the complexity of the component interactions.

Bouikni *et al.* (2006) proposed a model aimed at controlling the information flow needed to support a product definition evolution while ensuring its validation by all the involved disciplines. This model can be applied both to the product design and to the modification phases of the product life cycle. A product feature-discipline relationship table was used to identify the disciplines affected by a particular feature. An estimator of change predicts the type of impact detrimental or beneficial on every discipline due to a change in a particular discipline. The affected disciplines are identified from the shared product feature table. This model identifies the affected disciplines due to a change in a feature but does not identify the affected components or sub-assemblies that are affected.

Do *et al.* (2008) proposed a product data model that supported product data views and ECs. This model consisted of base product configurations, assembly structures, product data views and EC history. The model maintained consistency between the base product definition, product data views and ECs. A change propagation procedure is proposed based on the defined product model. The proposed propagation procedure consisted of three phases:

1 identification of changed products in the base product definition

2 the retrieval of corresponding parts in product data views

3 the product data views are changed according to the change history of the product structures in the base product definition.

The application of this propagation model is limited since it is based on the proposed product data model and not on a standard product data format.

Lee *et al.* (2006) proposed a model based on a case study of the ECM practices in a Korean automobile company. The developed model provided a basis for the integration of informal and unstructured offline collaboration with structured online workflows. The collaboration model demonstrates how semantic web technology can represent and share various types of EC-related knowledge in context. The Case-Based Reasoning (CBR) technique is used along with the concept-based similarity measure so that manual efforts in managing the cases could be minimised. The authors stressed knowledge accumulation and retrieval, and explained these with some examples of the prototype system.

Aurich and Martin (2007) developed a change impact matrix, derived from the Virtual Reality (VR) analysis, which captured the relationships between various production elements. The impact of a particular EC is estimated with the help of VR on the elements of the production system. The ECs are grouped based on the impact analysis and defined as the EC projects (a group of ECs which are selected to be implemented simultaneously).

Cohen *et al.* (2000) proposed a data representation model that facilitated change and change propagation in design representation of engineering products. They proposed a methodology ‘C-FAR’ to extract information from STEP data format, a recognised standard of design data representation. The product was broken down into elements which were later considered as attributes. A matrix called the C-FAR matrix, comprising of linkage values, linked one attribute of an entity to one attribute of another entity. This model is appropriate for small and relatively simple products due to its computational complexity.

Propagation of change is an essential task in developing redesign plans of a product. Ollinger and Stahovich (2004) stated that, "If a change is applied to the correct quantity, that change will propagate through the device, and ultimately modify the target quantity in the desired way." They developed a program 'redesignIT' that generated proposals for achieving redesign goals, identified side effects, potential or certain, and suggested additional changes to counteract those effects. The model used, consisted primarily of the relevant physical quantities, and the causal relationships between them. An important characteristic of the program is that it is designed to generate redesign plans for which undesirable effects are kept as small as possible. For each redesign plan, the program reports the degree to which the plan can be expected to achieve the redesign goal, as well as the nature and severity of the side effects. This allows the program to rank the proposed plans, thus enabling the designer to quickly identify the best one. This program aims at a specific redesign goal, during the redesign of a specific function, limiting its applicability.

Rutka *et al.* (2006) developed a model to support the decision-making process in ECM. The model captured the knowledge about dependencies between the various systems of a product and identified the impact and risk of an EC. Though the model captured the dependencies between the systems it ignored the dependencies between the attributes of the product and the systems (that are part of the product). It required more human intervention when the change was not defined in detail or was defined in terms of an attribute of the product. The main reason for this was that, the model ignored the attribute-component dependency. The model was intended to manage a change during the design phase also, which made it more complex with the presence of more attributes to define a change. This is because the design is not frozen in the design stage and may change due to any reason at any time.

3 Methodology

Though there are some research works (Clarkson *et al.*, 2004; Rutka *et al.*, 2006) addressing the propagation of change, it was a part of a work that emphasised on how to estimate impacts of propagation and risk of the ECs. Mathematical models were developed to express the impact of an EC in terms of time and cost. In this paper we address the propagation of change with the focus on identifying all the components of a product that are affected by an EC along with the way each of the components is affected.

While a few others (Eckert *et al.*, 2004; Keller *et al.*, 2005; Rutka *et al.*, 2006) have tried to address the propagation of the change during all the phases of the entire product life cycle, the framework presented in this paper addresses the changes only after the design is finalised. Considering an EC in product data during the design phase is very complicated because the product data is not fixed and is subjected to change anytime. Identifying the affected parts after the design is finalised can be simpler because the product data is determined by then. The mechanics of change propagation can be better understood by concentrating on the ECs after the design is finalised. Empirically this also makes sense because there are a large number of EC Requests made immediately after the product is released to the market according to Bhuiyan *et al.* (2006). They also reported that the number of ECs is few during its lifetime, but increases again towards the end of the product life. Considering the ECs after the design is finalised requires relatively less amount of information to define a change in product data and to identify the components affected by the change.

DSMs (Browning, 2001; Clarkson *et al.*, 2004; Eckert *et al.*, 2004; Rutka *et al.*, 2006) and other tables (Bouikni *et al.*, 2006; Aurich and Martin, 2007) have been used to capture the component-component, component-attribute dependencies but their use was limited to parts with a few number of components or dependencies. Though some papers (Clarkson *et al.*, 2004; Eckert *et al.*, 2004; Aurich and Martin, 2007; Rutka *et al.*, 2006) addressed the ECs in complex parts, the models presented require significant human involvement. This is because when the parts have more components affecting each other, nested DSM tables need to be developed to capture all these diverse dependencies. This also leads to a considerable increase in the number and dimensions of the DSM. When the matrix dimensions increase or the number of matrices increase, the retrieval of dependency information from the DSMs becomes difficult to automate.

Object-oriented concepts can improve the identification of propagation and the subsequent retrievals, yet no prior works have used the concepts. Use of object-oriented concepts along with a good database management can help to systematically identify all the components affected by a specific change in a component. This type of a framework can perform well for complex parts without sacrificing the performance and completeness of the parts. The use of object-oriented concepts also makes it easier and efficient to retrieve information from a database storing the dependency information.

Engineers involved in the design of the product from all the disciplines are provided with the framework that helps them to capture the dependencies between components during the design phase. Depending on the discipline and background of the engineer, the views of the dependency between the components differ, hence most of, if not all, the possible dependencies can be captured. The proposed framework comprises of two user interfaces and a database. The data is entered into the database via a user-friendly interface during a design phase and is retrieved during the ECM later. The User interface titled 'Components Dependency' is used to save the dependencies into the database, while the user interface titled 'EC Propagation' is used to retrieve the data from the database. 'EC Propagation' also lists the affected parts due to a particular Type of Change (TOC) in the product or its component.

The framework uses two classes: (1) EC class and (2) Propagation class.

3.1. EC class

EC classes capture the relationships between the various components and attributes of the product during the design phase. This information is retrieved to establish the components affected by an EC.

Each EC class is defined by the following attributes:

- Initiator
- Target
- TOC
- Likeliness.

3.1.1 Initiator

The Initiator is the component that is known to be affected by the EC at a given point of time. The affected targets are determined based on the initiator. This Initiator may be an attribute that is proposed to be changed or a component identified to be changed. At a particular point in the change propagation, there can be multiple initiators. Each Initiator along with the TOC undergoes identifying the Target and the TOC.

3.1.2 Target

The Target is the component that is identified to be affected in a particular way by a particular TOC in the corresponding Initiator Component. A combination of TOC and the Initiator identifies a corresponding combination of TOC and Target.

3.1.3 Type of change

This defines the TOC the component is going through due to the EC. Such change types can be industry-specific. Depending upon the industry, the framework can be used according to the way the TOC is defined. Some examples of industries which have different types of manufacturing processes, design processes or materials, *etc.*, are the aerospace, air conditioner manufacturing and computer manufacturing industries.

Industry-specific TOCs are also used to represent any possible way a component can be affected due to the change in another component. However, this TOC of the initiator and the target need not be in the same domain. For example, both an electric motor and an elevator need to define the power and type (induction or stepped, *etc.*) as its capacity (maximum allowable load). If there is an EC for the physical size of the vehicle, the change it initiates and propagates to the motor will also affect its electric specifications. A few possible typical entries for the TOC are Material, Shape and Size, *etc.*

3.1.4 Likeliness

This defines the likeliness that a change in the Initiator affects the Target. Depending upon the impact of the change on the Initiator, the change can influence or may not influence the Target by a large extent. The likeliness is indirectly defined by the extent to which the Initiators change. Sometimes the Target is affected when the Initiator undergoes a change of certain magnitude or more. The likeliness that the Initiator undergoes a change of such magnitude defines the likeliness of the Target being affected by a change in the Initiator.

The initiator and target fields can be selected from a list of components imported from the Computer Aided Design (CAD) software. Though this feature is not included in the present framework it can be implemented where the CAD is used to a large extent. The TOC field lists the possible ways in which the initiator can affect the target. This list can include all the types based on history. Likeliness field is chosen from a list of three possibilities: ‘High’, ‘Medium’ and ‘Low’.

3.2 Propagation class

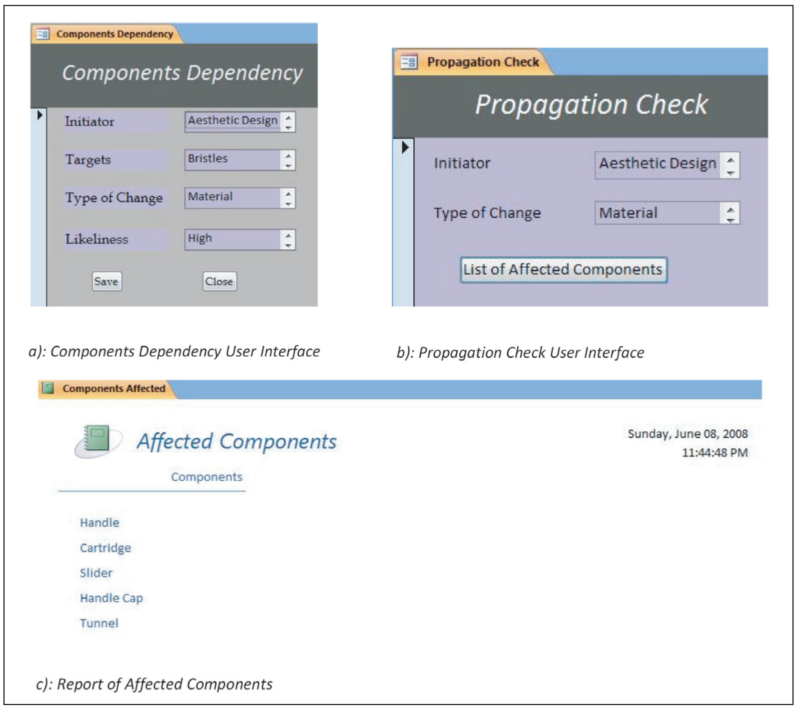
Propagation class is used to create an object that identifies the affected components based on the TOC in the initiator or change in the attribute of the product. Each object in the propagation class has two attributes: (1) Initiator and (2) TOC. The 'initiator' as defined earlier is the component that is to be affected by a change at a particular point of time. The 'type of change' is the way in which the Initiator is affected by a change.

These two attributes combine to identify the components affected by a particular EC. Using the two values given for 'Initiator' and 'Type of Change', the Propagation class searches the database for matching records repeatedly until it finds all the possible affected parts. The values of the Initiator and TOC for a given iteration of comparing the records do not change. Each iteration of the records comparing process has a different set of Initiator and TOC taken together. The logic used for this process is explained in the next section.

3.3 Implementation

As stated earlier the framework has two user interfaces: the first, the 'Components Dependency' interface, and the second, the 'EC Propagation' interface. The 'Components Dependency' interface asks a user to choose a value from a list of possible values for each required field. The user can add a value if not found in the list of possible values. This interface captures and stores the dependency data into the database. It asks the user minimum data that is required to define a dependency for our purpose. The required data includes 'initiator', 'target', 'type of change' and 'likeliness of the change to occur'.

Figure 1 Screenshots of an implementation of the framework (see online version for colours)



Components Dependency

Initiator	Aesthetic Design
Targets	Bristles
Type of Change	Material
Likeliness	High

a): Components Dependency User Interface

Propagation Check

Initiator	Aesthetic Design
Type of Change	Material

b): Propagation Check User Interface

Affected Components

Sunday, June 08, 2008
11:44:48 PM

Components

Handle
Cartridge
Slider
Handle Cap
Tunnel

c): Report of Affected Components

The initiator field lists all the possible components of the product that can affect other components, the target field lists all the components that can be affected by a change in other components, the TOC lists all the ways these targets can be affected, and the likeliness field lists the three possible values of ‘High’, ‘Medium’ and ‘Low’.

The ‘EC Propagation’ asks only two fields to identify the affected parts due to a particular change in a component. The interface lists all the possible values for Initiator and TOC in the same way as stated above. Once the initiator and TOC are selected from the respective lists, clicking the ‘List of Affected Components’ button in the Propagation Check screen opens a report with a list of components affected due to that particular TOC in the initiator. The procedure followed in identifying the affected components is explained in the following section.

3.4 Procedure and logic

Once a component (Initiator) is identified to be affected by the EC at the first level, then the way in which it is affected, determines the TOC. Based on these two ‘Initiator’ and ‘Type of Change’ the propagation is initiated and at the end we come with a list of affected components along with the TOCs they are expected to undergo.

Since the scope includes only EC requests of the products that are already released to the market, a database that defines the relationship between components is constructed and documented by the time an EC request comes in. Once an EC request has been made then the Initiator and the TOC are identified based on the EC request. The initiator and the TOC are input to the propagation class that identifies the various levels of affected components level by level. At each level, the previous level components act as initiator components. A component that is already used as an initiator component is not considered as an initiator for the levels further down. Finally a list of affected components along with the TOC is identified.

3.4.1 Summarisation of the steps

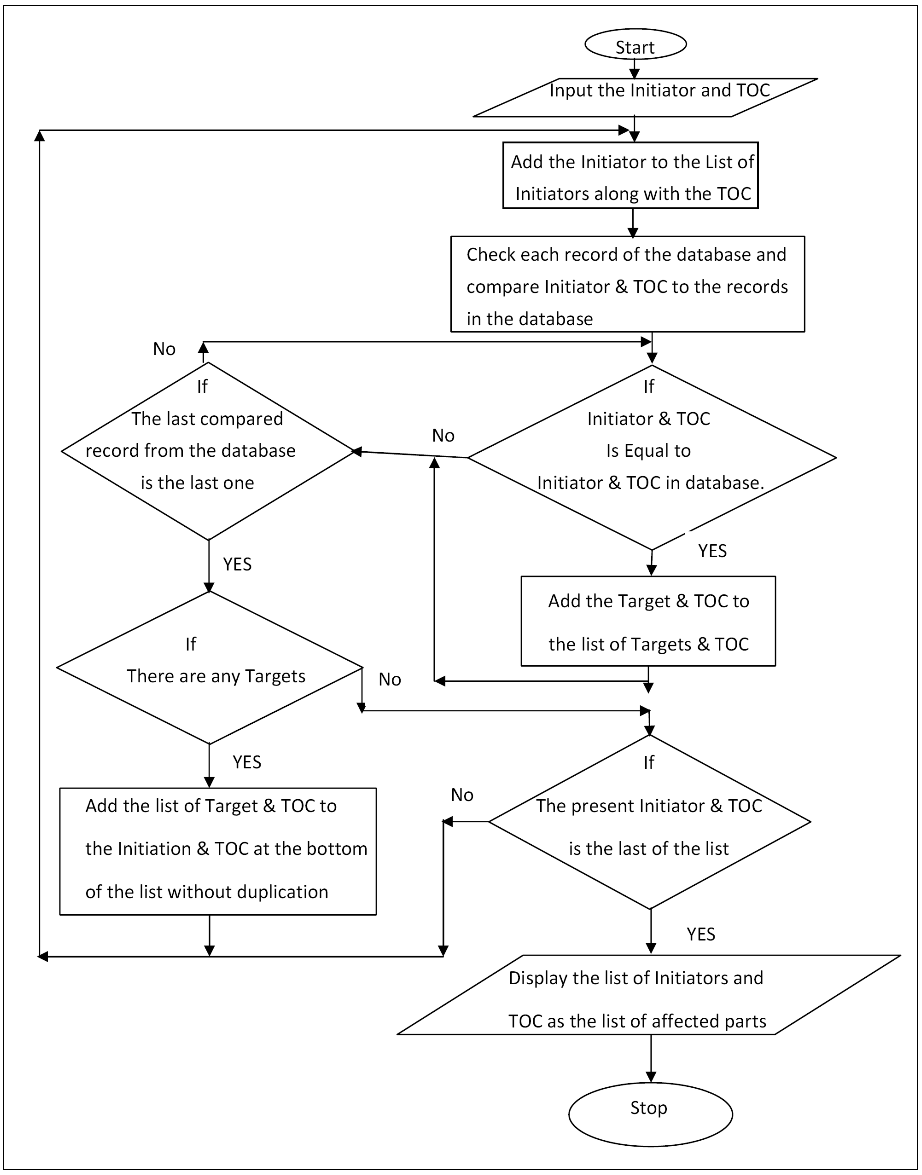
Step 1 The framework is used to capture the dependencies between the components of the product during the design phase of the product life cycle. The data is inputted by the engineers through the Components Dependency interface and saved into the database by the proposed framework.

Step 2 The product is released into the market and the database of dependencies is ready. As an EC proposal comes in, it is analysed and the initiator and the TOC or attribute are identified.

Step 3 The initiator and the TOC are selected from the lists in the Propagation Check screen and the ‘List of Affected Components’ button is pressed. This gives a list of components affected by the proposed TOC in the initiator.

Step 4 All the personnel related to the requested components are notified of the change and asked to review the extent of the impact.

Figure 2 Logic diagram of change propagation


Start
Input the Initiator and TOC
Add the Initiator to the List of Initiators along with the TOC
Check each record of the database and compare Initiator & TOC to the records in the database
No
If The last compared record from the database is the last one
No
If Initiator & TOC Is Equal to Initiator & TOC in database.
YES
YES
Add the Target & TOC to the list of Targets & TOC
If There are any Targets
No
YES
If The present Initiator & TOC is the last of the list
No
Add the list of Target & TOC to the Initiation & TOC at the bottom of the list without duplication
YES
Display the list of Initiators and TOC as the list of affected parts
Stop

4 Case study

4.1 Product description

A new toothbrush with built-in toothpaste has been developed and used here to illustrate the developed framework. The development of the toothbrush has gone through a typical product development process from idea generation to manufacturing. The toothbrush is made up of a Handle, Cartridge, Paste delivery Tube, Slider, Handle cap and Bristles.

The handle has a cylindrical hole along its length irrespective of its external shape. The cartridge is a flexible cylindrical tube that resides inside the cylindrical hole of the handle. A slider that moves along the cylindrical hole in the handle slides along the length of the handle and compresses the flexible tube cartridge, to deliver paste from the cartridge to the bristles through a delivery tube, which extends from the handle connecting the cartridge to the bristles. The paste delivery tube has an interface with the cartridge via the handle. The paste delivery tube can be latched to the handle when not in use. The bristles are inserted along the pattern of holes drilled in the handle. A handle cap covers the cartridge and sliders restricting them to the cylindrical hole in the handle. For better understanding please refer to Figures 3 and 4.

Figure 3 Attribute-component-component dependencies

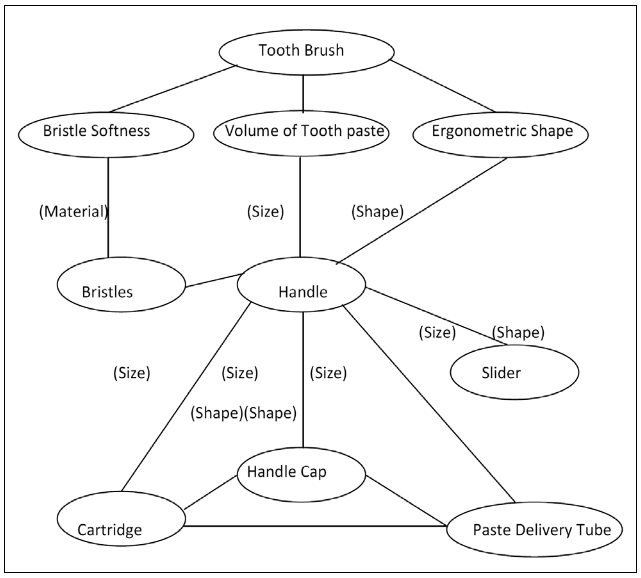
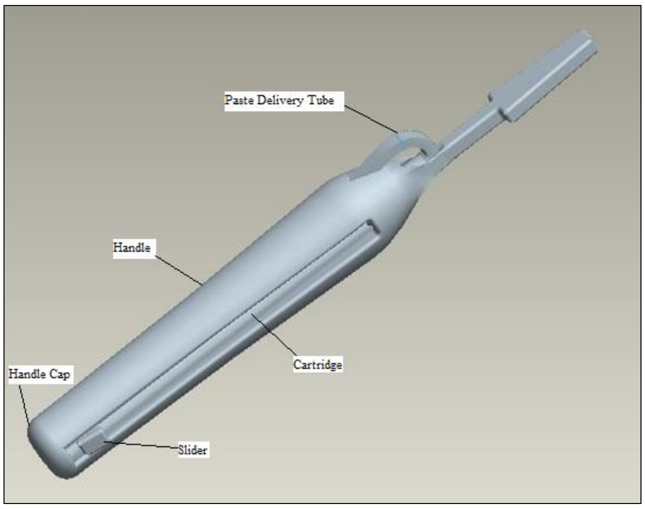


Figure 4 3-D model of the toothbrush (see online version for colours)



The customer needs for this particular toothbrush were collected by interviewing potential customers. These include: (1) sufficient volume of the paste in the cartridge, (2) softness of the bristles, and (3) aesthetic or ergonometric design of the handle. Considering a request for change in the volume of paste the brush can hold; we have implemented the above discussed concept to identify the affected components.

The steps for implemented procedure are as follows:

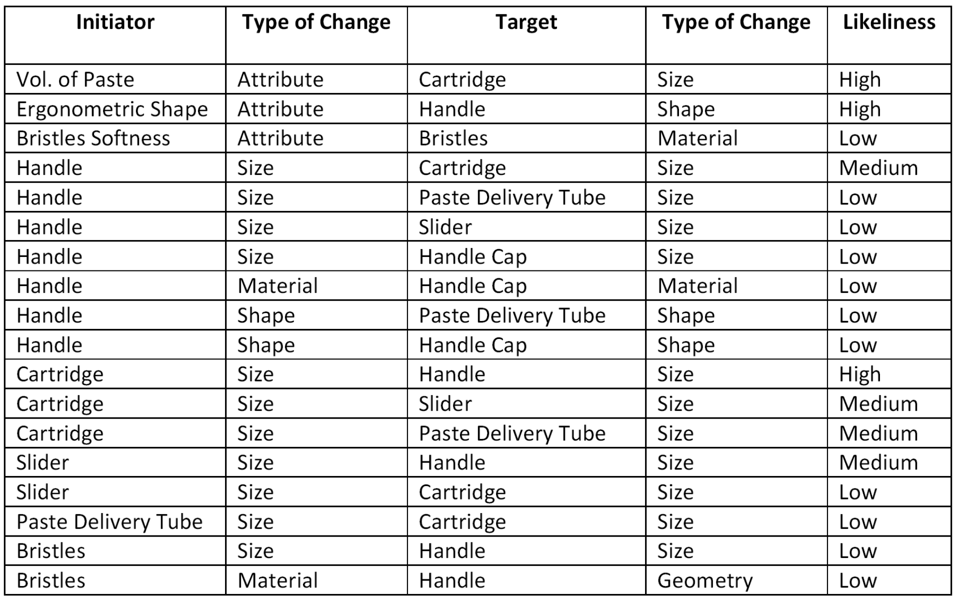
Step 1 The framework has been used to capture the dependencies between the components of the Toothbrush during the design phase. We inputted the data through the Components Dependency interface and saved them into the database by the proposed framework.

Step 2 The database of dependencies like the one in Figure 5 is ready. The EC proposal to change the volume of the paste came in, was analysed and the initiator was identified as 'Vol. of Paste' and the TOC as 'Attribute'.

Step 3 The 'Vol. of Paste' and the 'Attribute' were selected from the lists in the Propagation Check screen and 'List of Affected Components' button was pressed. This gave a list of components affected by the proposed TOC in the initiator as in Figure 1.

Step 4 All the personnel related to the requested components are notified of the change and asked to review the extent of the impact.

Figure 5 An example of dependencies database



Initiator	Type of Change	Target	Type of Change	Likeliness
Vol. of Paste	Attribute	Cartridge	Size	High
Ergonometric Shape	Attribute	Handle	Shape	High
Bristles Softness	Attribute	Bristles	Material	Low
Handle	Size	Cartridge	Size	Medium
Handle	Size	Paste Delivery Tube	Size	Low
Handle	Size	Slider	Size	Low
Handle	Size	Handle Cap	Size	Low
Handle	Material	Handle Cap	Material	Low
Handle	Shape	Paste Delivery Tube	Shape	Low
Handle	Shape	Handle Cap	Shape	Low
Cartridge	Size	Handle	Size	High
Cartridge	Size	Slider	Size	Medium
Cartridge	Size	Paste Delivery Tube	Size	Medium
Slider	Size	Handle	Size	Medium
Slider	Size	Cartridge	Size	Low
Paste Delivery Tube	Size	Cartridge	Size	Low
Bristles	Size	Handle	Size	Low
Bristles	Material	Handle	Geometry	Low

4.2 Component identification procedure

Steps involved in identifying the affected components:

Step 1 The propagation starts with the 'Vol. of Paste' as Initiator.

Step 2 The program checks for 'Vol. of Paste' in the 'Initiator' column and then picks up the 'Target' component and the 'Type of Change' from Figure 5. Here the 'Target' component is 'Cartridge' and 'Type of Change' is 'Size'.

Step 3 These two are taken as an input for the next level and a row with 'Cartridge' and 'Size' as the 'Initiator' and 'Type of change', respectively, is searched. If we find one, then the propagation goes further or else it stops at this level. Here, we have 'Handle', 'Slider' and 'Handle Cap' as the affected 'Target' components.

Step 4 For the next level these components 'Handle', 'Slider' and 'Handle Cap' are the 'Initiator' components and 'Type of Change' being the same, 'Size'.

Step 5 Further down we identify the:

- 'Paste Delivery Tube', 'Slider' and 'Handle' as the Targets from 'Handle'
- 'Handle' and 'Cartridge' as the Targets from 'Slider'
- 'Handle Cap' does not have any target as it does not affect any components.

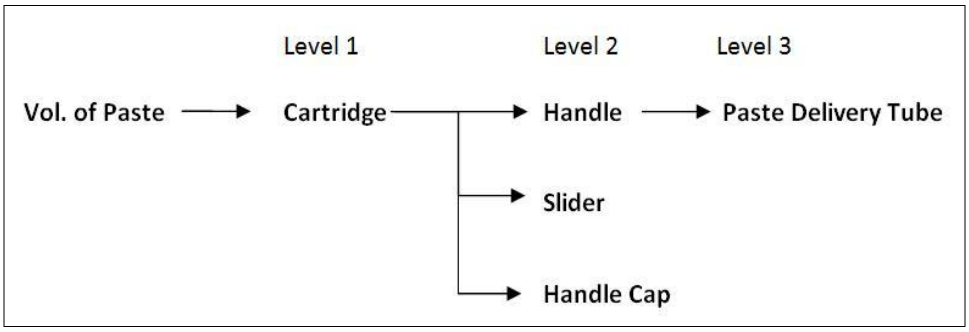
Step 6 As 'Slider', 'Handle', 'Cartridge' have already been the 'Initiator' components before, they are ignored and 'Paste Delivery Tube' is the only component which did not act as 'Initiator'.

Step 7 Going further down, 'Paste Delivery Tube' does not affect any components and hence the final list of affected components are 'Cartridge', 'Handle', 'Slider', 'Handle Cap' and 'Paste Delivery Tube'.

4.3 Results

The following figure shows the propagation result for an EC related to the performance factor of the toothbrush, *i.e.*, Volume of paste.

Figure 6 Change propagation result for the discussed case



5 Conclusions and limitations

A simple model can effectively identify the components affected by an EC provided all the relationships between the components are captured during the design stage. High quality cross-functional project teams are identified as one of the key factors for the success in Product Development (Cooper and KleinSchmidt, 2007). Use of such teams will help to capture all the possible dependencies and make the described tool effective. In case any dependencies are not identified they will probably be identified in the future when a change comes in and can be added to the database of dependencies at that point of time. This gradually improves the effectiveness of the model over time eliminating the uncertainties.

Ignoring the Design phase of the product life cycle substantially increases the complexity of the EC propagation. Complex parts have more dependency data and the resultant database is large. This model can handle large databases with ease, with the availability of latest database technologies. Use of advanced concepts like object-oriented programming and relational databases increases the performance of the framework.

Human intervention is minimised to a large extent by linking the attributes to the components. This model identifies the components and the way they are affected and does not measure the impact or risk of a particular EC. It is assumed that a change in the product data during design phase is part of the design of the product and is taken care of efficiently. Though it takes a good amount of effort to capture the dependencies between various components the increased use of design methodology like concurrent engineering makes it easy. The present model greatly depends upon the accuracy of the dependency information captured during the design phase and so greatly relies on the expertise and experience of the designers to foresee the possible changes to each component of the product and its affect on other components.

In the future, an extra field to the data can be added to determine the department responsible for a particular change, which is decided after all the possible changes that may come in during the life cycle of the product are discussed by brainstorming. Factors that determine the risk and impact of an EC can be merged into the model.

Acknowledgement

The toothbrush example was developed together with Weilin Li and Bochao Wang of Syracuse University.

References

Aurich, J. and Martin, R. (2007) 'Engineering change impact analysis in production using VR', in *Digital Enterprise Technology*, Springer, p.75.

Bhuiyan, N., Gatard, G. and Thomson, V. (2006) 'Engineering change request management in a new product development process', *European Journal of Innovation Management*, Vol. 9, No. 1, p.5.

Bouikni, N., Desrochers, A. and Louis, R. (2006) 'A product feature evolution validation model for engineering change management', *Journal of Computing and Information Science in Engineering*, Vol. 6, No. 2, p.188.

Browning, T.R. (2001) 'Applying the design structure matrix to system decomposition and integration problems: a review and new directions', *IEEE Transactions on Engineering Management*, Vol. 48, No. 3, p.292.

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

Cohen, T., Navathe, S.B. and Fulton, R.E. (2000) 'C-FAR, change favorable representation', *Computer Aided Design*, Vol. 32, Nos. 5–6, p.321.

Cooper, R.G. and Edgett, S.J. (2008) 'Maximizing productivity in product innovation', *Research Technology Management*, Vol. 51, No. 2, p.47.

Cooper, R.G. and KleinSchmidt, E.J. (2007) 'Winning businesses in product development: the critical success factors', *Research Technology Management*, Vol. 50, No. 3, p.52.

Diprima, M. (1982) 'Engineering change control and implementation considerations', *Production and Inventory Management*, First quarter, p.81.

Do, N., Choi, I.J. and Song, M. (2008) 'Propagation of engineering changes to multiple product data views using history of product structure changes', *International Journal of Computer Integrated Manufacturing*, Vol. 21, No. 1, p.19.

Eckert, C., Clarkson, P.J. and Zanker, W. (2004) 'Change and customisation in complex engineering domains', *Research in Engineering Design*, Vol. 15, No. 1, p.1.

Huang, G.Q. and Mak, K.L. (1999) 'Current practices of engineering change management in UK manufacturing industries', *International Journal of Operations Production Management*, Vol. 19, No. 1, p.21.

Huang, G.Q., Yee, W.Y. and Mak, K.L. (2003) 'Current practice of engineering change management in Hong Kong manufacturing industries', *Journal of Materials Processing Technology*, Vol. 139, Nos. 1–3, p.481.

Keller, R., Eckert, C.M. and Clarkson, P.J. (2005) 'Multiple views to support engineering change management for complex products', *Proceedings of the Third International Conference on Coordinated & Multiple Views in Exploratory Visualization CMV 2005*, IEEE Computer Society, London, UK, p.33.

Lee, H.J., Ahn, H.J. and Kim, J.W. (2006) 'Capturing and reusing knowledge in engineering change management: a case of automobile development', *Information Systems Frontiers*, Vol. 8, No. 5, p.375.

Moon, Y.B. (2007) 'Enterprise Resource Planning (ERP): a review of the literature', *Int. J. Management and Enterprise Development*, Vol. 4, No. 3, p.235.

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, p.208.

Riviere, A., Dacunha, C. and Tollenaere, M. (2002) 'Performance in engineering change management', in G. Gogu, D. Doutellier, P. Chedmail and P. Ray (Eds.) *Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering*, 1st ed., The Netherlands: Kluwer Academic Publishers, p.369.

Rouibah, K. and Caskey, K.R. (2003) 'A workflow system for the management of inter-company collaborative engineering processes', *Journal of Engineering Design*, Vol. 14, No. 3, p.273.

Rutka, A., Guenov, M.D., Lemmens, Y., Schmidt-Schäffer, T. and Coleman, P. (2006) 'Methods for engineering change propagation analysis', *25th International Congress of the Aeronautical Sciences*, Hamburg, Germany.

US Military Standard 480B (1988) 'Configuration control, engineering changes, deviations and waivers', Department of Defense, USA.

Yang, M.C., Wood, W.H., III and Cutkosky, M.R. (2005) 'Design information retrieval: a thesauri-based approach for reuse of informal design information', *Engineering with Computers*, Vol. 21, No. 2, p.177.