Extended Process to Product Modeling (xPPM) for integrated and seamless IDM and MVD development

Ghang Lee*, Young Hyun Park, Sungil Ham

Building Informatics Group, Department of Architectural Engineering, Yonsei University, South Korea

ARTICLE INFO

Article history:
Received 29 June 2012
Received in revised form 23 August 2013
Accepted 27 August 2013
Available online 17 September 2013

Keywords:
Extended Process to Product Modeling (xPPM)
Information delivery manual (IDM)
Model view definition (MVD)
Building Information Modeling (BIM)
Product modeling
Process modeling

ABSTRACT

This paper proposes a new extended Process to Product Modeling (xPPM) method for integrated and seamless information delivery manual (IDM) and model view definition (MVD) development. Current IDM development typically uses Business Process Modeling Notation (BPMN) to represent a process map (PM). Exchange requirements (ERs) and functional parts (FPs) specify the information required when information is exchanged between different activities. A set of information requirements, specifically defined as a subset of Industry Foundation Classes (IFC), is called an MVD. Currently however, PMs, ERs, FPs, and MVDs are developed as separate documents through independent development steps. Moreover, even though ERs and FPs are designed to be reused, tracking and reusing the ERs and FPs developed by others is practically impossible. The xPPM method is proposed to provide a tight connection between PMs, ERs, FPs, and MVDs and to improve the reusability of predefined ERs and FPs. The theoretical framework is based on the approach of the Georgia Tech Process to Product Modeling (GTPPM) to suit the IDM development process. An xPPM tool is developed, and the validity of xPPM is analyzed through the reproduction of existing IDMs and MVDs. The benefits and limitations of xPPM and lessons from the applicability tests are discussed.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Industry Foundation Classes (IFC) are the standard data specifications for exchanging information throughout the entire lifecycle of a facility [18]. The data that can be exported from a Building Information Modeling (BIM) tool is, however, intrinsically limited to the data included in the BIM tool. Thus a single software application cannot produce all the data. Thus a single software application cannot produce all the data. Thus a single software application cannot produce all the data. Thus a single software application cannot produce all the data. The other data required by various tasks within a BIM project is, by necessity, acquired through a data exchange between various tools.

For efficient and seamless data exchange, an early IFC case study [3] identified the necessity of exporting data in various views so that subsequent tasks could use the information without (or with minimal) reprocessing (of) the data. However, the same problem persists. The inflexibility of providing IFC data in various views to support various data use cases raises questions about the practicality of IFC [42].

Nevertheless, it is difficult to argue that this problem is solely attributable to software vendors. Even when a software developer wants to provide various types of IFC export options, no clear industry-agreed development guidelines exist about which information should be exported to support the various data exchange scenarios. In an effort to provide industry-wide guidelines to software developers for the development of IFC import/export functions for specific data exchange scenarios, buildingSMART, the international organization that develops and maintains the IFC specification, proposed the information delivery manual (IDM) and model view definition (MVD) [58].

IDM was first proposed by Jeffrey Wix in 2005 [44]. In 2010, the method and forms for specifying IDM documents became an international standard, ISO 29481-1 [17]. ISO 29481-1 defines an IDM as a “specification of information to be exchanged about a particular topic or business requirement in the construction process.” An MVD, defined by buildingSMART as a “subset of the IFC schema that is needed to satisfy one or many exchange requirements (ERs) of the architecture, engineering, and construction (AEC) industry” [5], specifies exchanged information in the context of IFC. An IDM is composed of a project map (PM), ERs, and functional parts (FPs). The IDM standard defines these elements and their relationships to one another as follows:

- **Process map**: A representation of the relevant characteristics of a process for a defined purpose. A process map includes many exchange requirements.

*Corresponding author.

E-mail addresses: glee@yonsei.ac.kr (G. Lee), nuyhyun@yonsei.ac.kr (Y.H. Park), archispaces@yonsei.ac.kr (S. Ham).

1474-0346/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.

http://dx.doi.org/10.1016/j.aei.2013.08.004
as the notation for specifying an IDM process map because BPMN is not mandatory, but IDM developers generally use BPMN for PM development. The use of Business Process Modeling Notation (BPMN) is not mandatory, but IDM developers generally use BPMN as a PM modeling notation [17].

The second step is information requirements’ collection and modeling. In a broad sense, use-case specification step can be included in the requirements’ collection and modeling step. A corresponding step in IDM is the specification of ERs. ERs are sets of data exchanged between activities in the PM.

The third step is to “spell out” (that is, to explicitly define) the individual information items that are included in each ER as FPs. Later, FPs are mapped to an MVD of the IFC.

The execution of these three steps is already complex. However, the IDM development process involves many more complex steps and mapping processes at different levels, including mapping processes between a single PM and PMs, between a PM and ERs, between ERs and FPs, between FPs and MVDs, and finally, between FPs and MVDs. These complex and multilevel mapping processes cause several problems.

First, it is practically impossible to track and reuse existing ERs, FPs, and MVDs, despite the fact that these elements are designed to be reused. Second, the weak connections between PMs, ERs, FPs, and MVDs caused by the multilevel mapping processes are a major hindrance to the development and application of a method for checking the validity of resultant MVDs [14,28]. Third, the development and review process takes a couple of years, and the resultant models typically include many errors. More details on these and other issues are discussed in Section 2.

This study proposes a method for extended Process to Product Modeling (xPPM) that is based on the Georgia Tech Process to Product Modeling (GTPPM) [24,26,27,30] to support integrated and seamless IDM and MVD development. It demonstrates the effectiveness of xPPM through an IDM–MVD development case study using a newly developed tool based on xPPM.

This paper first reviews previous works related to IDM and MVD development. It then describes the problems with the current IDM and MVD development processes. The fourth section describes the xPPM approach and how the approach resolves the complex issues of mapping between PMs, ERs, FPs, and MVDs during and after the IDM–MVD development processes. The fifth section validates the applicability of xPPM as a method of IDM–MVD development by reproducing existing IDMs using xPPM, and discusses the benefits and limitations of xPPM as well as the lessons learned from the validation tests. The IFC and IDM/MVD standards use many acronyms. The following summarizes the acronyms used throughout this paper in alphabetical order:

- BIM: Building Information Modeling.
- ER: Exchange Requirement.
- FP: Functional Part.
- GTPPM: Georgia Tech Process to Product Modeling.
- MVD: Model View Definition.
- PM: Process Map.
- xPPM: eXtended Process to Product Modeling.

2. Previous efforts

The most commonly known software engineering effort aimed at integrating the processes of requirements specification and data specification is the Unified Modeling Language (UML) [35]. UML was created to support the development of object-oriented software applications. As its name indicates, UML was developed by collecting and unifying major graphic notations for process and data modeling. The scope of UML is so broad that any current software engineering method will partially overlap with UML notation in one way or another. For example, UML activity diagrams greatly overlap with BPMN, the modeling notation for specifying a PM. Because UML is a generic modeling method, however, an excessive number of elements and semantics to support IDM–MVD specification structures and IFC semantics are required to be able to use UML as an effective method for IDM–MVD development.

The first study to directly address the issue of integrating IDM and MVD modeling processes was conducted by Lee et al. [25] using the GTPPM approach. GTPPM is a product modeling method developed at the Georgia Institute of Technology (Georgia Tech) to tightly link a process model and a product model [24,26,27,29,30]. A product model is a data schema that includes information for designing, engineering, and fabricating products, such as buildings, automobiles, and other products. The work of Lee et al. [25] showed potential for the improvement of IDM/MVD development by integrating process and product data modeling. However, GTPPM could not fully support the IDM and MVD modeling processes for several reasons:

- The eventual goal of IDM is to define a MVD of IFC, but GTPPM’s data structure could not fully support an MVD development process that included the specification of user-defined aggregated data types.
- An IDM should be reported in a specific format defined by ISO 29481-1 and requires a lot of metadata, such as revision history, author information, and prerequisite information. GTPPM does not support the metadata required by the IDM format.
- New IDM development requires the reuse of existing IDMs, ERs, FPs, and MVDs. However, neither GTPPM nor the current IDM development methods have mechanisms to support the reuse of existing IDM and MVD data. This inevitably going to become a serious problem as the number of IDMs and MVDs increased.
- GTPPM could not specify information that was in an aggregation relationship.
- GTPPM used its own notations based on UML activity diagrams, which did not support BPMN.

Another attempt to improve the IDM process was made in a study by Eastman et al. [14]. Eastman et al. provided details about how ERs and FPs could be cross-referenced at various levels. They referred to the ER as an exchange model (EM), and FP as an exchange object (EO), and the study proposed slightly modified forms for specifying EMs and EOs from the standard ER and FP forms.
Table 1 compares the terms used in the IDM, GTPPM, and the exchange model approaches. The proposed xPPM uses the same terms as the original IDM standard.

The major difference between the original IDM and Eastman’s approach was that the latter attempted to specify more details about required information items so that the ERs and FPs could be defined proximate to the IFC structure and the MVD could be mapped directly to the IDM with minimal iterative mapping processes.

A series of studies focusing on the improvement of the MVD specification process was conducted by Katranuschkov and Weise and their respective teams [20,43]. These studies developed an MVD editing tool called ViewEdit, which helps users easily specify multiple MVDs by selecting and adding IFC entities and properties [20]. ViewEdit was implemented using a metadata schema called the Generalized Model Subset Definition Schema (GMSD) [43] to store user-defined subsets.

Another effort to improve MVD development is an automated MVD extraction method proposed by Lee [22]. Lee presented a method for extracting the smallest MVD that matches a user-defined concept by introducing the notion of a minimal set and nine rules. These previous studies, however, focus on MVD development rather than the integration or complex mapping issues between PMs, ERs, FPs, and MVDs.

Concerns regarding the maintenance of complex multiple mapping relations between existing and current ERs, FPs, and MVDs still remained. Section 3 discusses in detail the challenges of the current IDM/MVD development processes.

### 3. Challenges of the current IDM/MVD development processes

The current processes for IDM/MVD development are challenging for several reasons. IDM developers face a challenge from the very first step of IDM development, namely, PM specification. The current IDM standard recommends using BPMN for PM notation. However, BPMN comprises over 160 symbols [34], and the number grows with every release. In addition, the complex and ambiguous rules associated with the symbols make it difficult for developers to fully understand and use the symbols correctly.

Many previous studies in software engineering have identified the issue of missing BPMN symbols due to the overwhelming number of symbols and rules as critically problematic for successful development [21,38–41]. As a matter of fact, only a small subset of BPMN is used in IDM development. We analyzed the use of BPMN symbols in existing PMs and derived a subset of BPMN common to current development processes. The subset of BPMN used in xPPM and the analysis process are discussed in detail in Section 4.

The second challenge, as mentioned above, is the tracking and reuse of existing ERs and FPs. Existing ERs and FPs are designed to be reused. Needless to say, while the reuse of the existing ERs and FPs defined in IDMs and specified by others is practically impossible, it should be emphasized that the tracking and reuse of ERs and FPs defined within an IDM by the same author is also challenging because the number of ERs and FPs increases throughout the IDM specification process. As soon as it seems that a collection of existing ERs and FPs is finished, often after weeks of effort, new ERs and FPs may be released while other ERs and FPs are being compiled concurrently. This problem will become even more serious as the number of IDM and MVD specifications increases in the future.

The third challenge is the complex mapping steps between PMs, ERs, FPs, and MVDs. Fig. 1 illustrates the mapping relationships between a PM, ERs, FPs, and MVDs. In effect, a PM may include many ERs; a PM cannot be mapped directly to FPs; an ER or an FP cannot include a PM; an ER may be composed of several ERs and/or FPs; ERs, which belong to a given ER, may belong to multiple other ERs as well, and likewise FPs, which belong to an ER or an FP, may be part of other ERs or other FPs. Furthermore, an FP cannot include an ER; FPs are mapped to concepts in an MVD; and a concept is mapped to a small subset of an MVD.

The IDM committee adopted these complex mapping structures for several reasons. First, specification of ERs and FPs takes a great deal of time and effort. The committee tried to ease the specification process by allowing developers to declare that any designated ER or FP has, or includes, the same content as previously defined ERs or FPs. Second, the IDM and MVD are separate documents in order to ensure that an IDM specification can be independent of the IFC versions. The protocol is to develop a new MVD when a new IFC version is released.

However, the complexity of these mapping relationships makes the reuse of information very difficult both for information specified outside an IDM or MVD, as well as for information specified within an IDM or MVD. Moreover, if a new version of IFC is released, then the generation of a new MVD corresponding to the target IDM is not any simpler than updating FPs in an IDM to conform to the new version of IFC. However, this does not mean that the old IDM will be abandoned. For example, the old IDM can simply be named an IFC2x3 and the updated IDM an IFC4. Thus the separation of IDMs and MVDs does not have a practical implication. Rather the separation serves only to make the IDM and MVD development process overly complex.

The fourth challenge has to do with the maintenance and tracking of IDM and MVD documents. Currently, PM, ER, FP, and MVD documents are produced as separate document files. Relationships between the documents are very difficult to track and understand because the documents do not have links between them, despite their very complex mapping characteristics. An easy solution is to add hyperlinks to each document. However, the simple hyperlink approach does not allow developers to search through existing ERs, FPs, and MVDs. To tackle both the second and fourth challenges, the proposed xPPM approach suggests storing information about ERs, FPs, and MVDs in a searchable format, such as eXtensible Markup Language (XML), instead of a document file format. Section 4 describes in detail how the xPPM approach deals with these challenges.

### 4. The xPPM approach

The xPPM approach proposes the following three solutions to the four challenges inherent to IDM/MVD development, which were described in Section 3:

1. The use of a limited subset of BPMN symbols for PM development instead of the full set of BPMN symbols.
2. The elimination of the distinction between an IDM and an MVD.
3. The storage of ERs, FPs, and MVDs in XML instead of document files.

To derive the subset of BPMN symbols required for PM development, we collected and analyzed 54 processes in 14 existing PMs.
collected in 2011 from the IFC Solution Factory \[4\] and the buildingSMART International IDM website \[7\]. Table 2 shows the list of PMs used in the frequency analysis of BPMN elements. Currently these IDM documents have been consolidated into the buildingSMART International IDM overview site \[8\]. Preliminary analysis results were reported by Park et al. \[37\]. Fig. 2 shows the frequency of BPMN symbols used in the 54 processes in the previous PMs.

The analysis showed that the previous PMs were developed using a total of 36 BPMN elements out of a possible 161 BPMN 2.0 elements. Details regarding the 161 BPMN 2.0 elements can be obtained from the BPMN 2.0 specification \[34\]. The 36 BPMN elements used in previous PMs could have been defined as a BPMN subset for IDM specification. However, many of the 36 BPMN elements were scarcely used. We therefore further simplified the BPMN subset for IDM into 22 elements by merging less frequently used elements into very frequently used elements to derive the essential subset of BPMN elements for IDM specification.

First, we selected and included the very frequently used elements in the essential subset. Very frequently used elements were basic process-modeling elements (e.g., the task, the start, intermediate, and end events, the sequence and association flow, and the gateway), the expanded and collapsed sub-processes that were used to group a set of process elements in different levels of detail, the pool and lanes that represented participants, and the data object that represented information exchanged between participants.

We then categorized the less frequently used elements into the elements that were substitutable and non-substitutable by the very frequently used elements described above. For example, special types of sub-processes, events, and flows were categorized as substitutable by their top-level elements, which had already been included in the set of very frequently used elements. These special types of elements represented certain semantic differences between elements, such as repeated vs. non-repeated tasks. However, the differences on that level were not critical in developing IDMs, and if necessary, they could be represented by combining several other elements. For example, a repeated task can be represented as a looped sub-process instead of using the repeated task symbol. Non-substitutable elements included three types of gateways, each of which represents the exclusive OR (or XOR) condition, the AND condition, and the inclusive OR condition, respectively, and the three types of artifacts (i.e., the group, the message, and the text annotation). Fig. 3 shows the essential BPMN subset for IDM specification.

Currently, xPPM supports these 22 BPMN elements. However, the use of the 22 essential subset elements are more a recommendation than a mandatory rule as the use of BPMN as a PM notation in IDM is also a recommendation.

Table 2

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Developer</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDM for Precast Concrete</td>
<td>2009</td>
<td>US PCI</td>
<td>IFC_Solution_Factory</td>
</tr>
<tr>
<td>BSA-002_Design_to_EnergyPOerformance_Analysis_PM</td>
<td>2009</td>
<td>US GSA, Statsbygg, and Senate Properties</td>
<td>IFC_Solution_Factory</td>
</tr>
<tr>
<td>GSA-001_Design_to_Spatial_Prog_Validation_PM</td>
<td>2009</td>
<td>US GSA, Statsbygg, and Senate Properties</td>
<td>IFC_Solution_Factory</td>
</tr>
<tr>
<td>GSA-002_Design_to_Circulations&amp;Security_Analysis_PM</td>
<td>2009</td>
<td>US GSA, Statsbygg, and Senate Properties</td>
<td>IFC_Solution_Factory</td>
</tr>
<tr>
<td>pm_electrical_engineering</td>
<td>2006</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_energy_analysis</td>
<td>2006</td>
<td>AEC3</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_excavation</td>
<td>2008</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_facilities_management</td>
<td>2006</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_HVAC_engineering</td>
<td>2006</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_MEP_QTO</td>
<td>2008</td>
<td>AEC3</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_piping_engineering</td>
<td>2006</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_service_life_planning</td>
<td>2008</td>
<td>AEC3</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_structural_engineering</td>
<td>2007</td>
<td>buildingSMART</td>
<td>Old IDM FTP</td>
</tr>
<tr>
<td>pm_cost_modelling</td>
<td>2007</td>
<td>Jeffrey Wix</td>
<td>Old IDM FTP</td>
</tr>
</tbody>
</table>

Fig. 1. Complex mapping steps between a PM, ERs, FPs, and MVDs.
The second proposed solution is to eliminate the distinction between IDM and MVD, and to incorporate the MVD process into IDM and replace the FP specification process. Fig. 4 illustrates the concept. Instead of having an MVD as a separate document from an IDM (Fig. 4a), xPPM proposes specifying FPs directly using IFC elements. It sounds logical to separate a technology-specific solution (i.e., an MVD) from a generic definition (i.e., the IDM). However, FPs in their current form are already very close in shape to the current IFC and are not any less complex than the IFC definition. This situation only compounds the amount of work if developers are required to specify FPs independently of IFC and then to specify them again based on a new IFC. Thus it is reasonable to merge the processes of FP and MVD specification. Currently, the ISO 29481-1 IDM standard does not explicitly state that FPs cannot be specified using IFC. Indeed, there are many examples of early IDMs that specified FPs using IFC. Generally speaking, however, an MVD specification process is treated as a separate process from IDM development.

Nevertheless, if every FP has to be defined based on the prevailing IFC, then information items that are not yet supported by IFC cannot be included in an IDM. For such cases, xPPM suggests that users add any items not already included in IFC as user-defined items in the context of a specific IFC element by using “USR_” as the prefix. For example, if IfcMaterial requires a new “finish” property, then it can add the user-defined item “USR_finish” to IfcMaterial. If a new “road” entity is required, the item can be added as “USR_road” to a subtype of IfcElement.

The third proposed solution is to store and manage ERs, FPs, and MVDs in XML instead of separate documents. If ERs and FPs are specified as discrete documents, as they currently are, and an IDM includes 20 ERs, then 20 ER document files will be produced along with probably a greater number of FP document files to support the IDM.

Figs. 5 and 6 are XML schemas of the ER document structure and the FP document structure. The proposed data structure is the same as those for the ER and FP requirements specified in ISO 29481-1 except that the FP uses IFC to specify information items. An ER is composed of a header, an overview, and an information requirements section. The header includes a name, a unique identifier, a change log for the document, and other basic information items about the ER document. The overview specifies the scope and aims of the document. The information requirements section is composed of preconditions for information exchange and a list of specific information units required by a target PM. An FP includes a header section and a list of required information items.

Fig. 2. Frequency of BPMN element usage in previous PMs (selected elements are shaded grey).

Fig. 3. An essential set of BPMN symbols for PM development in xPPM.
During the review of this paper, buildingSMART International released mvdXML [32]. mvdXML serves basically the same role as the FP XML schema specified in Fig. 6. The difference is that mvdXML is developed based on the MVD document structure whereas FP XML was developed based on the FP document structure. In the future, FP XML can be modified to support mvdXML.

In xPPM, the associations between a PM, the ERs, the FPs, and the MVDs are maintained through several links. The “Data Object” shape of the BPMN (shown in Fig. 3) is used to represent an ER in a PM. An “Information Unit” of an ER (shown in Fig. 5) is linked to an FP. An “Infotem” of an FP (shown in Fig. 6) is linked directly to the IFC elements without having a separate MVD. Based on these aspects of the xPPM approach, an xPPM tool was developed. Section 5 describes in detail how the xPPM tool is configured and how it supports these concepts.

5. Development of an XPPM tool

An xPPM tool was developed using Java as the programming language and yFiles [46] as the graphic engine. The xPPM tool mainly comprises four interfaces: the Process Modeling interface (Fig. 7), the ER interface (Figs. 8 and 9), the FP interface (Fig. 10), and the IFC Menu (Fig. 11). Modelers define a process map by

Fig. 4. Elimination of the distinction between IDM and MVD documents.

Fig. 5. XML schema of the xPPM ER document structure.
dragging and dropping BPMN elements from the BPMN list in the right pane of the xPPM Process Modeling interface. A BPMN rule checker associated with flow types was implemented to minimize potential errors.

After specifying a process map, modelers can specify ERs by double-clicking an instance of the data object. The ER interface guides modelers through four sections of an ER, as defined in the ISO 29481-1 IDM standard. Existing ERs can be read into a PM and reused without specifying them.

The FP interface can be accessed from the Information Units section of the ER interface by pressing New or Edit to add or edit information items. Existing ERs and FPs in the XML format can be imported into a PM. The ERs and FPs specified in XML can also be shared and searched through shared uniform resource locators (URLs), but functionality has not been implemented to date.

An information item of an FP can be added in three ways. The options include adding information items by selecting existing FPs, selecting an IFC element from the IFC Menu shown in Fig. 11, or specifying a user-defined information item in the context of IFC in the IFC Menu.

The IFC Menu can be viewed as a graphic user interface for specifying an information item in the context of IFC. The IFC Menu is composed of the “information selection” window (on the left-hand side of Fig. 11) and the “added information” window (on the right-hand side of Fig. 11). The “information selection” window shows options that users can select, in a similar way to items on a restaurant menu. For example, if IfcObject is selected in the Top-Level Class window pane, as shown in Fig. 11, the other window panes show the subtypes and associated, derived, and inversed properties of IfcObject so that users can choose an information item from a limited number of options without worrying about the EXPRESS grammar and without needing to know the IFC structure in detail.

The selected information item will be listed in a tree structure (the top-right pane in Fig. 10) and also automatically in a native IFC format (the bottom right pane in Fig. 10). The bottom right pane of the “information selection” window lists FPs that include IfcObject so that users can select an FP from the list—if an intended FP exists—without re-specifying a set of information items.

User-defined properties are defined using the “user-defined property” interface (Fig. 12). Constraints such as OPTIONAL,
UNIQUE, and cardinality can be specified using the options in the bottom right corner of the interface.

Fig. 13 shows a portion of ER data specified in XML. The example ER file was exported from an xPPM–PM model. FPs can also be exported from a PM model as separate XML files or as a single integrated XML file. IFC entities in integrated XML files can be easily extracted and converted into an IFC subset schema (view) for the specific IDM. We developed a separate IDM Generator tool so that these ER and FP files can be automatically converted into formatted MS Word files.

The benefits of xPPM were analyzed by reproducing and analyzing existing IDMs. For more details on xPPM, an introduction video of the functioning xPPM tool are available on the YouTube channel of the Building Information Group at Yonsei University [23].
Section 6 reports the analysis results and benefits of xPPM for IDM development.

6. Validation

The validity of xPPM for IDM development was analyzed through an xPPM reproduction of existing IDMs. With effect March 2013, 101 IDM/MVD projects were registered to the IDM official site [8]. Of the 101 projects, four were approved and 63 were in the draft stage. The remaining 34 projects were in the ideation stage or on hold. We collected and analyzed 47 IDM projects, which could be downloaded from various sources, for the validation of xPPM [1,2,4,6–8,13,15,16,19,31,33,36,45]. Table 3 summarizes the distribution of IDM/MVD documents used in the validation by document type. Although the documents were collected from 47 IDM projects, the numbers of documents in each document type is less than 47 and the numbers for each document type differ because most projects included only some of the documents (e.g., only PMs or only MVDs), as shown in Table 5.

Fig. 9. The xPPM–ER information unit interface.

Fig. 10. The xPPM–FP interface.
First, we examined how effectively xPPM could deal with problems during the PM specification process by reproducing 39 existing PMs using xPPM. The reproduction of the existing PMs validated that the 22 BPMN elements were sufficient to express the semantics in the existing PMs. Furthermore, during the reproduction of the existing PMs, the misuse of the BPMN symbols was frequently observed in the existing PMs. Table 4 summarizes the common errors associated BPMN rules and the frequency of the errors in the existing PMs that were eliminated by adopting xPPM. Examples of the common errors included the following:

---

**Fig. 11.** The xPPM-IFC menu interface.

**Fig. 12.** The user-defined property interface.
a. Data objects should be linked with events or tasks using data associations, but in some cases in the existing PMs, data objects were connected using message flows (e.g., the data object on the far left-hand side of Fig. 14). This was the most frequent error in that we observed 656 cases of misuse.

b. Regarding message events, when a message is outgoing from a task, a throwing-type message event should be used. In 109 cases in the existing PMs, the catching-type message events were used (e.g., the outgoing message events in Fig. 14).

c. Similarly, we also observed misuse of off-page connector symbols.

d. When the start event is used, the end event must exist. In some cases (7 in total), either the start or the end event was missing.

e. A process is a sequence of activities. Thus, activities in a process should be linked by the sequence flow from the beginning to the end. Among the 39 PMs examined, 18 projects had disconnected processes.

These errors were not observed in the PMs created using xPPM. This result is inevitable because it is cognitively much easier to recall and select the correct symbols and associated rules from a set of 22 symbols than from a set of 161 symbols. In addition, the BPMN rules associated with the flows implemented in the xPPM tool also helped modelers to prevent errors. However, this automated rule-checking function is the secondary mechanism for preventing errors because the rule checker can only detect syntactic errors and cannot catch semantic errors. Semantic errors can only be prevented when modelers clearly understand the meaning and usage of each process-modeling element. The test result shows that the use of a small number of selected BPMN symbols could help modelers learn and deploy BPMN correctly during PM specification.

Second, the effectiveness of eliminating the distinction between FPs and MVDs could not be validated by reproducing the existing IDMs. However, we collected 47 sets of IDM/MVDs, which had been developed during the review of this paper, from the IDM Overview website [8] and analyzed how they were developed. More than half of the IDM/MVDs (55%) were developed without

**Table 3**

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of collected cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td>39</td>
</tr>
<tr>
<td>ER</td>
<td>35</td>
</tr>
<tr>
<td>FP</td>
<td>12</td>
</tr>
<tr>
<td>MVD</td>
<td>28</td>
</tr>
</tbody>
</table>

**Table 4**

<table>
<thead>
<tr>
<th>Error type</th>
<th>Associated BPMN rules</th>
<th>Number of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data objects connected using message flows</td>
<td>A data object may not be either source or target of sequence flows and message flows</td>
<td>656 cases PMs</td>
</tr>
<tr>
<td>Misuse of throwing message events instead of</td>
<td>An intermediate event with outgoing message flow must be throwing type with message</td>
<td>109 cases PMs</td>
</tr>
<tr>
<td>catching message events</td>
<td>or multiple trigger</td>
<td></td>
</tr>
<tr>
<td>Misuse of off-page connectors (link intermediate</td>
<td>A link intermediate catch event must not be the target of a sequence flow</td>
<td>10 cases PMs</td>
</tr>
<tr>
<td>catch event)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing start or end events</td>
<td>If there is a start event, then there must be at least one end event</td>
<td>7 processes</td>
</tr>
<tr>
<td>Disconnected process</td>
<td>If the activity does not have an outgoing sequence flow, then the activity marks the end</td>
<td>18 processes</td>
</tr>
</tbody>
</table>

**Fig. 13.** An ER specified in XML format.
specifying the FP documents. Of the 47 IDM/MVD sets, 40% stopped at the ER or FP specification phase and did not move onto the MVD specification phase. Only one case of the IDM/MVDs was fully developed and defined the PMs, ERs, FPs, and MVDs. This shows that the distinction between FPs and MVDs is not essential in defining IDM/MVDs, and the IDM/MVD process can be made more efficient by not requiring the distinction. In addition, the fact that quite a few MVDs were developed without specifying PMs and/or ERs shows that the current PMs, ERs, FPs, and MVDs are not closely interconnected, and thus, the initial intention for including PMs and ERs in the IDM process is becoming diluted. Furthermore, we could not find any case in which existing ERs and FPs were reused unless the IDM/MVD projects stemmed from the same project.

To test how well xPPM can support the initial IDM’s intent of use-case-based MVD (IFC subset) development, similar to the PM test, existing ERs and FPs were reproduced using xPPM.

Third, we analyzed the reuse cases of ERs and FPs in existing IDM/MVDs and reproduced a case, which specified all PMs, ERs, and FPs, to validate the effectiveness of xPPM in tightening the relationship between PMs, ERs, and FPs and increasing the potential for the reuse of these components.

First of all, by not having a mechanism to track, search, and share existing ERs and FPs, it took us two weeks to locate and collect the existing ERs and FPs, and another three months to review and analyze the content of the documents. Unsurprisingly, there were cases where some projects adopted all the IDM documents of other projects with minor modifications, but there was no project that reused existing ERs and FPs. Indeed, the projects that adopted all the IDM documents of other projects seemed to be successor projects of the other projects rather than new projects that intended to reuse existing ERs and FPs. The reproduction of existing PMs, ERs, and FPs of projects demonstrated how xPPM could reduce this tracking and search problem (see Table 6).

The reproduction process showed that many disconnection problems existed between PMs, ERs, and FPs, even in a single project. The disconnection problems and benefits of xPPM were as follows:

a. The most challenging task during the reproduction of ERs and FPs was to learn and keep track of pre-specified ERs and FPs. Just learning which ERs and FPs were available in the existing IDMs was impractical, and keeping track of ERs and FPs within a single IDM was becoming diluted. Because xPPM stores the ER and FP information in XML format and the xPPM tool has a function to import and show existing ERs and FPs as lists, we were able to simply select an ER and an FP from an existing list or an existing ER list.

b. The xPPM approach simplified the process of finding and specifying all the IFC entities and properties that correspond to a concept during an FP specification process. The approach could not guarantee that an FP included all the semantically required IFC elements, but it did guarantee that an FP included all the syntactically valid IFC elements required to form a complete IFC view.

c. Many errors in the definitions of the ER information items were also detected. An ER information item can be defined as one FP or a set of multiple ERs and FPs. However, an ER information item should not be defined as several FPs connected by “OR.” This erroneous approach to defining ER information items is the same as defining a single word with multiple meanings. The far right-hand column in Fig. 15 shows an example. If an

---

### Table 5

<table>
<thead>
<tr>
<th>Type</th>
<th>Used components</th>
<th>Number of cases</th>
<th>Percentage (%)</th>
<th>Sum (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All components</td>
<td>PM, ER, FP, MVD</td>
<td>1</td>
<td>2%</td>
<td>2</td>
</tr>
<tr>
<td>IDM and MVD without FPs</td>
<td>PM, ER, MVD</td>
<td>13</td>
<td>28%</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>PM, MVD</td>
<td>6</td>
<td>13%</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>ER, MVD</td>
<td>2</td>
<td>4%</td>
<td>23</td>
</tr>
<tr>
<td>IDM/MVDs stopped at the ER or FP phase</td>
<td>PM, ER</td>
<td>8</td>
<td>17%</td>
<td>40</td>
</tr>
<tr>
<td>MVD only</td>
<td>PM, ER, FP</td>
<td>11</td>
<td>23%</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>MVD</td>
<td>6</td>
<td>13%</td>
<td>13</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>47</td>
<td>100%</td>
<td>100</td>
</tr>
</tbody>
</table>

---

### Table 6

<table>
<thead>
<tr>
<th>Task</th>
<th>Time required (man-hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of existing ERs and FPs</td>
<td>Two weeks</td>
</tr>
<tr>
<td>Review and analysis of the contents of each document</td>
<td>Three months</td>
</tr>
</tbody>
</table>

---

Fig. 14. Example of misused BPMN shapes in an existing IDM.
### Information Requirements

<table>
<thead>
<tr>
<th>Type of Information</th>
<th>Information Needed</th>
<th>MAN</th>
<th>OPT</th>
<th>Actor</th>
<th>Functional Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Survey Area</td>
<td>Boundary of the area within which the survey is carried out. This should be defined using either a closed polyline, closed composite curve or as a 2 dimensional surface. The purpose of defining the area is simply to set the limiting boundary for the survey. The value of the object type for the annotation should be set to 'SurveyArea'. The value of the representation identifier for the 2D footprint should be set ‘Footprint’ regardless of the geometric representation used. The value of the representation type for the footprint should be set according to the geometric representation used. If the representation is a polyline or composite curve, then the value of the type should be set to ‘Curve2D’ or ‘GeometricCurveSet’. If the representation is a surface model, then the value of the type should be set to ‘Surface Model’.</td>
<td>✓</td>
<td></td>
<td>Land surveyor</td>
<td>fp_annotate_geometry[polyline] OR fp_annotate_geometry[composite_curve] OR fp_annotate_geometry[face_based_surface_model] OR fp_annotate_geometry[shell_based_surface_model]</td>
</tr>
<tr>
<td>Survey Area Identity</td>
<td>A unique identity should be applied to the survey in the context of the project using the IfcAnnotation.Name attribute. Note that this is the identity assigned to the survey and not the globally unique identifier assigned to the object representing the survey area.</td>
<td>✓</td>
<td></td>
<td>Land surveyor</td>
<td>IfcAnnotation.Name</td>
</tr>
</tbody>
</table>

**Fig. 15.** Example of an ER information item mapped to multiple FPs with “OR” relationships in an existing IDM.

| For site compositions | | |
|-----------------------|-----------------|-----|-----|-------|
| Set the composition type of the site occurrence (complex, element or part) | IfcSite.CompositionType ➔ IfcElementCompositionEnum | ✓ |   | |
| If the composition of the site is a ‘complex’, identify the elemental sites that form part of the complex | fp_nested(site) | ✓ |   | |
| If the composition of the site is a ‘part’, identify the elemental site of which it forms a part | fp_nested(site) | ✓ |   | |

**Fig. 16.** Example of two different FP concepts mapped to one FP in an existing IDM.

<table>
<thead>
<tr>
<th>Description</th>
<th>Entity/Property/Functional Part</th>
<th>MAN</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the direction of the local X axis.</td>
<td>IfcAxis2Placement3D.RefDirection IfcDirection.DirectionRatio</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 17.** Example of a conflict between the mandatory and optional constraints in an existing IDM.

---

Information item can be interpreted in several ways, then an IFC translator developer will have the freedom to choose any FP from the list of interpretations. Eventually, software users will not be able to expect a consistent data set from different software applications, regardless of whether or not the applications are built according to the same IDM. At this point, there is no reason to have an IDM. If a similar situation to the example in Fig. 15 occurs, then the information item should be broken down into smaller and more specific information items until each item can be mapped to one FP. The proposed xPPM prevented this problem by only allowing an ER to be defined as a set of ERs and FPs.

d. Another common error in the existing ERs and FPs was an information item that is defined as both “mandatory” and “optional” information, as shown in Fig. 17. This error did not occur during IDM specification processes using xPPM because xPPM allowed IDM developers to choose only one way of designating information.

e. Editorial errors were also common. Examples include the placement of wrong IFC entities and properties (e.g., placement of IfcSite instead of IfcSlab for a concept Slab), misrepresentation of IFC entity structures (e.g., missing supertypes/subtypes), and the misspelling of an IFC entity or property names. The xPPM approach reduced spelling errors by restricting the selection of the IFC elements to only the elements from the IFC Menu (Fig. 11).

In summary, through the reproduction of the existing IDMs, the proposed xPPM demonstrated the potential to reduce many types of common errors and problems with the current IDM development.
process. Although the effectiveness of eliminating the discussion between FPs and MVDs could not be demonstrated through the reproduction process, the current IDM/MVD specification practice showed that the distinction was not necessary. However, the IDM/MVD specification process still has many challenges that require resolution. Future tasks for xPPM and the IDM community are discussed in Section 7.

7. Discussion and limitations

The reproduction process of the existing IDMs not only validated the effectiveness of xPPM, but also identified several future tasks. First, the essential BPMN subset for IDM specification can be further simplified in the future. As described in Section 4, our analysis showed that a total of 36 BPMN elements were used. However, when only the number of BPMN elements used in each PM was counted, each PM used only a small number of BPMN elements. In the case of IDM, the required level of PM is usually very simple. That is, the representation of sophisticated business cases or rules is not often an issue for IDM because the context of information use is clearly depicted. Accordingly, several symbols may be merged into one high-level concept. This may restrict the capability of describing differences with the subtle nuances signified by discrete symbols, but the technique will simplify the rules of process modeling and, more importantly, will reduce the current learning curve as well as the errors made by users in PM construction.

Second, with respect to the reusability of ERs and FPs, xPPM can eliminate the problem of keeping track of existing ERs and FPs and can improve the reusability of existing ERs and FPs. However, if ERs and FPs are selected from existing ER and FP lists, they will be typically selected based on an ER name or an FP name, as opposed to being based on the actual content of an ER or an FP. In practical terms, it remains very difficult to check all the content of existing ERs and FPs. This means that there is a chance that a selected ER or FP may not include the content that an IDM developer intends or requires. Because an IDM developer knows the exact content of the ERs and FPs that he or she has specified, limiting the reuse of ERs and FPs within only one IDM document or within one PM project might be a simple way to mitigate this problem. In order to allow the reuse of ERs and FPs specified within other IDMs, however, the IDM community will need to develop a rigorous method for checking whether or not an ER includes all the information required by an IDM developer.

The third challenge was to determine whether or not all the required ERs or FPs for the target PM (i.e., a data exchange scenario) were identified. Technically, there was no way to check the validity or the completeness of the ERs or FPs for a PM. One solution might be to adopt the “dynamic information consistency checking method” [24,29] for xPPM, which verifies the validity and completeness of information based on the set-theoretical relations between the input and output information.

The fourth issue is the lack of a method to guarantee that the final MVD is a syntactically complete and valid IFC subset, which can be used as a data schema for developing an IFC translator. The specification of a valid IFC subset schema requires basic IFC elements that may not be specified as concepts or information items in ERs and FPs but should be included in an IFC subset. A good example is an ABSTRACT ENTITY, such as IfcRoot and IfcObject. A potential solution is to incorporate the “concept-based valid IFC subset extraction method” [22] into the xPPM IFC subset schema extraction module. The concept-based IFC subset extraction method is a rule-based subset method of schema generation that automatically extracts all the necessary IFC elements required to form a valid IFC subset schema. The necessary IFC elements include selected and non-selected elements based on a selected IFC entity or concept. If an MVD specification is automatically generated from an xPPM model using the “concept-based valid IFC subset extraction” algorithm, then the resultant MVD specification will populate even non-selected IFC elements to form a valid subset. This technique consequently guarantees that the automatically generated MVD is valid.

The fifth challenge was how to control the mapping of non-synonymous concepts to the same set of information items (FPs). It is not difficult to map several concepts to the same information item as long as the concepts signify the same thing (i.e., the concepts are synonymous). However, if several concepts signify different things, as in the example provided in Fig. 16 (i.e., a complex site and a partial site), the concepts should be mapped to two different FPs (i.e., “ftnest[complex_site]” and “ftnest[partial_site]”) instead of mapping both concepts to one FP “ftnest[site]”. Computer-based approaches are generally inefficient in dealing with semantic issues. A possible solution might be to systematically warn an IDM developer when he or she tries to map a concept to an FP that has already been mapped to another concept, and accordingly, to ask the IDM developer to check whether or not the current concept is equivalent to the concept that is already mapped to the same FP. If the two concepts are different, then the IDM developer should create a new FP.

The problems identified in this section continue to pose challenges and require resolution by the IDM/MVD community.

8. Conclusion

Our review of existing IDMs revealed that IDM development is challenging and error-prone due to an excessive and overwhelming number of BPMN shapes, difficulties in reusing and tracking ERs and FPs, the lack of a rigorous and standardized data structure for ERs and FPs, segmentation between a PM, ERs, and FPs, and difficulties in finding IFC elements that correspond to FP concepts. Common problems attributed to these issues include the misuse of BPMN shapes, errors in the definitions of ERs and FPs, invalid mappings between ERs, FPs, and IFC elements, and incorrect representation of IFC structures.

To overcome these problems, we developed an extended Process to Product Modeling, or xPPM, method and tool. The xPPM method provides an approach to integrated PM, ER, and FP modeling that allows IDM developers to easily track and maintain information items specified for a PM. The xPPM method comprises a PM module, which uses a subset of BPMN, and ER and FP specification modules, which use an XML schema as a basic data structure. The subset of BPMN for IDM development was derived by analyzing the BPMN shapes most commonly used in existing IDM PMs. A set of 22 BPMN shapes was selected as an essential BPMN subset for IDM development. The XML schemas for ERs and FPs were developed according to the requirements of the international IDM standard [17]. The FP specification module includes an IFC Menu. An IFC Menu is a module that enables users to easily navigate through IFC elements and also to add new user-defined elements in the context of an IFC element. The potential benefits and limitations of xPPM were analyzed through a process involving the reproduction of existing IDMs. The major findings were as follows:

a. The PM modeling process is facilitated and made less error-prone by providing only the essential subset of BPMN to IDM developers as well as a BPMN rule checker.

b. The use of XML schemas and the xPPM tool enforced valid mapping between ERs and FPs, as well as between FPs and IFC elements.
c. By specifying ERs and FPs in XML, the existing ERs and FPs could be easily read and reused in IDMs. Likewise, ER and FP documents could be generated automatically from ERs or FPs saved as XML files. This opens the future possibility of sharing ERs and FPs among international IDM developers through integrated online IDM storage.

d. The misrepresentation of IFC structures as well as typographical errors and other editorial problems regarding IFC elements could be eliminated using the IFC Menu.

e. One of the ultimate challenges in information modeling is how to guarantee the completeness of the specified information. That is, will the MVD developed through the IDM/MVD process include all the information items required by an intended data exchange scenario? Although it may not be possible to safeguard against this issue altogether, concerns about completeness may be mitigated by incorporating existing methods to guarantee the completeness of the specified information, including the “dynamic information consistency checking method” [24,29] and the “concept-based valid IFC subset extraction method” [22], into xPPM.

Several problems that we, the IDM/MVD community, should consider have been identified, and suggestions for changes have been made:

a. Even PM specification using only 22 BPMN shapes still leaves potential for confusion between similar shapes (e.g., message flow and data association) and, eventually, the misuse of shapes. Because the purpose of a PM is to provide a context for data exchange and not to specify complex and subtle business rules, a more straightforward and simple notation can be developed and adopted in the future.

b. ERs and FPs are intended to be reused instead of redefined. However, it is very difficult to understand and know all the ER and FP content specified by others. It may be practical to restrict the reuse of ERs and FPs within a single IDM only.

c. Currently, an MVD is developed as a separate effort from IDM development. However, the FP structure is so complex that FPs are already very close in structure to the structure of IFC. Consequently, a great many FP and MVD specification efforts overlap. Furthermore, if a new version of IFC is released, then a new MVD should be developed for the new version of IFC. The amount of time and effort that goes into developing an MVD is not any less than the time and effort required to update FPs. The old FP definitions can be kept as FPs for the older versions of IFC. The xPPM method proposes to eliminate the distinction between IDM and MVD and thereby integrate the efforts.

d. Currently, no detailed rules for specifying information items are provided. Several rules regarding the use of “OR” in information item lists, designating mandatory and optional constraints, and protecting synonymous concepts are proposed herein.

The proposed xPPM method aims to be a catalyst for delivering practically deployable IDMs within an acceptable timeframe. The method is evolving toward this goal.

Acknowledgements

This research was supported by the Technology Innovation Program (World Best Software Project, 10040794) of the Ministry of Science, ICT & Future Planning (MSIP, Korea), and also partially by the Science, ICT & Future Planning, Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (NIPA-2013-H4041-13-1003) supervised by the NIPA (National IT Industry Promotion Agency).

References


