Hybrid Model for Developing BIM Software Extensions

Jesus N. Zaragoza-Grife, Romel G. Solis-Carcaño, and Gilberto A. Corona-Suarez
Faculty of Engineering, Universidad Autonoma de Yucatan, Merida, Mexico
Email: { zgrife, tulich, csuarez}@uady.mx

***Abstract*—The use of Building Information Modelling (BIM) has become more popular lately. In the beginning, BIM was considered an alternative for 3D building drawings with the innovation of doing so in a parametric fashion. Nowadays, using BIM in design and construction seems to be unlimited. Since the first versions of the software for BIM, users are allowed to develop extensions in order to customize and enhance these tools, using their corresponding Software Development Kit (SDK). In this paper, a model for developing extensions on Autodesk Revit is presented. This model uses a hybrid approach which combines isolated external commands that are linked to a controller class that exposes all the extension functionality. A class diagram was created which depicts classes and associations of the model. Finally, the process at runtime on interaction between Revit and extension is described. The proposed model allows the development of an extension that gives several operations through the Revit user interface.**

***Index Terms*—BIM, 3D modeling, Autodesk Revit, Revit extension, Revit user interface**

I. Introduction

The design and construction of a building is a team activity, in which each type of specialty is supported and augmented by its own computer applications. Given the diversity of contractor types, it is not surprising that there is a wide range of processes and tools currently in use across the industry. Larger firms typically use computer-based systems for almost all of their key work processes, including: estimating, construction planning, scheduling, cost control, accounting, procurement, supplier and vendor management, marketing, and so forth [1]. For tasks related to the design, such as estimating, coordination and scheduling, paper plans and specifications are the typical starting point, even if the architect used 2D or 3D CAD systems for the design. These require contractors to manually perform quantity takeoffs to produce an accurate estimate and schedule, which is a time-consuming, tedious, error-prone, and expensive process. For this reason, cost estimates, coordinated drawings, and detailed schedules are often not performed until late in the design process. Perhaps even more important, the contractor is not involved during the design process and is not able to offer suggestions that would reduce costs without sacrificing quality and sustainability.

Fortunately, this methodology is beginning to change, as contractors are recognizing the value of Building Information Modelling (BIM) for project team collaboration and construction management. According to the National Building Information Modeling Standard (United States) a Building Information Model (BIM) is "a digital representation of physical and functional characteristics of a facility. As such it serves as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its lifecycle from inception onward." [2]

II. The Use of BIM Extensions

The use of Building Information Modelling (BIM) has become more popular recently. At the beginning, BIM was considered an alternative for 3D building drawings with the innovation of doing so in a parametric fashion. A building information model features computable building information that enables a model to be understood by a computer as a building [3]. A wall for example, "knows" what it is and how to react to the rest of the building. As such, it can be scheduled or quantified as a wall: a building assembly made of real materials.

Nowadays, the use of BIM supports numerous building design and construction activities. By using BIM tools, architects are potentially able to provide models earlier in the procurement process that contractors can use for estimating, construction planning, coordination, fabrication, procurement, and other functions [1]. For instance, one of the key processes in which BIM can be used to support the planning of construction projects is the generation of accurate quantity take-offs and cost estimates throughout the lifecycle of a project. This process allows the project team to see the cost effects of their changes, during all phases of the project, which can help curb excessive budget overruns due to project modifications. Specifically, BIM can provide cost effects of additions and modifications, with potential to save time and money and is most beneficial in the early design stages of a project [4]. In addition, when preparing their cost estimates, estimators typically begin by digitizing the architect's paper drawings, or importing their CAD drawings into a cost estimating package, or doing manual takeoffs from their drawings. This introduces the potential for human error and propagates any inaccuracies

Manuscript received May 5, 2014; revised July 31, 2014.

doi: 10.12720/joams.3.3.227-232

there may be in the original drawings [3]. By using a building information model instead of drawings, the takeoffs, counts, and measurements can be generated directly from the underlying model. Therefore the information is always consistent with the design. And when a change is made in the design – a smaller window size, for example – the change automatically ripples to all related construction documentation and schedules, as well as all the takeoffs, counts, and measurements that are used by the estimator. Estimators usually invest a significant amount of time on such kind of quantification, thus the huge advantage of using a building information model for cost estimating can be appreciated. When manual takeoffs are not required, time, cost, and human errors can be saved. By automating the tedious task of quantifying, BIM allows estimators to use that time instead to focus on higher value project-specific factors – identifying construction assemblies, generating pricing, factoring risks, and so forth – that are essential for high quality estimates [3]. Some other potential benefits of using BIM for preparing quantity take-offs and cost estimates include [4]:

- Precisely quantify modeled materials;
- Quickly generate quantities to assist in the decision making process;
- Generate more cost estimates at a faster rate;
- Better visual representation of project and construction elements that must be estimated;
- Provide cost information to the owner during the early decision making phase of design and throughout the lifecycle, including changes during construction;
- Saves estimator's time by reducing quantity take-off time;
- Allows estimator's to focus on more value adding activities in estimating such as: identifying construction assemblies, generating pricing and factoring risks, which are essential for high quality estimates;
- Added to a construction schedule (such as a 4D Model), a BIM developed cost estimate can help track budgets throughout construction;
- Easier exploration of different design options and concepts within the owner's budget;
- Quickly determine costs of specific objects.

Furthermore, BIM provides a framework to assess the feasibility of a project and the possibility of evaluating performance according to the current codes for design and construction, taking into account functionality and constructability [5]. Even the precast industry is using BIM for the automation of production processes with satisfactory results Bernstein [6].

A basic premise of BIM is collaboration by different stakeholders at different phases of the lifecycle of a facility to insert, extract, update, or modify information in the BIM to support and reflect the roles of that stakeholder. The BIM is a shared digital representation founded on open standards for interoperability. [2]

Interoperability is the ability to pass data between applications, and for multiple applications to jointly contribute to the work at hand [1]. Interoperability, at the minimum, eliminates the need to manually copy data already generated in another application. As mentioned before, manual copying of partial project data greatly discourages iteration during design, as required for finding best solutions to complex issues, such as structural or energy design. It also leads to errors, since manual copying inevitably leads to some level of inconsistency. It is also a great restriction to the automating of business practices.

That is why, software companies such as Autodesk, Bentley, Graphisoft, Tekla, among others, as well as very important participants involved in construction such as designers, contractors, building owners and academic researchers are willing to work for BIM technology extension. The intention of their work is to use BIM in other related project life cycle processes and not only for design visualization purposes [7]. For example, beside the capability to support geometry and material layout, BIM can support project scheduling, cost estimating and cost control are examples of using BIM extensions [2]. Operation and maintenance of buildings and facilities are other processes where BIM makes a significant impact [8].

Since the first versions of the software for BIM, users have been allowed to develop extensions in order to customize and enhance the functionality of these tools, using their corresponding Software Development Kit (SDK). As an example of this, Autodesk allows it in most of their software packages.

This paper features a model for developing extensions on Autodesk Revit, which is a purpose-built BIM solution that is commercially available worldwide. Actually, Revit was used in the development of this model due to its presence in the global market and its relative ease of use. Besides, there is a significant amount of information about the development of extensions for Revit over the internet. For example, there is blog called The Building Coder by J. Tammik, which provides a comprehensive set of code samples to develop Revit extensions [9].

The proposed model uses a hybrid approach which combines isolated external commands that are linked to a main controller class that exposes all the extension functionality.

III. BIM Interoperability

In order to understand the contribution of the proposed model it is necessary to highlight that no BIM tool provides the full capabilities of a specialized application such as an estimating package for example [1]. Thus it is necessary the interoperability of the selected BIM tool with other specialized application in order to leverage the information and data that the building information model contains. For example estimators can use a variety of options to leverage BIM for quantity takeoff and to support the estimating process. Three primary options are:

A. Export Building Object Quantities to Estimating Software

Most BIM tools offered by software vendors include features for extracting and quantifying BIM component properties. These features also include tools to export quantity data to a spreadsheet or an external database. In the United States alone, there are over 100 commercial estimating packages and many are specific to the type of work estimated [1]. However, surveys have shown that MS Excel is the most commonly used estimating tool [10]. For many estimators, the capability to extract and associate quantity takeoff data using custom Excel spreadsheets is often sufficient. This approach, however, may require significant setup and adoption of a standardized modeling process. To go beyond the use of Excel, one of the following processes is required.

B. Use a Quantity Takeoff Tool

A specialized quantity takeoff tool can be used to import data from various BIM tools. This allows estimators to use a takeoff tool specifically designed for their needs without having to learn all of the features contained within a given BIM tool. Examples of these are: Autodesk QTO, Innovaya, and Vico Takeoff Manager. These tools typically include specific features that link directly to items and assemblies, annotate the model for "conditions," and create visual takeoff diagrams. These tools offer varying levels of support for automated extraction and manual takeoff features. Estimators will need to use a combination of both manual tools and automatic features to support the wide range of takeoff and condition checking they need to perform. Changes to the building model require that any new objects be linked to proper estimating tasks so that accurate cost estimates can be obtained from the building model, depending on the accuracy and level of detail already modeled. [1]

C. Directly Link BIM Components to Estimating Software

This includes using a BIM tool that is capable of linking directly to an estimating package via a plug-in or third-party tool [1]. Many of the larger estimating software packages now offer plug-ins to various BIM tools. These include: Sage Timberline via Innovaya, U.S. Cost, Nomitech, and Vico Estimator. These tools allow the estimator to associate objects in a building model directly with assemblies, recipes, or items in the estimating package or with an external cost database such as R.S. Means [1]. These assemblies or recipes define what steps and resources are needed for construction of the components onsite or for the erection or installation of prefabricated components. Assemblies or recipes often include references to the activities needed for the construction, for example, place forms, place rebar, place concrete, cure, and strip forms. The estimator is able to use rules to calculate quantities for these items based on the component properties or manually enter data not extracted from the building information model. The assemblies may also include items representing necessary resources such as labor, equipment, materials, and so forth and associated time and cost expenditures. As a result, all information required to develop a complete cost estimate and detailed list of basic activities can be used for construction planning. If this information is related to the BIM components, it can be used to generate a 4D model. The graphic model can also be linked to the estimate to illustrate the model objects associated with each line item within that estimate. This is very helpful for spotting objects that have no cost estimate associated with them.

The model proposed in this paper is based on the third approach discussed before. Such approach works well for contractors who have standardized on a specific estimating package and BIM tool. Integrating BIM component information from subcontractors and various trades, however, may be difficult to manage if different BIM tools are used. There are clear benefits to this highly integrated approach, but one potential shortcoming is the need for the contractor to develop a separate model. If the project team is standardized on a single software vendor platform, this method may be suitable. This may require either a design-build approach or a contract that integrates the main project participants from the beginning of the project (e.g. Integrated Project Delivery). [1]

IV. Methodology

There are a variety of ways of getting data (e.g. quantities and material definitions) out of a Revit building information model into a specialized computer application (e.g. cost estimating system). The Application Programming Interface (API) is one the main integration approaches that allows a direct link between the specialized computer application and Revit [3]. For example, from within Revit, a user can export the building model using costing program's data format and sends it to the estimator, who then opens it with the costing solution to begin the costing process. Commercially available estimating programs from vendors such as U.S. COST or Innovaya have this capacity. [3]

A. RevitAPI SDK

Revit can be extended by using its application programming interface (API) through three different kinds of projects which have different capabilities and goals [11]. In this work a combination of two of them is used: the external application and the external command.

The Revit API consists of two main assembly files which are class libraries to be used with the Microsoft NET Framework: RevitAPI.dll and RevitAPIUI.dll. The first assembly exposes classes that allow developers to manage the elements of the BIM model at a database-level. On the other hand, the second assembly provides classes for managing the BIM model, but from the user interface (UI). These classes allow instantiation of menu bars, panels, buttons, text boxes, etc. which can be attached to the Revit main menu. It is important to mention that there is a strong relationship between both assemblies.

Unified Modeling Language (UML) class diagram is

showed in Fig. 1, which describes the interfaces provided by Revit API to develop extensions for external commands and external applications, respectively.

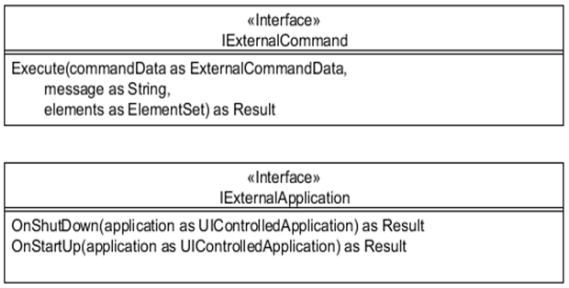

Figure 1. UML class diagram of the Revit API interfaces for developing extensions.

Fig. 1 shows that, unlike *IExternalCommand* interface which has only a function called *Execute*, the other interface *IExternalApplication* has two functions: *OnShutdown* and *OnStartup*. The passed argument in the *Execute* function has three variables: *CommandData* of *ExternalCommandData* type, *message* of *string* type, and finally *Elements* of *ElementSet* collection type. In *OnShutdown* and *OnStartup* a single parameter called *application* is passed. The *application* parameter is *UIControlledApplication* type, which returns a reference to Revit UI.

Revit uses the document approach for its BIM model database files. Such files have RVT extension as a naming convention. In Revit, generally BIM models are composed of multiple instances of elements, each having different attributes and behaviors. The domain model of Revit uses a class called Element. From this, other super-specialized classes are derived to model the various objects that make up the BIM model such as walls, beams, columns, slabs, etc. This architecture allows a certain level of abstraction which helps to understand the Revit BIM model as a set of instances of Element type class.

B. Revit UI Items

Revit UI allows the user to interact with a set of commands arranged so that the main menu is divided into several command containers called *RibbonTab* and each *RibbonTab* groups several *RibbonPanel*. Each *RibbonPanel* container can group several *RibbonItems*. Fig. 2 shows the UML class diagram for UI menus used in Revit. It is important to note that *RibbonItems* are the most general case of a menu item in Revit. Its implementation is done by *RibbonItem* derived classes such as *PushButton*, *PulldownButton*, etc. In order to execute a function of the extension, each menu item must be associated with a class that implements the interface *IExternalCommand*.

For the final proposed model a UML class diagram was created. Such diagram depicts classes and associations used in the proposed model. Finally, the process at runtime on interaction between Revit and extension is described.

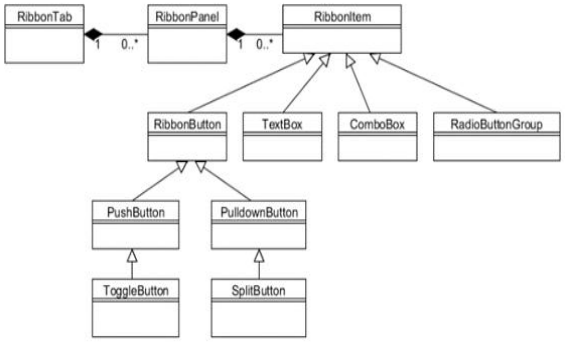

Figure 2. UML class diagram of the Revit menu.

V. Results

A. The Proposed Extension Model

For the development of Revit extensions, a hybrid model formed by a controller class which implements the interface *IExternalApplication* and a linked set of several classes that implement the interface *IExternalCommand* is presented. The controller class exposes the functions that the extension would be able to perform. For each function in the controller class you want to make available to the user via the UI in Revit, a class that implements the interface *IExternalCommand* must be created. These classes must be associated with menu items Revit UI to access the corresponding function in the controller class. Fig. 3 shows the proposed hybrid model through a UML class diagram.

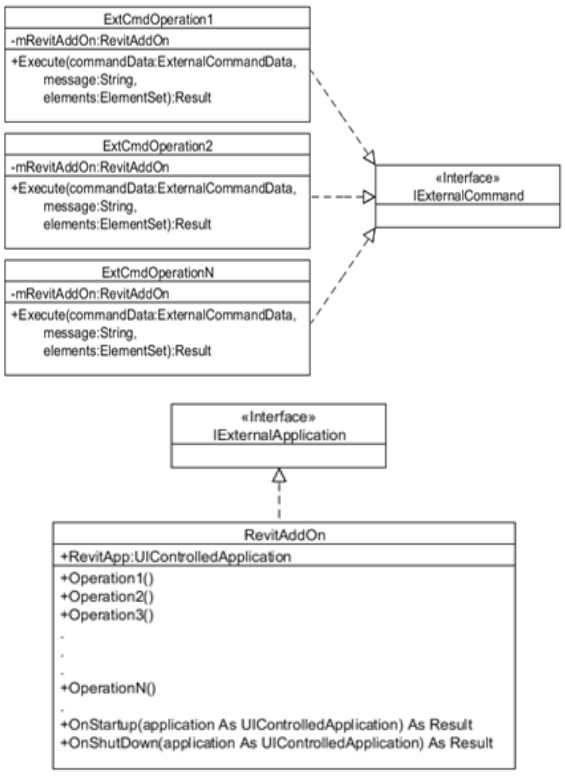

Figure 3. Proposed hybrid model UML class diagram.

B. Runtime Process

At runtime, the extension consists of two basic scenarios that are user-initiated: launching a new instance of Revit and requesting the execution of a command of the extension.

Launching a new instance of Revit: When an instance of Revit is launched, it loads all extensions declared in the configuration file which are implementing the *IExternalApplication* interface. The configuration file consists of a XML file located in the path C:\ProgramData\Autodesk\Revit\Addins\2014. In this file identification information for the extension is provided. Among other parameters as shown in Fig. 4, the most important are: <AddIn Type>, <FullClassName> and <Assembly>.

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Application">
      <AllowLoadingIntoExistingSession>No</AllowLoadingIntoExistingSession>
      <ClientId>2134F622-EB0A-4829-A9C3-9194B77A7FE4</ClientId>
      <Text>RevitAddOn</Text>
      <Name>RevitAddOn</Name>
      <FullClassName>RevitExt.RevitAddOn</FullClassName>
      <Assembly>C:\Program Files\RevitAddOn\AddIn\RevitAddOn.dll</Assembly>
      <VendorId>VendorId</VendorId>
      <VendorDesc>Vendor Description</VendorDesc>
  </AddIn>
</RevitAddIns>
```

Figure 4. RevitAddOn.AddIn XML configuration file

When Revit loads an extension, it fires the OnStartUp event. Through this event the Revit instance passes its own reference in a variable of *UIControlledApplication* type on the event. This allows the controller class to know what instance of Revit invoked it. The reference to the instance of Revit which invoked the extension is stored as a property in the controller class called *RevitApp*. A sequence diagram for this scenario is shown in Fig. 5.

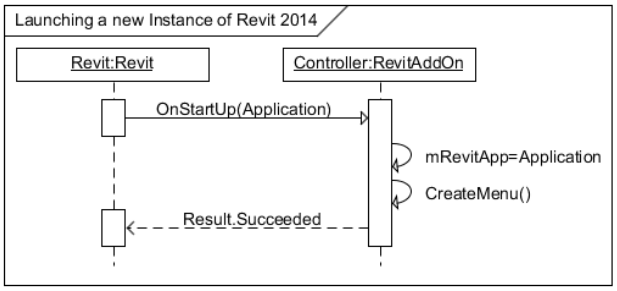

Figure 5. Sequence diagram of launching a new instance of Revit 2014

After that, the controller class has the responsibility to create the menu items and associate them with the respective class that implements an interface *IExternalCommand*. This must be done in the *OnStartup* event of the main controller of the extension.

Creating the UI menus involve the use of helper class that exposes a set of functions called AddPushButton, AddTextBox, AddToggleButton, etc. The source code for these functions implemented in a helper class and their use are deeply discussed in [11].

Requesting the execution of a command of the extension: Furthermore, the extension functionality is exposed by the classes that implement the interface *IExternalCommand*. As already mentioned, each class maintains a reference to the main controller of the extension. When the user requests the execution of an available command from the Revit UI and Revit calls to the *Execute* method of the associated command class, then the inner code in the *Execute* method calls the corresponding function exposed by the controller class.

In order to call the corresponding extension function, the reference to the main controller is needed. This reference is obtained through a search in the collection of external applications loaded by Revit. The reference to this collection is obtained from *commandData* argument, which was passed in the *Execute* method of the class implementing *IExternalCommand* interface. The full path to this collection is:

CommandData.Application.LoadedApplications.

A sequence diagram of this scenario is shown in Fig. 6.

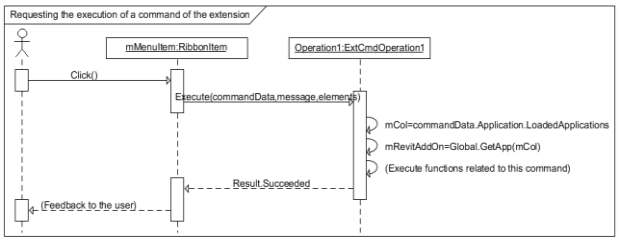

Figure 6. Sequence diagram of requesting the execution of a command of the extension

Finally, the proposed model allows the developer to combine the tools provided Revit to create an extension that works as a set of linked commands instead of working as proposed in the isolated command interface implementation *IExternalCommand*.

VI. Conclusions

The proposed model allows the development of an extension that exposes several operations that can be performed through the Revit UI.

It is possible to extend the model for accessing a database that could hold additional data which is managed in a parallel fashion to the BIM model in Revit.

In huge BIM models, this kind of extension could help to manage additional data that is not in the Revit model.

However, this way of working would lead to an additional challenge for keeping synchronization between the BIM model and a database.

References

[1] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, 2nd Ed. New Jersey: John Wiley & Sons, 2011.

[2] *United States National Building Information Modeling Standard, Version 1 – Part 1: Overview, Principles, and Methodologies*, National Institute of Building Sciences, December 2007.

[3] *BIM and Cost Estimating, Autodesk Revit*, Press: Autodesk, Sept. 11, 2008.

[4] *BIM Project Execution Planning Guide*, Version 2.1, The Computer Integrated Construction Research Program at the Pennsylvania State University, May 2011.

[5] S. Azhar, A. Nadeem, J. Mok, and B. Leung, "Building information modeling (BIM): A new paradigm for visual interactive modeling and simulation for construction projects," in

Proc. First International Conference on Construction in Developing Countries, Karachi, Pakistan, 2008.
[6] H. Bernstein, J. Gudgel, and D. Laquidara-Carr, "Prefabrication and modularization: Increasing productivity in the construction industry," *Smart Market Report*, McGraw Hill Construction, 2011.
[7] H. Pentilläj, "Describing the changes in architectural information technology to understand design complexity and free-form architectural expression," *Journal of Information Technology in Construction*, 2006.
[8] H. S. Kim, H. S. Moon, S. Y. Kim, and L. S. Kang, "Utilization of visual object information for civil engineering BIM implementation in construction lifecycle stages," in *Proc. 2nd International Conference on Education and Management Technology IPEDR*, vol. 13, 2011.
[9] J. Tammik, *The Building Coder: Blogging of the Revit API*, 2013.
[10] T. Sawyer and T. Grogan, "Finding the bottom line gets a gradual lift from technology," *Engineering News Record*, August 12, 2002.
[11] D. Rudder, *Instant Autodesk Revit 2013 Customization with, NET How-to*, Birmingham: Packet Publishing, 2013, pp. 70.

Jesus N. Zaragoza-Grife is a full-time associate professor at the Faculty of Engineering in Universidad Autonoma de Yucatan, Mexico. He obtained his BSc degree in Civil Engineering and MSc degree in Construction Engineering at Universidad Autonoma de Yucatan. His research work includes topics related to cost engineering, scheduling, as well as software development for construction management and engineering.

Romel G. Solis-Carcaño is a full-time professor at the Faculty of Engineering in Universidad Autonoma de Yucatan, Mexico. He obtained his BSc degree in Civil Engineering and MSc degree in Construction Engineering at Universidad Autonoma de Yucatan. His research work includes different topics related to construction management and engineering, as well as materials engineering.

Gilberto A. Corona-Suarez is a full-time associate professor at the Faculty of Engineering in Universidad Autonoma de Yucatan, Mexico. He obtained his BSc degree in Civil Engineering at Instituto Tecnologico de Merida, Mexico; his MSc degree in Construction Engineering at Universidad Autonoma de Yucatan; and his PhD degree in Construction Engineering and Management at the University of Alberta in Canada. His research work includes several different topics related to construction engineering and management, such as quality management, sustainable construction, and building information modelling.

