CONTEXTUAL INFORMATION DELIVERY ON CONSTRUCTION SITES

Hiam Khoury
American University of Beirut, Beirut, Lebanon
hiam.khoury@aub.edu.lb

Vineet Kamat
University of Michigan, Ann Arbor, Michigan, USA
vkamat@umich.edu

Abstract

The information-intensive nature of construction projects requires site personnel to have on-demand access to several pieces of construction project data (plans, drawings, schedules, specifications, etc). This paper evaluates the capability of the CIMsteel Integration Standards (CIS/2) product model, and project information systems such as Microsoft Access databases for improving traditional, and manual information retrieval processes by automatically retrieving contextual project information through continuous tracking of mobile users (i.e. engineers, managers, and inspectors). Related experiments were conducted at the National Institute of Standards and Technology (NIST) outdoors at the structural steel test bed structure on the main campus, and in the Structural Engineering Laboratory at the University of Michigan. Results highlighted the potential of contextual information access from the aforementioned information repositories in saving time and improving work productivity on construction sites.

KEYWORDS: construction, information retrieval, BIM, databases

INTRODUCTION

The unstructured and dynamic nature of a construction site and the on-site work complexities necessitate the development of intelligent techniques to support the tasks of on-site construction personnel. Standard interoperable product models and databases have been proposed to facilitate the exchanging, sharing and delivery of pertinent construction information among project participants, as well as enhancing work efficiency and collaborative processes (Halfawy et al., 2001 and Aziz et al., 2005). This paper evaluates the capability of the CIMsteel Integration Standards (CIS/2) product model (Lipman and Reed 2003), and project information systems such as Microsoft Access databases (Allison and Berkowitz 2005) for improving traditional, and manual information retrieval processes and automatically retrieving contextual information and presenting it to construction site personnel.

INFORMATION RETRIEVAL PROCESS FROM CIS/2 PRODUCT MODELS

In this section, the utility of the interoperable CIS/2 product model (Lipman and Reed, 2003) is presented as a suitable data structure to represent cross-referenced building data for the

evaluation of the designed automated information retrieval process. As a mobile user is moving on the jobsite, relevant information on visible and identified entities at a particular instant in time can be retrieved from the underlying product model. This consists in gathering as-built information from the ambient environment and then cross-referencing recognized construction entities with an existing CIS/2 building product model for automated information support on construction sites. CIS/2 deals with information about the steel structure throughout its analysis, design, detailing, and fabrication life cycle (Reed, 2002). The geometry of the structure is only one property of the product data model. Other attributes of the structure include how parts are combined into assemblies and structures, analysis loads and reactions, material types, connection details, associations between members and drawings, and modification history. Figure 1 shows a sample of a CIS/2 file illustrating how steel parts are combined into assemblies and structures (Kamat and Lipman, 2006).

```
#43= LOCATED_PART(92,'92','brace',#42,#33,#20);
  #42= (COORD_SYSTEM('','Part CS',$,3)
      COORD_SYSTEM_CARTESIAN_3D(#40)COORD_SYSTEM_CHILD(#18));
    #40= AXIS2_PLACEMENT_3D('Part axes',#34,#38,#36);
      #34= CARTESIAN_POINT('Part origin',(0.,0.,0.));
      #38= DIRECTION('Part z-axis',(0.,0.,1.));
      #36= DIRECTION('Part x-axis',(1.,0.,0.));
    #18= COORD_SYSTEM_CARTESIAN_3D('','Assembly CS',$,3,#17);
      #17= AXIS2_PLACEMENT_3D('Assembly axes ',#11,#15,#13);
        #11= CARTESIAN_POINT('Assembly origin ',(720.,540.,120.));
        #15= DIRECTION('Assembly z-axis ',(-0.37139068,0.,0.92847669));
        #13= DIRECTION('Assembly x-axis ',(0.92847669,0.,0.37139068));
  #33= (PART(.UNDEFINED.,$)PART_PRISMATIC()PART_PRISMATIC_SIMPLE(#21,#26,$,$)
      STRUCTURAL_FRAME_ITEM(92,'92','brace')STRUCTURAL_FRAME_PRODUCT($)
      STRUCTURAL_FRAME_PRODUCT_WITH_MATERIAL(#27,$,$));
    #21= SECTION_PROFILE(1,'W14X158',$,$,5,.T.);
    #26= POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(258.48791),#3);
      #3= (CONTEXT_DEPENDENT_UNIT('INCH')LENGTH_UNIT()NAMED_UNIT(#1));
        #1= DIMENSIONAL_EXPONENTS(1.,0.,0.,0.,0.,0.,0.);
    #27= MATERIAL(1,'GRADE50',$);
  #20= LOCATED_ASSEMBLY(92,'92','brace',#18,$,#19,#10);
    #18= COORD_SYSTEM_CARTESIAN_3D('','Assembly Coordinate System',$,3,#17);
      #17= AXIS2_PLACEMENT_3D('Assembly axes ',#11,#15,#13);
        #11= CARTESIAN_POINT('Assembly origin ',(720.,540.,120.));
        #15= DIRECTION('Assembly z-axis ',(-0.37139068,0.,0.92847669));
        #13= DIRECTION('Assembly x-axis ',(0.92847669,0.,0.37139068));
    #19= ASSEMBLY_MANUFACTURING(92,'92','brace',$,$,$,$,$,$,$);
    #10= STRUCTURE(1,'cis_2','Unknown');
```

Figure 1: CIS/2 File Structure Example

The file is represented in the Standard for the Exchange of Product model data (STEP) format (ISO 10303:1992, STEP Website, 2008). Each CIS/2 entity instance is assigned a number indicated by a pound (#) sign. The name of the CIS/2 entity then appears in upper case letters. Every entity has a number of fields that contain text strings, numeric values, Boolean values, references to other entities, or null values indicated by a dollar sign. For instance, the #43 CIS/2 entity, together with all its sub-entities, encompasses information about the steel part location within the assembly. The #33 instance refers to information about the steel part properties, such as cross-section, length, and material, and the #20 instance points to the location of the assembly within the structure.

One major step in using CIS/2 in the retrieval and delivery process consisted of using the NIST CIS/2 to Virtual Reality Modeling Language (VRML) translator (Lipman, 2002) to convert CIS/2 files to their corresponding VRML representation. The intermediate

conversion of CIS/2 files to VRML was adopted for the following reasons (Kamat and Lipman, 2006). First, the VRML translator provides a visual interface to the underlying CIS/2 data thereby providing a means to visualize the represented steel structure in a 3D virtual world. Second, implementing a VRML file parser that can traverse the described scene graph to extract member information is relatively straightforward compared to the implementation of a full-fledged CIS/2 file parser. Thus, the parsing and interpretation of CIS/2 information contained in a converted VRML file, yield information for each steel member contained in the represented structural steel frame, including but not limited to the name and the geometry of each member. Any CIS/2 file can provide data structures for multiple levels of detail ranging from frames and assemblies to nuts and bolts. Therefore, prior to translating the CIS/2 file to a VRML file, the user can specify what type of information is of interest to him in order to expedite the translation process and only get the needed information. Figure 2 depicts snapshots of the CIS/2-VRML translator showing its different information level options.

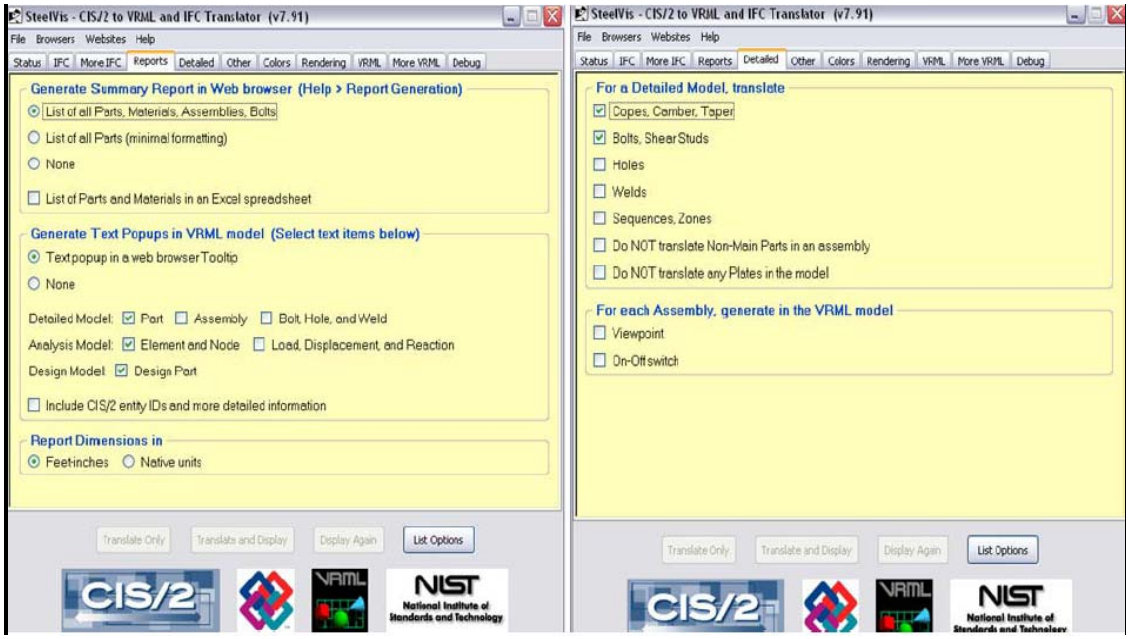

Figure 2: CIS/2-VRML Translator Snapshots

After getting the desired translated VRML file, the next step included using a VRML reader-writer or loader application (Fisher 2005) that automatically parses all of the VRML nodes (together with the attached information), and then translates back to OSG nodes. Information attached to the nodes is then automatically retrieved for each of the detected contextual steel objects.

INFORMATION RETRIEVAL PROCESS FROM MS ACCESS PROJECT DATABASES

A MS Access project database was designed as an option for demonstrating information retrieval capabilities for use in Architecture, Engineering, and Construction (AEC)

applications. The goal is to enable efficient information sharing between different project parties across the life-cycle of a constructed building facility. To achieve this objective, a building was completely described in one coherent model, from which users can extract all the information they need, and to which they can add information they contribute. In this case, where lots of data need to be presented to the user, it was found that it is preferable to separate data and user interface concerns, so that changes to the user interface do not impact the data handling, and that the data can be reorganized without changing the user interface. Therefore, the Model-View-Controller (MVC) architectural pattern (Holub 1999) was adopted to solve this problem and decouple data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller. MVC consists thereby of three different parts: Model, View and Controller (Figure 3).

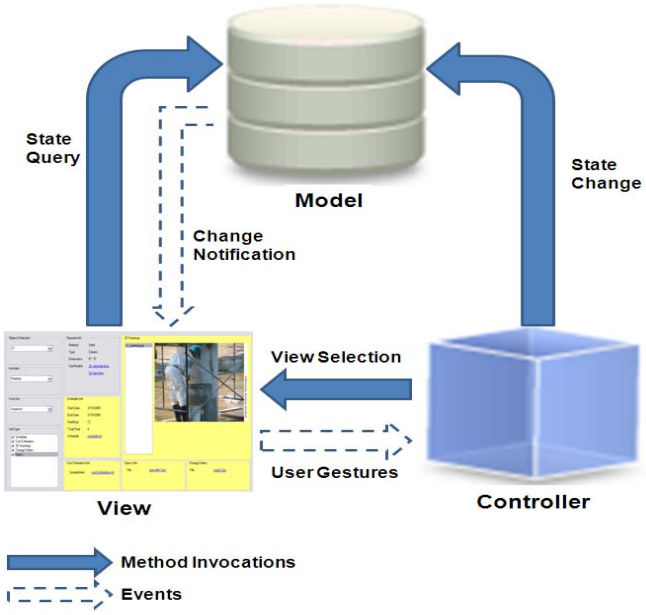

Figure 3: The Model-View-Controller Architecture

The model is the part of the component that encapsulates the application's data structures, i.e. MS Access database tables. It often provides routines to manage and manipulate this data in a meaningful way in addition to routines that retrieve the data from the model. In general, the underlying data access technique should be encapsulated in the model. Therefore, if, for example, the application is to be moved from one system to another system covering a different construction project, the model is the only element that needs to be changed, not the view or the controller. As depicted in Figure 4, the main database tables are 1) *Item* houses the different building objects or structural entities and their attributes like material, type, dimensions, etc., 2) *Activity* takes into account the multilayered nature of a construction *Item* and covers several *Item* construction activities (i.e. concrete pouring, reinforcement, finishing, curing, etc.), 3) *ImgFiles, ItemCadModels, Schedules, Specs, CostEstimation* and *Orders* each reflects a different type of information, and most importantly 4) *ItemtoActivityMapping* maps all information to item-activity pairs.

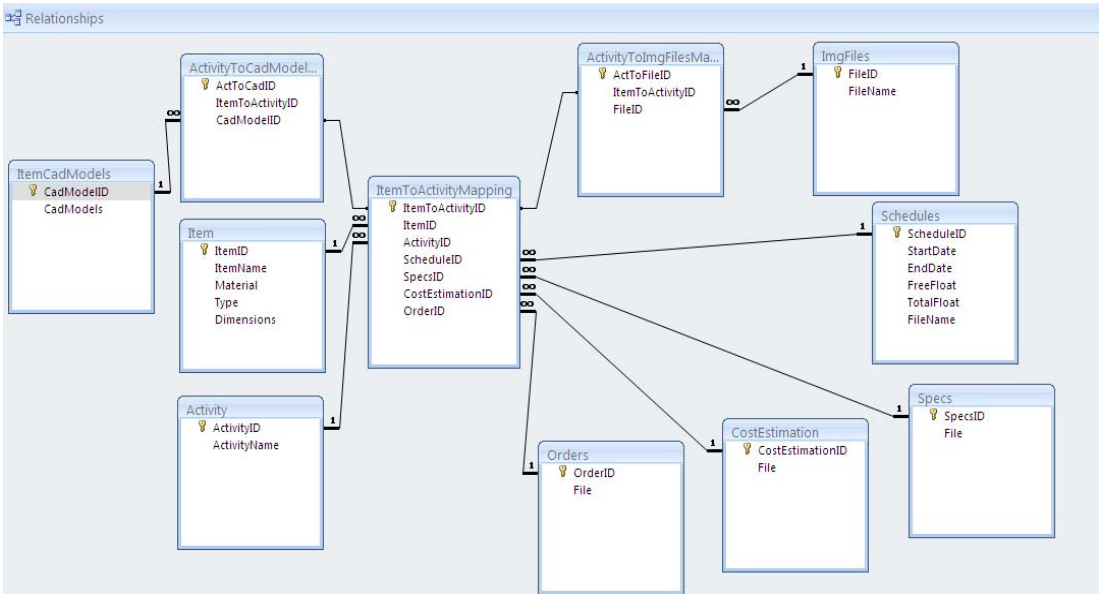

Figure 4: MS Access Database Structure

On the other hand, the controller is responsible for responding to user actions. It determines what request is being made by the user and responds appropriately by triggering the model to manipulate the data and passing the model into the view. In order to interact with the MS Access database and retrieve the requested information, a Microsoft Application Programming Interface (API) called Object Linking and Embedding, Database (OLE DB) (MSDN Website, 2009) was used to define the connection string, display the data source name and connect to the database as follows:

```
connection = new OleDbConnection (@"Provider=
Microsoft.Jet.OLEDB.4.0;
Data Source=.\\db\\InfoRetrievalDB.mdb;Persist Security Info=False");
connection.Open();
```

After connecting to the database, a second step comprised retrieving stored information using methods called queries. Queries provide the capability to manipulate and combine data from multiple tables and place specific conditions on the data retrieved. Most users rely on queries because they only need a certain group of records or fields at any one time, i.e. records that adhere to a specific set of criteria are retrieved. For example, a query can be developed to retrieve general design information about specific contextual items. The query was built and executed by attributing the *ItemName* in the SQL *SELECT* statement to the identified contextual building entity string variable *itemName*:

```
command = new OleDbCommand();
command.Connection = connection;
command.CommandText = "SELECT * FROM Item WHERE ItemName=?";
parameter = new OleDbParameter("ItemName",itemName);
```

A record set was then created as an instance of the *OleDbDataReader* class, item properties obtained using reflection (Sullivan, 2001), all handles closed, and then data retrieved as follows:

```
private OleDbDataReader dataReader;
command.Parameters.Add(parameter);
dataReader= command.ExecuteReader();
DataTable dt = ConvertDataReaderToDataSet(dataReader);
Type tp = typeof(Item);
refController = new ReflectionController(tp);
foreach (DataRow row in dt.Rows)
{
   foreach (DataColumn col in dt.Columns)
      {
       string field = col.ToString();
       string val = row[col].ToString();

      refController.SetProperty(tp.GetProperty(field),val);
      }
}
dataReader.Close();
Item item = (Item)refController.getObj();
```

The last MVC component is the view (Figure 5). It is used to render the data from the model in a manner that is suitable for user interaction. The view pulls data from the model (which is passed to it from the controller) and feeds the data into a template which is populated and presented to the user. The view does not cause the data to be modified; it only displays data retrieved from the model.

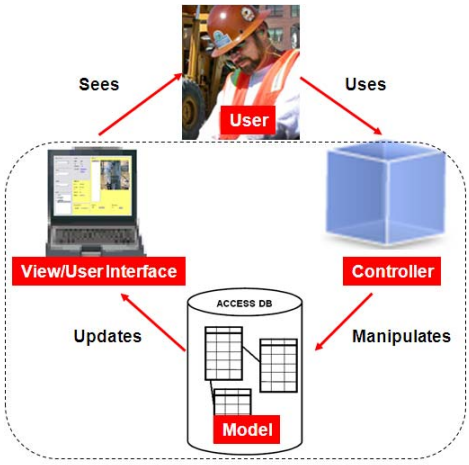

Figure 5: Retrieval Application from a User Point of View

As such, MVC's control flow starts when the user interacts with the user interface in some way (user gestures, i.e. mouse event). Then, the controller handles the input event from the user interface, often via a registered handler or callback. The controller notifies the model of the user action, possibly resulting in a change in the model's state. A view uses the model indirectly to generate an appropriate user interface. The view gets its own data from the model. The model and controller have no direct knowledge of the view. The user interface waits for further user interactions, which restarts the cycle. Therefore, successful use of the MVC pattern isolates business logic (controller) from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other.

VALIDATION RESULTS

This section presents the validation exercises for the information retrieval process, which as defined above, is the ability to extract in real-time, information from product models and databases about contextual construction objects. In order to validate this delivery process, an outdoor structural steel inspection operation and an indoor operation were simulated on NIST's main campus and in the Structural Engineering Laboratory at the University of Michigan respectively.

The goal of the experiment conducted at NIST (Figure 6) was to evaluate whether contextual information identified based on the user's spatial context (Khoury and Kamat, 2009) can be retrieved from a CIS/2 product model.

Figure 6: Steel Structure on NIST Main Campus

In this case, four Ultra Wide Band (UWB) receivers (Multispectral Solutions Website, 2007) and one reference tag were deployed around the area of the steel structure as shown in Figure 7, and one UWB tag and the orientation tracker were mounted on the mobile user who was navigating around the steel structure. As described in (Khoury and Kamat, 2009), the user's position and orientation were continuously obtained from the UWB system and magnetic tracker.

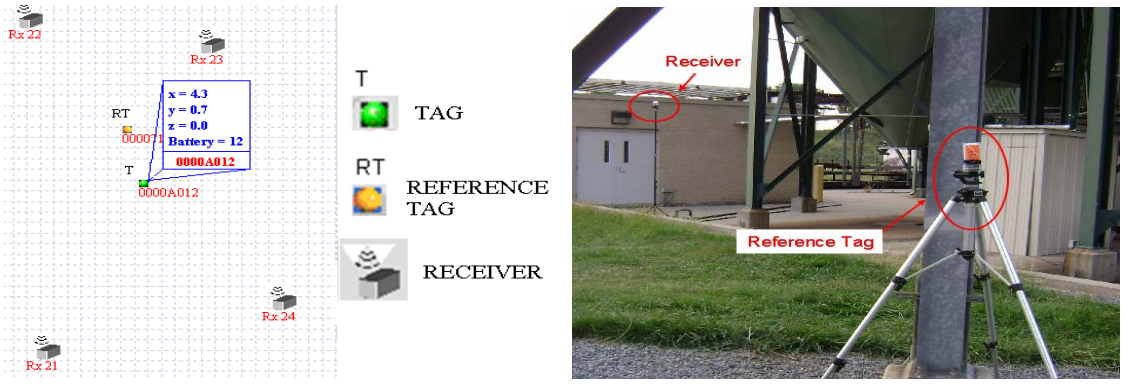

Figure 7: UWB Receivers and Reference Tag Setup at the Steel Structure (NIST)

Figure 8 shows how contextual information on identified specific steel members was automatically retrieved from the converted CIS/2 model.

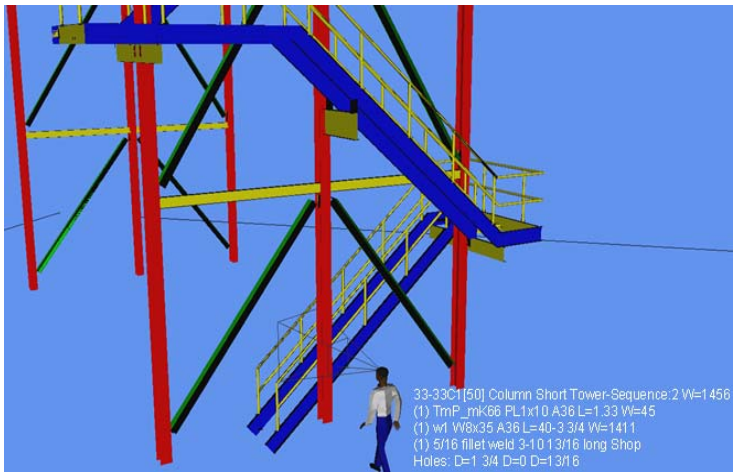

Figure 8: Snapshot of Contextual Information Retrieval of Identified Steel Elements

Another validation experiment, using Ekahau positioning system (Ekahau, 2004) and magnetic orientation trackers, was conducted in the Structural Engineering Laboratory on the first floor of the GGB building at the University of Michigan. The indoor positioning testbed has an area of 40 ft by 25 ft and contains various structural elements including concrete walls and columns, steel columns and beams (Figure 9). As described in (Khoury and Kamat, 2009), the user's position and orientation were continuously obtained from the Ekahau system and magnetic tracker.

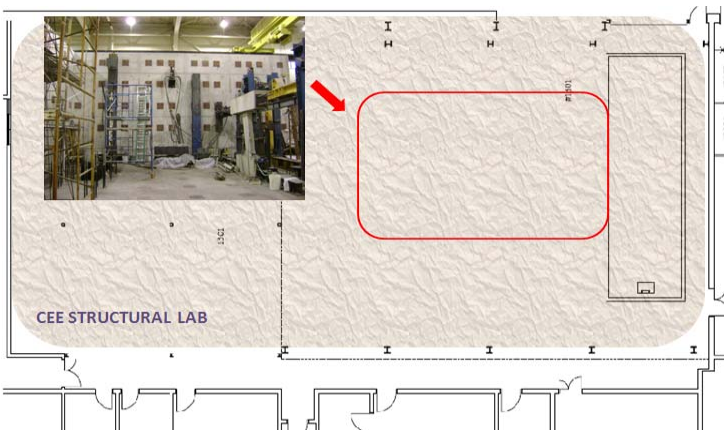

Figure 9: Ekahau Testbed within the Laboratory

Figure 10 represents snapshots of both graphical and textual contextual information retrieval. Upon user's choice, drawing and cost estimation files were retrieved after a specific wall has been detected as visible to the mobile user navigating inside the laboratory.

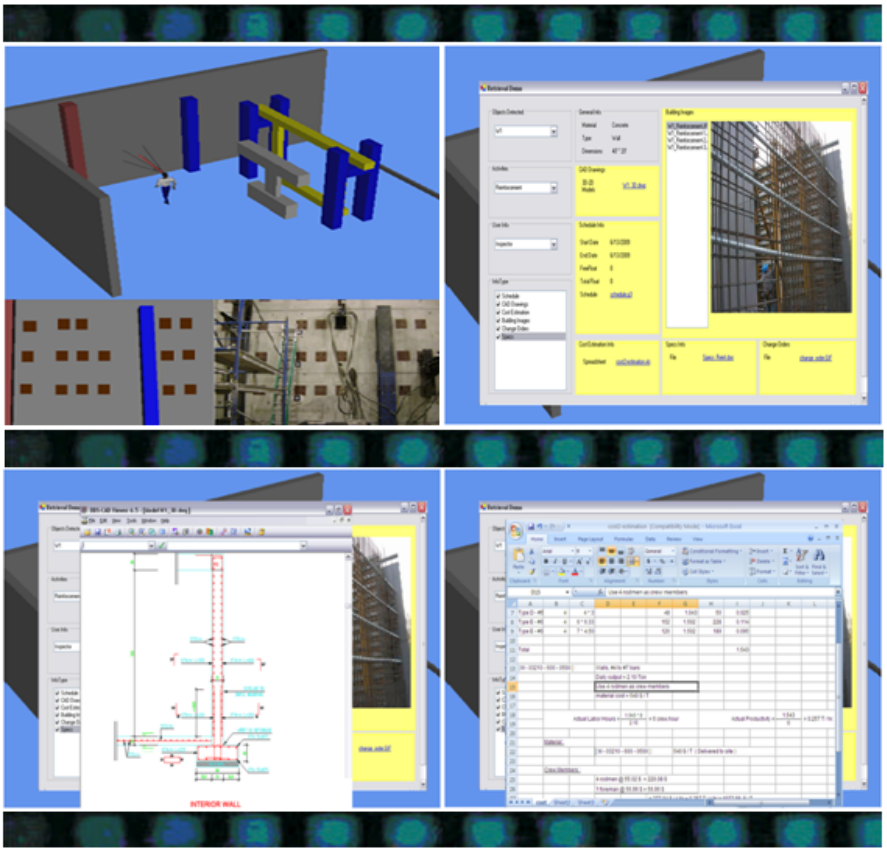

Figure 10: Snapshots of Contextual Information Retrieval (Drawing and Cost Estimation Files)

CONCLUSIONS

The objective of this paper was to explore the means for directly comparing relevant design and construction data in product models and project databases with the construction entities identified to be in a user's field of view at a given time instant, and retrieving contextual information. The authors have successfully evaluated CIS/2 product models and MS Access databases for contextual information retrieval.

In order to demonstrate the feasibility of the adopted information repositories, experiments were conducted at the steel structure on NIST's main campus and in the Structural Engineering Laboratory at the University of Michigan. The experiments results highlighted how contextual information can be readily retrieved and delivered to mobile users in real-time, thereby saving time and improving work productivity in dynamic construction environments.

REFERENCES

Allison, C.L., and Berkowitz, N. (2005), SQL for Microsoft Access, Edition: illustrated, Wordware Publishing, Inc., Plano, TX.

Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo, P.M., and Bouchlaghem, D.N. (2005), Context Aware Information Delivery for on-Site Construction Operations, 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-32.

Ekahau, Wi-Fi Based Real-time Tracking and Site Survey Solutions, available at: http://www.ekahau.com.

Fisher, A. (2005), OSG Loaders/Plugins to the OSG, available at: http://www.openscenegraph.org/projects/osg/wiki/Community/Plugins

Halfawy, M., and Froese, T. (2001), Leveraging information technologies applications in the Canadian AEC/FM Industry, In the Conference Proceedings of the Canadian Society of Civil Engineers, Victoria, BC, Paper A22.

Holub, A. (1999), Building User Interfaces for Object-Oriented Systems, Java World, available at: http://www.javaworld.com/javaworld/jw-07-1999/jw-07-toolbox.html .

Kamat, V.R., and Lipman, R.R. (2006), Evaluation of Standard Product models for Supporting Automated Erection of Structural Steelwork, Journal of Automation in Construction, 16, 232-241.

Khoury, H. M., and Kamat V. R. (2009), Indoor User Localization for Context-Aware Information Retrieval in Construction Projects, Automation in Construction, 18 (4), 444-457.

Lipman, R.R., and Reed, K.A. (2003), Visualization of Structural Steel Product Models, Electronic Journal of Information Technology in Construction, 8, 43-50.

Lipman, R.R. (2002), Mobile 3D visualization for construction, 19th Intl. Symposium on Automation and Robotics in Construction (ISARC), 53–58.

MSDN Website, MSDN Library, available at: http://msdn.microsoft.com/en-us/library.

Multispectral Solutions, Inc. Website, Sapphire DART System, available at: http://www.multispectral.com/products/sapphire.htm

Reed, K.A. (2002), Role of the CIMsteel integration standards in automating the erection and surveying of constructional steelwork, 19th International Symposium on Automation and Robotics in Construction (ISARC), 15–20.

Sullivan, G. T. (2001), Aspect-Oriented Programming using Reflection and Metaobject Protocols, Communications of the ACM, 44(10), 95-97.