AUTOMATING THE LIFT DESIGN PROCESS OVER THE WEB

Georgios Andreadis[1], Konstantinos D. Bouzakis[1], Pavlos Katsonis[2], Evangelos Rousoudis[3] & Lazaros Asvestopoulos[3]

[1] Aristotle University of Thessaloniki–Greece, Department of Mechanical Engineering, 541 24, Thessaloniki, Greece
[2] Aristotle University of Thessaloniki-Greece, Department of Informatics, 541 24, Thessaloniki, Greece
[3]Kleemann Hellas SA, Greece

Corresponding author: Georgios Andreadis, andreadi@eng.auth.gr

Abstract: Lift manufacturing often requires human intervention in tasks that could be fully automated. Aim of this study is to automate the task of creating a lift's plans, after its characteristics have been defined. This goal is achieved through the development of a software system, acting as a server that accepts data and produces lift plans. Inbound data defining the lift is received from a rule-based software system, using XML and ASP.NET technology. The plans are produced by controlling an existing commercial CAD application, through an add-in developed especially for the CAD application, in C#. The result is the fully automated production of construction plans for the lift to be manufactured.
Key Words: lift, elevator, automated design, CAD

1. INTRODUCTION

Computer Aided Design software has significantly lowered the difficulty of creating technical designs, to the extend that few drafters are needed any more. Their work is usually done by the engineers through CAD. Nevertheless, a lot of work time is spent on designing plans and at issue lift plans. The current research focuses on developing a framework for the full automation of the design of lifts (Wu et al., 2007).
In Figure 1 the operational model of an interactive ordering system is shown.
According to the architecture, a rule based software helps the designer to precisely define the lift. The description of the lift is forwarded to a server, through a web service. The server is responsible for controlling a commercial CAD application into producing the final lift plans, completely automatic. The designer simply gets the plans in a matter of seconds (Myers et al., 1996).
In order to create the automation process, a framework for defining each lift type was developed. It supports the object oriented approach of the lifts.
Relative research has been done (Daum & Merten, 2003), about the automation of gating systems for die-casting die. Research in the lifts field about the current topic is new.

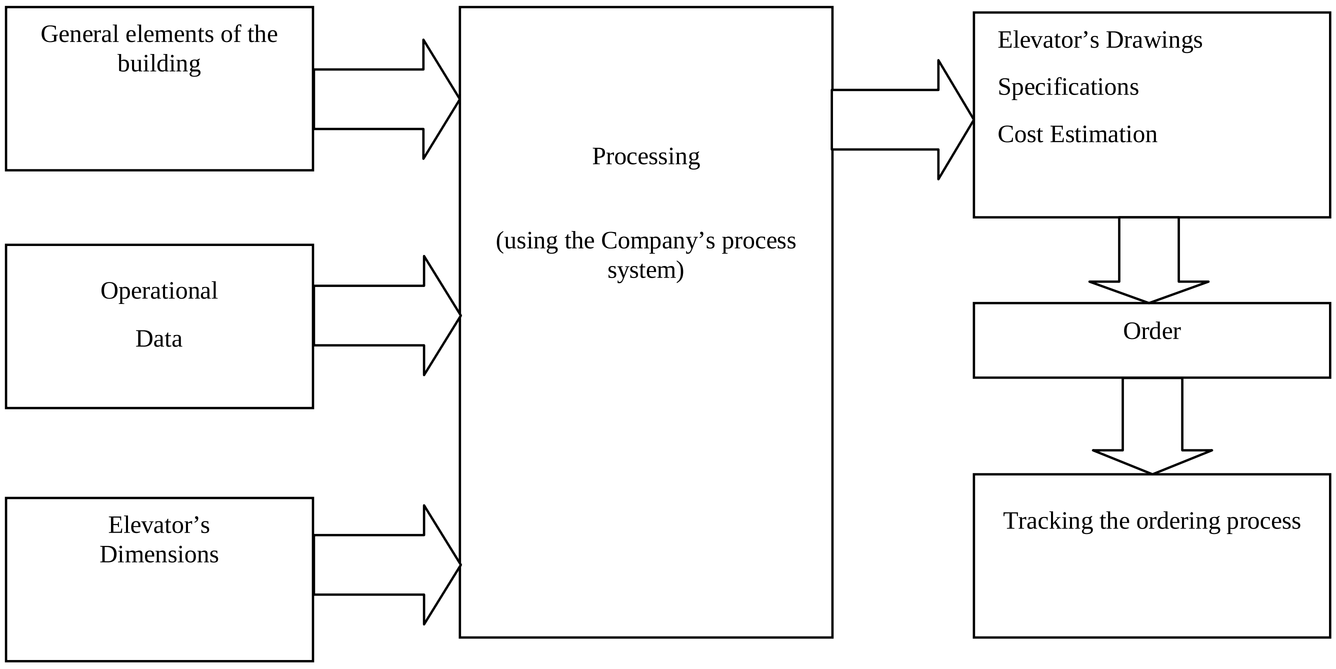

Fig. 1: Operational model of an interactive ordering system

2. SYSTEM'S ARCHITECTURE

Figure 2 illustrates the arrangement of the various modules composing the automation system. Inbound data is interpreted by the Designer module, which in turn controls the CAD platform in designing the lift plans, using a database of pre-made design blocks. After the automatic drafting operation is completed, the resulting lift plans are returned to the calling service or user.

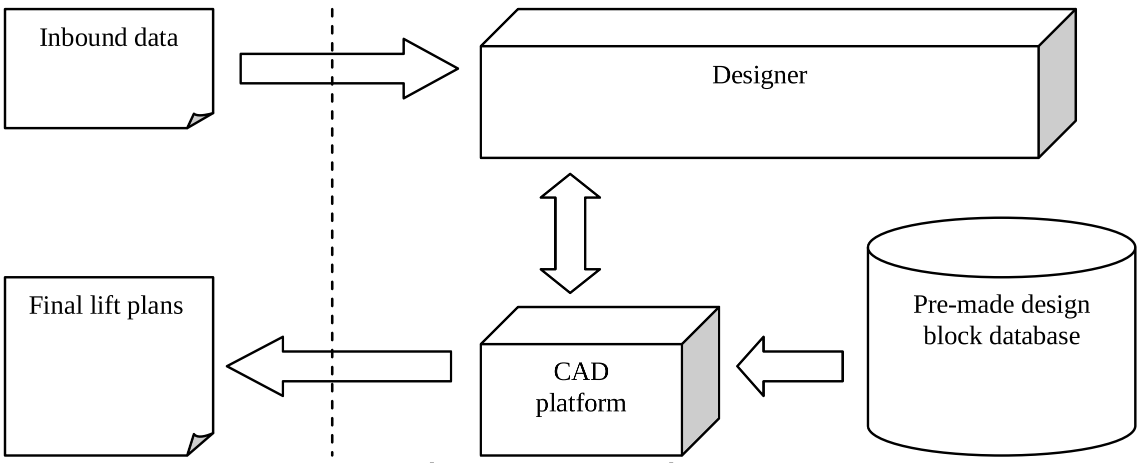

Fig. 2: Abstractive system's architecture

2.1. Inbound Data

The data arriving to the automation system completely and accurately defines the lift to be manufactured. Any decisions that need to be taken are already taken by the rule based system that calculates the lift system appropriate for the given shaft.

The technology selected for the data exchange is the XML, due to (a) its openness, (b) the fact that its an international standard, (c) the ease of access by APIs, software and users alike and (d) the hierarchical structure of the contained data (Bouzakis et al., 2005).

An XML file, containing all the information necessary for the lift plans to be created, is forwarded to the Designer module, via a combination of the technologies: Web services, ASP.NET and .NET Remoting. This file must comply with an XML Schema which only describes which variables have to be defined inside the XML. After its acceptance, the data set is checked for consistency by the Designer module, but not for correctness, since this attribute is ensured by the rule based system and any relevant logic inside the designer could contradict the respective one of the rule based system.

2.2. Commercial CAD Platform Used

The adoption of any new method or tool inevitably induces transient deduction in the efficiency of the design process, as the novelty has to be used to by the designers and incorporated in the rest of the development procedure. In order to ensure a steep learning curve and to minimize the adaptation procedure, software systems that are already used were selected. Specifically, the commercial CAD platform that is used in the company was chosen to be the base where the automation system would be build on.

The specific platform supports a number of APIs including VBA, COM, a C++ library of tools and an equivalent .NET Framework library. Since the refereed solution is to be connected to a web interface and to allow for remote connections, the .NET Framework library was selected being the most flexible and the one that better supports the desirable goals.

2.3. Pre-made Design Block Database

Since a large number of objects on the lift plans are reused, it's efficient to retain a library of pre-drawn blocks of these objects and insert them when needed. These objects include doors, brackets, pilots, mechanisms etc. The most useful aspect of this approach is that it enables the company using the automatic drafting system to expand the variety of this components, by creating new blocks. In any other case, the whole Designer module would have to be edited by a software engineer, recompiled and updated on the server. This way, only the appropriate blocks have to be prepared and inserted in the database. Thereafter, one only has to referee the new block in the XML file, in order to use it.

2.4. Output

The output are the final plans of the lift (top view and sections), ready for printing. The file is in the DWG format. Even though it's not as widespread as the DXF format, it allows for more advanced features and it's the innate format of the CAD platform used by the company. Anyway it does comply with the OpenDWG Specification, allowing any application to access it. Moreover, the ability to export DWF files for publishing on web pages is supported.

2.5. Designer Module

The Designer Module is the central part of the automation system. As can be seen from Figure 3, it consists of three layers: (a) Data Layer, (b) Specific Implementation and (c) Designer Framework. The CAD platform API can not be considered as a part of the Designer module, since it's an external and independent entity, but it can be interpreted as the presentation / user interface layer of a multi tier application.

Using a multi tier architecture (a) infuses the system with great scalability, (b) lowers the maintenance costs, (c) allows for standardization of business rules, (d) increases the flexibility of the application, (e) decreases the debugging time, (f) promotes reusability and (g) allows the use of heterogeneous tools (Bouzakis et al., 2005).

The data layer is responsible for importing and storing the data from the XML file. The lift is precisely defined by the set of data, leaving no decisions to be taken. The only check that is done is a consistency check on the data.

The CAD platform API allows the control of the CAD platform by the automation system, in order to create the final plans. It exports internal functions of the platform to a .NET Framework library, allowing for the development of add-ons to the CAD platform.

The Designer Framework is an intermediate layer acting as a *stored procedures layer* in a client – server multi tier application. It provides base classes and methods that allow for easy construction of an object oriented representation of the lift to be designed. These classes fall into two categories: abstract classes and utilities classes.

Abstract classes provide a skeleton for building the lift from discreet objects like shaft, doors, cabin etc. This way the implementation of a new lift is sped up and the addition of new elements to it is simplified.

Utilities classes on the other hand, group together several functions of the CAD platform API, that are frequently used. Block insertion, entity movement and viewport creation are just some of these functions that are used by many objects. By removing the implementation of these functions from the business layer, the system becomes more modular and the specific implementation layer becomes independent of the CAD platform.

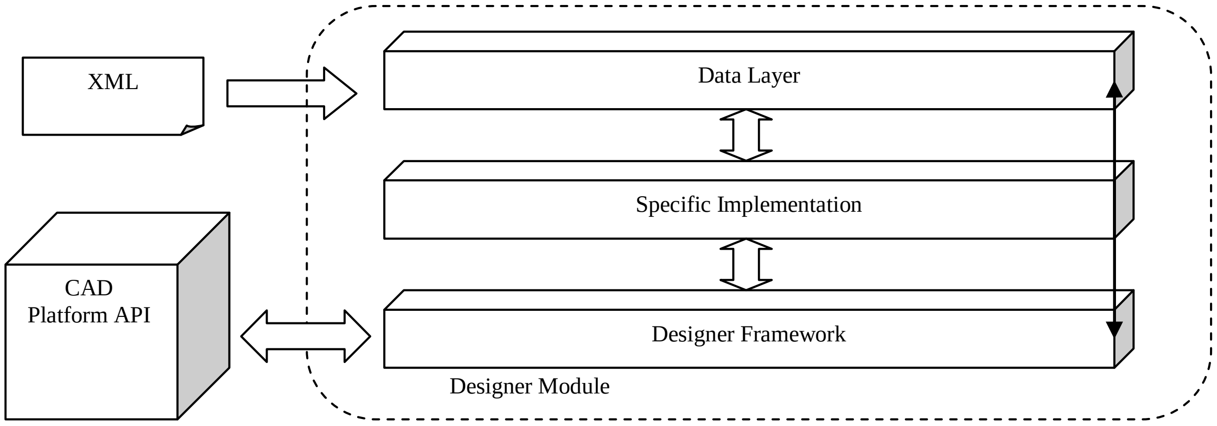

Fig. 3: Designer's architecture

All the previous layers support the implementation of an automated design system. This layer completes the automation system by building each specific lift type. By extending the object oriented concept to the design process, the lift is build by a number of sub-objects. These sub-objects are put together during the automatic design process, and result in the complete lift.

The objects are implemented using classes from the Designer Framework, thus utilizing the functionality already build into it. The only aspect of the objects that need explicit implementation is their parametric geometry and the way they connect with each other.

3. CASE STUDY

For the construction of the lift plans, the whole lift was split in its parts in order to create the object tree. On Figure 4 some of these discreet objects can be seen enclosed in different areas. A number of these objects are composed of a number of sub-objects (like the doors).

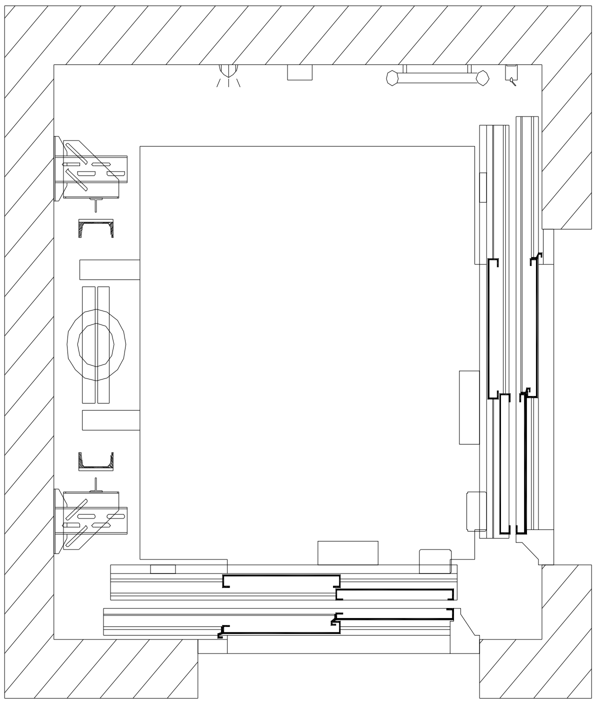

Fig. 4: Lift divided in objects

Secondly, the full range of the variables defining the lift was determined resulting in the XML Schema that prescribes the structure of the data file (on Figure 5 a part of it can be seen).

```
<xs:element name="HAI" nillable="true" type="DataHolder" />
<xs:complexType name="DataHolder">
  <xs:all>
<!--Shaft-->
<xs:el minOccurs="1" maxOc="1" name="ShaftWidth" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="ShaftDepth" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="ShaftHeight" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="Headroom" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="Pit" type="xs:int" />
<!--Cabin-->
<xs:el minOccurs="1" maxOc="1" name="CabinWidth" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="CabinDepth" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="CabinHeighth" type="xs:int"
/>
<xs:el minOccurs="1" maxOc="1" name="Headroom" type="xs:int" />
<xs:el minOccurs="1" maxOc="1" name="Pit" type="xs:int" />
```

Fig. 5: XML Schema portion

The next step was to implement the object tree, with the help of the Designer Framework. Each object's geometry was defined dependent on the received data.
Lastly, the overall server was set-up, consisting of (a) a web interface, (b) a windows service acting as an intermediate server, (c) the Designer Module and (d) the CAD platform.
The use of the system is simple: as the data file is received, it's checked by the data layer and after its approval, the design process initiates. During the design process, the objects composing the lift are created and then the CAD platform is controlled in designing the lift plans.
Finally, the lift plans are saved in the output file.

4. CONCLUSIONS

The results of the current research were really promising. By automating the creation of the lift plans, the throughput of the design department was increased to a manifold of the previous production, requiring mere seconds for each design. The designers can almost instantly see the result of their choices and either try a new set-up or accept the already designed one.
On the other hand, the modelling of each lift type does require an amount of work, but it is significantly aided by the developed framework, which removes the implementation of the CAD platform specific functions from the modelling operation.

5. REFERENCES

1. Bouzakis K.-D., Vakali A., Andreadis G., Karapidakis E., (2005). *Manufacturing Automation of a Workpiece Using XML*, Proceedings of International Conference on Manufacturing ENgineering (ICMEN), pp. 741-750, Chalkidiki, Kassandra
2. Bouzakis K.-D., Andreadis G., Vakali A., *Development of a Web Based System Providing Communication Between Designers And Manufacturers*, Proceedings of International Conference on Manufacturing ENgineering (ICMEN). pp. 759-766, Chalkidiki, Kassandra,
3. Daum B., Merten U., (2003). *System Architecture with XML*, Morgan Kaufmann Publishers, An Imprint of Elsevier Science
4. Myers, B., Hollan, J. and Cruz, I. (1996). *Strategic directions in human-computer interaction*, ACM Computing Surveys, Vol. 28 No. 4, pp.794-802
5. Wu, S.H., Fuh, J.Y.H., Lee, K.S. (2007). *Semi-automated parametric design of gating systems for die-casting die*. Computers & Industrial Engineering, Vol. 53 pp. 222-232

Received: 17 November 2009 / Accepted: 20 December 2009 / Abstract published online: 15 January 2010