Advances in Engineering Software 119 (2018) 60–67

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

An effective and user-friendly web application for the collaborative analysis of steel joints

J. Gracia[*,a], E. Bayo[b]

[a] University of Oviedo, Spain
[b] University of Navarra, Spain

ARTICLE INFO

Keywords:
Web applications in the Internet
Software as a service
Structural analysis of steel joints
Eurocode 3

ABSTRACT

The Internet has increased its potential exponentially since its inception. This progress has been possible due to new standards and technologies, which have also allowed the development of a new type of web applications that are fully integrated in web browsers. In addition, structural analysis has become a collaborative task in which different people have to share information and outputs of analysis programs. In this paper, an effective and user-friendly web application for the collaborative analysis of steel joints is presented. The latest cutting edge technologies in the Internet are used to address fundamental issues inherent in structural analysis software such as visualisation, interaction, and structural evaluation. Specifically, WebGL API, part of the HTML5 standard, is used to solve the visualisation issues of the proposed application. A rigorous analysis of simple and rigid structural joints is performed according to the standards and criteria set by the Eurocode 3.

1. Introduction

Structural engineers have to analyse and design increasingly complex structures and this cannot be done without sophisticated engineering software. This software is continuously benefiting from advances in software engineering. The first revolution started in early 1980s with the advent of personal computers. Monochrome command line applications were superseded by a generation of software with powerful graphic interfaces. They were capable of rendering complex 3D structures in realtime. Later, the Internet revolution, which began in the early 1990s, started a technological race that so far has increased its velocity exponentially, becoming exceedingly dynamic. Structural analysis software has also benefited from this revolution.

The work of a structural engineer includes several tasks: structural sketch and analysis, dimensioning of main elements (beam, columns, bracing systems, etc.) and joint (also called connection) detailing, among others. Nowadays, structural analysis, which requires a high level of computation, can easily be adapted to distributed computing environments over Internet by means of structural analysis web services.

In regard to previous works developed in this area, Yang et al. [17] developed a generic web-based platform for conducting model-based numerical simulations on line. Vacharasintopchai et al. [14] proposed a framework to join web services and the semantic web in the field of computational mechanics. Chen and Lin [4] presented an Internet-based finite element analysis framework supported by a java client application and a parallel computing environment. Also, Chen and Lin [5] proposed an Internet-based computing framework for the simulation of structural systems composed of two levels of parallel processing. Most of this research was focused on the server side; more specifically on ways to distribute the computing tasks that are necessary to solve the large sparse matrices involved in structural analysis.

More recently, Gracia and Bayo [8] developed an integrated 3D web Application for Structural Analysis Software as a service. In this article, a web application, which allowed navigating, analysing and visualizing pre-defined trusses, was presented. The geometry, loads and boundary conditions were predefined and the web application did not allow for changes. The 3D visualization requirements were reduced to render points and lines. According to those requirements, the development was done with plain JavaScript and WebGl [11], a low level API to handle 3D visualization.

However, as mentioned before, structural engineers not only deal with structural analysis but also with joint (or connection) detailing. Currently, there are several commercial applications for joint analysis. Most of them perform automatic designs. However, a desktop approach is contrary to current trends in the global computing market where computer tools based on Hardware as a Service (Haas), Platform as a Service (PaaS), and Software as a Service (SaaS) are attracting more users.

This paper presents a user-friendly web application for the

[*] Corresponding author.
E-mail address: graciajavier@uniovi.es (J. Gracia).

https://doi.org/10.1016/j.advengsoft.2018.02.007
Received 21 March 2017; Received in revised form 8 January 2018; Accepted 25 February 2018

collaborative analysis of 2D steel joints, focused on manual design rather than automatic design. Structural engineers working on building rehabilitation will welcome this characteristic; they usually need to check existing joints. This new application has been implemented for two typologies of joints: simple beam to column joint with bolted angles, and fully rigid beam to column joint with extended end plate. Although this application is aimed at these two types of joints, it has been designed with modularity in mind so that other types of joints can be easily added.

This research primarily deals with the web software architecture necessary to provide a user experience similar to that provided by desktop applications. Interaction and visualisation are critical for this purpose. The application has been developed as a stand-alone front-end web application using current Internet technologies, like HTML5 [16] and WebGl for 3D visualisation. Once it is downloaded and rendered by the browser, there is no need for an Internet connection. The only exception is to persist the data (enable collaborative work), which requires a little back-end server application.

The 3D Visualization is more complex than that of Gracia and Bayo [8]; this new application deals with 3D solids instead of lines and points, and it also allows changing the geometry. Each time the input is changed, the 3D Visualization is updated in real-time. Consequently, the authors have used a different approach: instead of a low-level 3D API, a third-party library has been chosen. Moreover, the model-view-controller architecture to handle this interaction is much more complex. For this reason, a model-view-controller framework is used instead of plain JavaScript. As part of this research, a brief revision of modern JavaScript frameworks for developing web applications is also presented.

2. Web application architecture

Like desktop applications, web software is based on design patterns or architectural schemes. In this type of application the most important architectural software decisions are: 3D visualisation, client–server communication and application logic. Hereafter, those issues are briefly addressed.

2.1. 3D visualization in the web

Nowadays, 3D visualisation is critical in any application targeted at structural analysis. Visualisation feedback is necessary to perceive what it is being modelled and designed. Desktop applications have supported 3D content since the development of the first window based operating system. However, support of 3D content in web browsers was not so common until the development of the WebGL standard.

Before the development of this standard several solutions had been provided. The first approach was Virtual Markup Modeling Language [13] that was superseded by X3D [15]. X3D was the ISO standard, with XML-based file format for representing 3D computer graphics in terms of geometry and material. However, these languages only defined the geometry, and consequently, a 3D plugin was necessary by the browser to render the content. The main solutions to render 3D content in web browsers were: Adobe Air framework, Microsoft ActiveX, and Java applets, which were probably the most commonly used. Specifically related to structural analysis applications, Chen et al. [6] presented a prototype development of web-based structural engineering systems to provide analytical services over the Internet. One of the proposed applications relied on serving 3D content by means of Java applets using GL4Java interface packages, that is, an API to develop OpenGl applications on Java.

However, all the solutions mentioned above suffer from the same weakness: the technology behind them is based on either several abstraction layers or a virtual machine above the hardware layer. This means that a lot of hardware resources are consumed to step through these layers (or virtual machine) before the graphics card displays the data. In addition, potential security issues may constitute another real weakness. It is the opinion of the authors that the applications developed using those technologies are not fully integrated web applications. The web browser is only used to download byte-code applications that are executed by the Java Virtual Machine (JVM) or the plug-in. At this stage the browser is no longer needed. Moreover, as shown by Gracia and Bayo [8], WebGl performance is better than Java applets, starting from a factor of two for low complexity models and reaching a factor of seven for complex models.

WebGl, developed by Khronos Group, is an API specification based on OpenGL for Embedded Systems 2.0 (OpenGL ES) that is also based on OpenGL 2.0 specification. OpenGL 2.0 is an open source standard API for the development of cross-platform 3D desktop applications. This API allows direct communication between the application and the graphics hardware layer with high performance and optimised use of resources. WebGl is widely supported by main web browsers. With this technology JavaScript code inserted in web applications can directly handle the communication with the graphics card. Direct communication means better performance and reuse of computer graphics techniques only available in traditional desktop applications.

However, WebGL, following OpenGL design philosophy, is a low-level API which requires solid knowledge on computer graphics and advance coding skills. Currently, there are several WebGL JavaScript libraries available which provide higher level API to render 3D content. Some of them are oriented to game development but the most relevant ones are general purpose [1,10,12].

Although it is possible to develop a 3D web application only with WebGL low-level API [8] in this article a high-level API has been choosen. Three.js [2] library is preferred instead of plain WebGL.

2.2. Client–server communication channel

Stand-alone web applications do not need to handle communications with the server; the application files are downloaded automatically by the web browser. However, if data persistence is required for collaborative purposes, then client–server communication must be addressed. A reliable mechanism of communication and data exchange format are needed to speed up client–server interaction.

Unlike traditional client–server applications, the development of web applications involve protocols and data formats limited to those allowed by the browser. Currently they allow several protocols, and the main two are: XML HTTP Request (XHR) and Web Sockets (WS). Although both of them are considered matured technologies, the former is best suited for short one-way asynchronous communications while the latter is better for two-way communications. The client application used in this paper makes a unique request to the server, and therefore XHR is preferred.

Once the communication method is addressed, data exchange format should be considered. A HTTP protocol was designed to allow the transmission of any kind of data. Originally, XHR was designed to work with eXtended Markup Language (XML) data; however, aside from its features, it has a major drawback, browsers do not have native support to parse and serialise XML. In this application, the main server tasks are to persist and to distribute data, no data processing is done by the server. Hence, the server does not condition the data format, only the browser does. It stores data in memory as JavaScript objects, then JavaScript Object Notation (JSON) format is selected as data exchange format. Its main benefits over XML are: it runs faster, the data structure is simpler, and the browsers have native support to serialise and parse.

2.3. Application logic

The model-view-controller (MVC) pattern (Fig. 1) is a classic procedure to develop interactive desktop applications. It relies on a clean separation of objects into one of three categories: models for maintaining data, views for displaying all or a portion of the data, and