PROTOTYPING A WEB BASED SYSTEM FOR METAL FORMING PROCESS CHAIN ASSISTANCE

Vencel Biro
Dorel Banabic
CERTETA-Research Center in Sheet Metal Forming
Technical University of Cluj-Napoca,
Memorandumului 28, 400114, Cluj-Napoca, Romania
vencel.biro@tcm.utcluj.ro
tel.: +40740 352 243

Abstract
The equipment in Research Center in Sheet Metal Forming (CERTETA) laboratoty of Technical University Cluj-Napoca need a solution to transport data between them, pieces of equipment that are at significant distance from each other. After a measuring system produces its results these results will have to be copied to a data storage device (CD, flash drive) and physically transported to the laboratory where the simulation stations are located. After the simulators produced their data, the information needs to be, again physically, transported to the presses or other manufacturing equipment. We decided to automatize this process between the machines and different workstations. The process of obtaining the first working prototype is detailed in this paper.

Keywords: prototype, process chain, communication, internet, web development

1. Introduction

A standard metal forming research process includes equipment in different locations. The data between the testing, simulation and manufacturing equipment are usually transmitted with different storage devices. The purpose of the DaCoTraP project is to eliminate the need of these storage devices by allowing the different devices to synchronize between them using the internet. The solution to the distance problem is a platform capable of collecting data, processing it and serving it back upon request. Using this platform, the machines will be able to communicate with each other in almost real-time independently from their location. After studying the existing research in this field we concluded that there are numerous others working on machine-to-machine communication and control through the internet. Using the existing research we will be able to build a platform of applications that will be able to collect data from different devices, process it and redistribute it.

Plenty of similar research has been done in the field. Amos H.C. Ng et al. offer good insight into modeling and simulation [12] together with Byrne et al. who presents three different approaches to simulation exploring the advantages and disadvantages of web-based simulation [16]. Brown et al. paper implements a web-enabled repository system that has been designed for supporting distributed automotive component development [13]. C.D. Tarantilis et al. focus their efforts on a system implementation that combines as many functionalities as possible into a single, integrated software program that runs on a single database, in order that the various modules and parts can easily share information and communicate with each other [14]. Lan presents a remarkably similar system to DaCoTraP specialized in rapid prototyping and manufacturing. A central server application is used by users and machines to work together on rapid prototyping tasks [15]. They also chose a similar architecture for another project of theirs [17].

To prove that there is a possible solution to this problem and that the solution we came up with can be implemented there is a need for a proof-of-concept prototype.

2. The DaCoTraP platform

2.1 Naming of the project

The name chosen for the application boundle was DaCoTraP, an abbreviation of the "Data Collect/Transfer Platform" title. A project codename was urgently needed even at the beginning of the research and implementation so the DaCoTraP was invented. This name will most certainly change during the research period but until then it denotes the real life application part of the whole project.

2.2 Requirements

To be able to include machines from outside of our laboratory complex, maybe from anywhere around the world we decided to use the internet to transfer data thus creating the basic but most important requirement that a piece of machinery has to have in order to be part of the system: access to internet. Using the internet the devices should be transferring the experiment data between them creating the following path of information: the information arriving tto the simulation system should be composed of multiple experiments.
We could see that we are going to need a server and separate client applications that will be deployed on the devices themselves to communicate with the server. To keep track of the equipment, protect the data and have some kind of overview of the status of the projects there is a need of a web application that allows users to register themselves and the

devices and issue tasks. There is a need for a webserver that communicates with the different clients and of corse we need different clients for different machines. Different clients are needed becasue the platforms differ from each other mostly in operating systems and data structure. The most important part of the system is the web service through which the communication flows.

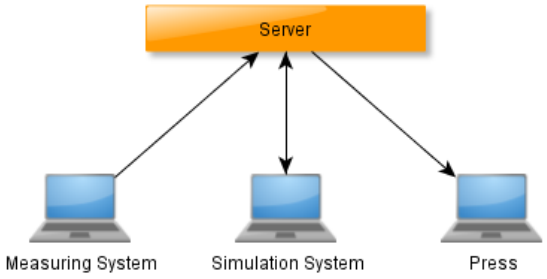

Fig. 1 - The direction of information in the new sytem

All the information has to be stored somewhere and be accessible by the web application and the web service so there is a need for a central database . To maintain this central database there is need for another application that is responsible for archiving, email sending, cleanup and monitoring.

2.3 Not Real time (NRT) application

The principal responsibility of a real-time (RT) system can be summarized as that of producing correct results while meeting predefined deadlines in doing so. Therefore, the computational correctness of the system depends on both the logical correctness of the results it produces, and then timing correctness, i.e. the ability to meet deadlines, of its computation.

A RT application can be modeled as a set of cooperating tasks. These tasks can be classified according to their timing requirements, as hard real-time (HRT), and not real-time (NRT). A HRT task is a task whose timely (and logically correct) execution is labeled as critical for the operation of the whole system. The deadline associated to a HRT task is pronounced hard deadline. Consequently it is assumed that the missing of a hard deadline can result in a tragic system failure. NRT tasks are those tasks which exhibits no real-time requirements (e.g. system maintenance tasks that can run occasionally in the background).

The taxonomy of application tasks can de further expanded with the terms periodic, aperiodic and sporadic. Periodic tasks are tasks which enter the execution state at regular intervals of time, i.e. every T time units. These tasks are usually associated with hard deadlines.

Aperiodic tasks are tasks whose execution time cannot be anticipated, as their execution is determined by the occurrence of some internal or external events. These tasks are usually NRT tasks. Finally, aperiodic tasks bound to hard deadlines are termed sporadic tasks, e.g. tasks dealing with the occurrence of system failures (exceptions) or prioritized responses to some event. With the above classifications in mind, one can observe that the principal responsibility of a RT operating system is to guarantee that each individual execution of each application task can meet the timing requirements of that task. However, it is worth noting that, in order to fulfill that responsibility, the objective of a RT operating system cannot be stated just as that of minimizing the average response time of each application task; rather the fundamental concern of a RT operating system is that of being predictable [1].

Being non-real-time is the biggest difference between the application and other virtual laboratory implementations. C. C. Ko even uses a Java applet web page tocontrol the laboratory in real-time [4].

2.4 Time-Triggered

Two general paradigms for the design of predictable RT system can be found: Event-Triggered (ET) and Time-Triggered (TT). In ET RT system any system activity is initiated in response of the occurrence of a particular event, caused by the system environment (mainly software or hardware interrupt vectors). In TT RT system activities are initiated as predefined instants of the globally synchronized clock recur (scheduling, dispatching of events).

In the robotic systems there are all the classes presented above. Path planning and servo-loops are periodic (Time-Triggered), the sensor events are usually aperiodic, emergency signals and some other sensor events are sporadic (Event-Triggered) [1].

The system will not communicate in a Real Time manner because it is not necessary, also not possible with a large number of connected devices. This means the system is non-real time (NRT).

Each communication will be initiated by the devices connected to the system in certain preset intervals of time thus the system is time triggered (TT) [2].

2.5 Technologies

The application bundle needs to be implemented as fast and with as little effort as possible. It also needs to be highly maintainable. Microsoft's *.Net* technology was chosen as the basic technology (unlike an enormously high percentage of researchers who use Java [4] or C++ on Linux [5]). The web application will be an *Asp.Net* application, the communication service (called Data Service) an *Asp.Net web service*, the maintaining application (called Maintenance Service) a *Windows service* and the database *SQL Server*. The clients that run on machines supporting Windows will also be Windows services.

2.6 Trimming features to reduce effort

To put together a platform that includes user registration, web service communication, database use, web technologies, multiple experiments to simulation features and many others is a huge task and requires a very long implementation time. This is why the following steps were taken to reduce the implementation time and have a prototype ready as soon as possible:

- Eliminate the option to use multiple experiments with one simulation. The system will send the experiment results to the simulators separately. If multiple packets are needed then the simulator can wait. This way the complexity is reduced and human interaction can handle the few exceptions when the system is used differently form the simple approach.
- Database approach will be handled with *Linq To SQL* object relational mapper. This replaces the need for a database access layer with a generated, already available solution. Using ORMs also drastically increases the maintainability of the system because database change errors will be signaled immediately at compile time [6].
- Eliminate the manufacturing part from the equation: the prototype will only handle the measurement and simulation systems.
- Focus on Windows clients: the only supported machines by the prototype are the ones that run on Windows so a single technology has to be used for the clients.
- Reuse as much as possible from the client code. In order to avoid the implementation of one client/manufacturer the basic application code will be reused by all clients. The manufacturer-specific features will be implemented as plugins.
- Existing frameworks and resources will be used for the prototype; nothing is implemented new if it is not necessary. The web application will use a modified Visual Studio basic webpage template, the CSS library will be Twitter's Bootstrap [7] and the JavaScript library used will be JQuery. All these libraries are highly efficient, robust solutions used and supported by many [8]. For CAPTCHA we will use Google's ReCAPTCHA [9], for login and register functionality we will use a modified *membership provider* from the basic Visual Studio template.
- Manager Service (the service responsible for the maintenance of the database) will not be included in the prototype.

3. Implementation and communication

3.1 Basic implementation, feedback

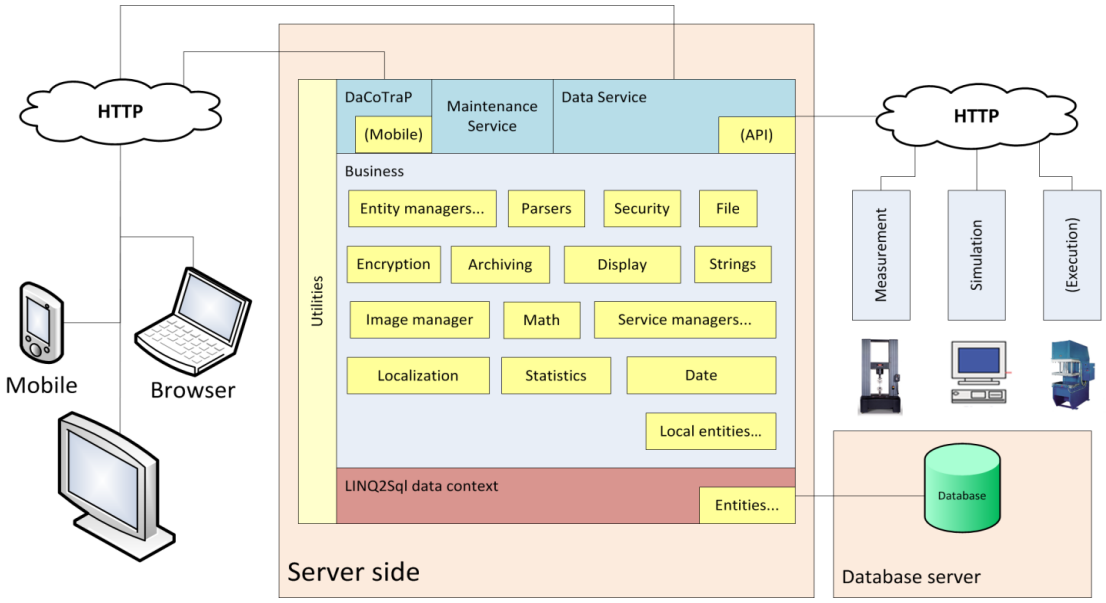

Fig. 2 - Overall structure of the DaCoTraP platform

After the requirements and the technology were defined together with the new restrictions a basic version was implemented but this alpha prototype version was not usable without feedback from the lab workers who will be the majority of the platform's user-base.

During discussions we have discovered that the routine of our colleagues in the laboratory includes them taking their work (aka the experiment files) home with them. Because the clients all work locally on the machines we had to make possible for the users to manipulate the data a bit more freely. The prototype needs this feature in order to be able to prove that it does the job it is supposed to

do: making the users' life easier so two new requirements were added: upload/download experiments/simulations as files to/from the website.

3.2 Communication, web service

Institutions and companies usually have very strict firewall rules and forbid communication in through almost every port. This is why the port 80 will be used for client-server communication solving the quite big problem of firewalls; the port 80 communication is allowed by every firewall.

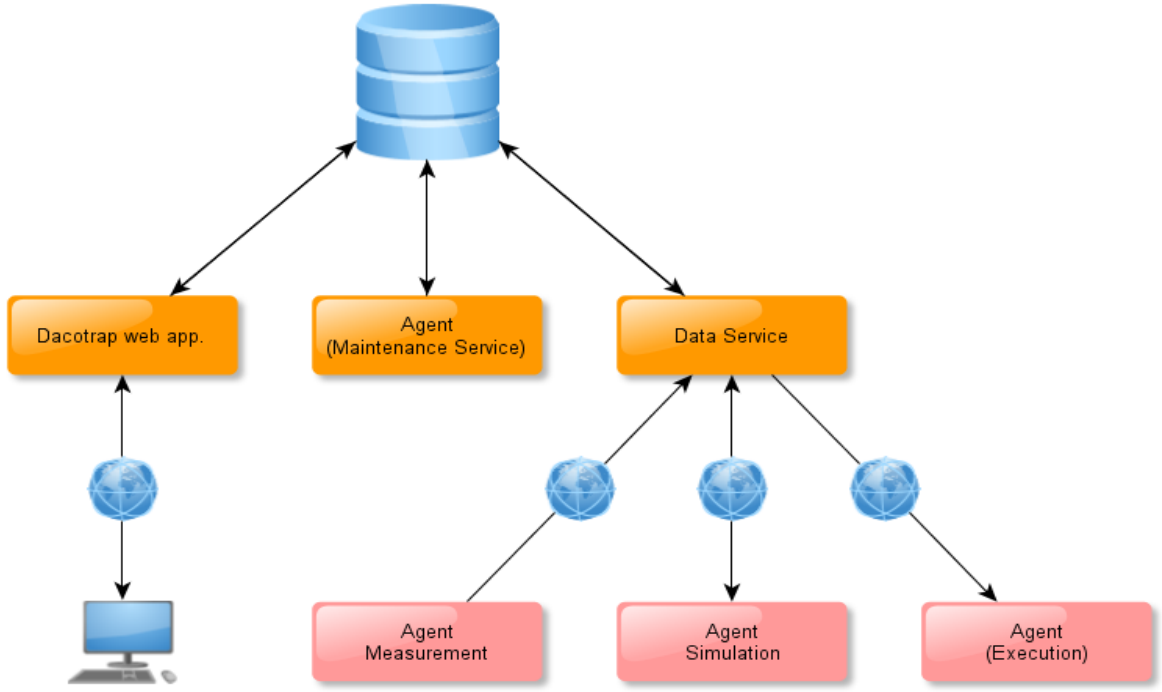

Fig. 3 - Path of communication between the different parts of the project

The key component in the communication is a .Net web service which will be called *DataService*. The Data Service is a RESTful web service. The communication protocol will consist of very basic HTTP methods: GET, POST, PUT and DELETE. This DataService exposes two categories of methods: the first category only contains one method which is aimed to serve the devices; the other category will contain API members mostly targeted to serve data (mostly test results) to third party clients. The system will only have a single method responsible for device communication. It will have a *string* return type and a most important *string* parameter. The service description (WSDL) will be short; this will shift the implementation effort from the devices to the DaCoTraP application because the devices will only have to be able to call one web method. The communication will be one sided; always initiated by the devices by calling this method.

3.3 Web application

The Web interface is an ASP.Net webpage. After passing the login page the users can organize the devices, visualize the data sent by them and issue commands.

The web application will take advantage of Ajax and the JQuery framework. The design of the application will strive to be as efficient, clean and effective as possible from the point of view of the architecture [10] and used pratices [11].

User will have access to all devices and all experiments; otherwise the assignment of the equipment would hinder the users in the use of the application. This is why the application supports a single user role which has maximum access to every zone of the application.

```
public string UploadMeasurementFile(string serial,string validationCode, string collectedDate,
string fileName, Stream fileContent)
```

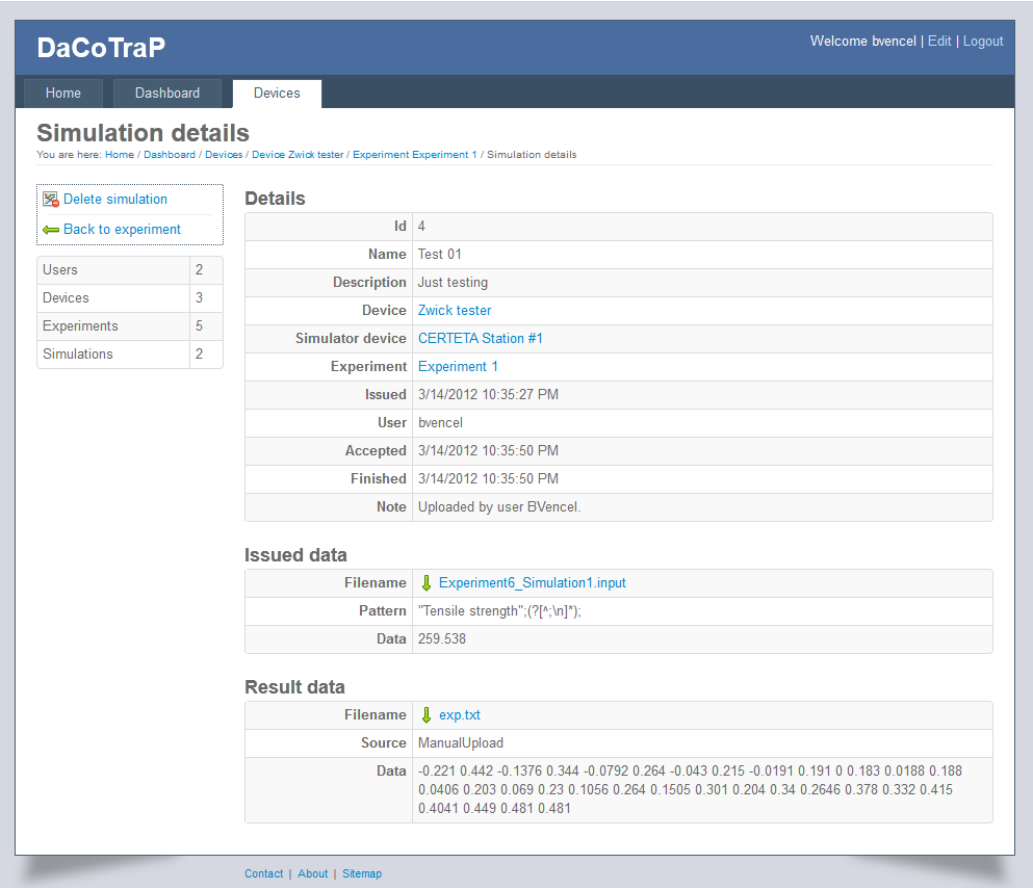

Fig. 4 - Completed experiment screenshot

3.4 Agents, clients

All clients are Windows Services that run different plugins for every job. After startup the service loads all plugins from a specific plugin folder. Each plugin gets a separate timer assigned to them. The plugin provides information about itself: name and the interval at which it should be executed. Also each plugin is separately configured with a configuration file. All the upload clients and the Manager Service use the Agent-Plugin architecture.

4. Conclusions

DaCoTraP platform will be more flexible than the already existing solution by allowing limitless number of devices to be connected with each other and making the communication independent from the content. It eliminates the need for physical data carriers and the limitation of range; the number of device types can be easily increased by implementing the client application for more types of machines. After completion the solution can be extended with numerous third party features by using the API. For example a desktop application can download data and visualize it on a big screen placed at the entrance of the facility; ATMs can display donation requests while offering real production data. A mobile version could offer users almost the same functionality as the web application and mobile apps could use the service as their service façade.

After having a working prototype the effort can be directed to implementing new features.

5. Acknowledgements

The paper has been elaborated as part of the projects: "Studii doctorale în ştiinţe inginereşti în scopul dezvoltării societăţii bazate pe cunoaştere - SIDOC", Contract no. POSDRU/88/1.5/S/60078 and PCCE-100/2010.

6. References

[1] [11.05.2012] http://www.ifr.mavt.ethz.ch/research/xoberon/introduction.html

[2] J. Heilala, *Open Real-time Robotics Control - PC Hardware, Windows/VxWorks Operating Systems and Communication*, PhD Thesis, 2001.

[3] S. Kumar Parida: *Framework and Implementation of a Vision Based Tele-robotic Control over Internet for an Industrial Robot*, PhD Thesis, 2009.

[4] C. C. Ko et al.: *Creating Web-based Laboratories*, Springer, Heidelberg, 2004.

[5] F. Davioli et al.: *Remote Instrumentation and Virtual Laboratories*, Springer, Heidelberg, 2010.
[6] [11.05.2012] http://en.wikipedia.org/wiki/Object-relational_mapping
[7] [11.05.2012] http://twitter.github.com/bootstrap/
[8] Bear Bibeault, Yehuda Katz: *jQuery in Action*, Second edition, Manning, 2010
[9] [11.05.2012] http://www.google.com/recaptcha
[10] Microsoft Patterns & Practices Team: *Microsoft Application Architecture Guide*, Microsoft Press, 2009
[11] Stephen D. Ritchie: *Pro .Net Best Practices*, Apress, 2011
[12] Amos H.C. Ng et al.: *Virtual manufacturing for press line monitoring and diagnostics*, International Journal of Machine Tools & Manufacture, 2008, p. 565–575
[13] D. Brown et al.: *A Web-enabled virtual repository for supporting distributed automotive component development*, Advanced Engineering Informatics, 2004, p. 173–190
[14] C.D. Tarantilis et al.: *A Web-based ERP system for business services and supply chain management: Application to real-world process scheduling*, European Journal of Operational Research, 2008, p. 1310–1326
[15] H. Lan: *Web-based rapid prototyping and manufacturing systems: A review*, Computers in Industry, 2004, p. 51–67
[16] J. Byrne et al.: *A review of Web-based simulation and supporting tools*, Simulation Modeling Practice and Theory, 2010, p. 253–276
[17] H. Lan et al.: *A web-based manufacturing service system for rapid product development*, Computers in Industry, 2009, p. 643–656