DaCoTraP – A web based platform for metal forming process chain

BIRÓ VENCEL *, BANABIC DOREL **

* Technical University of Cluj-Napoca, vencel.biro@tcm.utcluj.ro

** Technical University of Cluj-Napoca, banabic@tcm.utcluj.ro

Abstract

A standard metal forming research process includes equipments in different locations. The data between the testing, simulation and manufacturing equipments are usually transmitted with different storage devices. The purpose of the 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. The most important one of the applications will be the ASP.Net webpage project that will allow users to manipulate the devices and manage/visualize data. The different components of the system will communicate using HTTP protocol through the port 80 and use an XML-based custom language. Because of its high configurability, flexibility, low price and robustness the solution will be applicable in almost every situation.

Key words

Manufacturing, metal forming, process chain, communication, internet

Introduction

A standard and simplified manufacturing process includes machines in different rooms, buildings, even continents. Testing devices determine the parameters of a work piece; simulation equipment processes the data from the simulation equipment. All these results are used by the cold presses that use simulation data and the measuring data to test the manufacturing of the part by creating the actual item. The data between the testing, simulation and manufacturing devices is usually transmitted with optical discs or any kind of storage devices.

The purpose of the project is to eliminate the need of these storage devices by allowing the different devices to synchronize between them using the internet. Using the platform offered

by the project devices will be able to communicate with each other in almost real-time independently from their location. The solution to the distance problem is a platform capable of collecting data, processing it and serving it back upon request. The platform needs to be generic enough to be used with multiple machines (by offering interfaces and data formats that are standard within the system). It also needs to be easily accessible by the authorized personnel and needs to be able to scale well. Besides the communication the system will be able to keep track of many other parameters of the devices like status, error messages and statistics.

State-of-art in the web based platforms in manufacturing

With the web based framework Bouzakis et al. (2009) propose that designers and manufacturers communicate for the manufacturing of a work piece in a platform-independent, easy, fast and more economical way. A similar structure we intend to develop in DaCoTraP project. Lan (2004) presents a similar system specialized in rapid prototyping and manufacturing. A central server application is used by operators and machines to work together on rapid prototyping tasks. Brown et al. (2004) describe a web-enabled repository system that has been designed for supporting distributed automotive component development. Byrne et al. (2010) presents three different approaches to simulation while Chen et al. (2006) describe a framework for an automotive body assembly process design system. The system has the advantage of an open structure that can be easily modified and adapted to accommodate existing assemblies and to suggest areas for improvement.

DaCoTraP (Data Collect/Transfer Platform)

DaCoTraP is an application formed by multiple components all tied together by a central database. These components all work together to gather data from different devices, process this data and distribute it among the same or other devices. The uniqueness of the platform is given by its robustness and configurability. There are countless real life applications that try to solve the issue of different machines communicating with each other over the internet over large distances, like Kumar Parida (2009) in his PhD thesis. The application differs from these by not considering speed as a priority and not worrying about packet loss. Instead the project focuses on being robust and supporting thousands of connected machines while making sure that no data is lost or corrupted.

The robustness of the system is ensured by design: the system is a non-real-time system. The packages sent by the machines will not arrive at the same time instead the machines each send packets in certain configurable intervals ensuring that the system has a steady low load. The system also aims towards being very open towards different types of machines. While the server side of the application uses state-of-the-art technologies, abstractions and frameworks the client are allowed to be very low level application which only need to support sending packets through the internet and parsing text. This allows the system to support a large number of different client types.

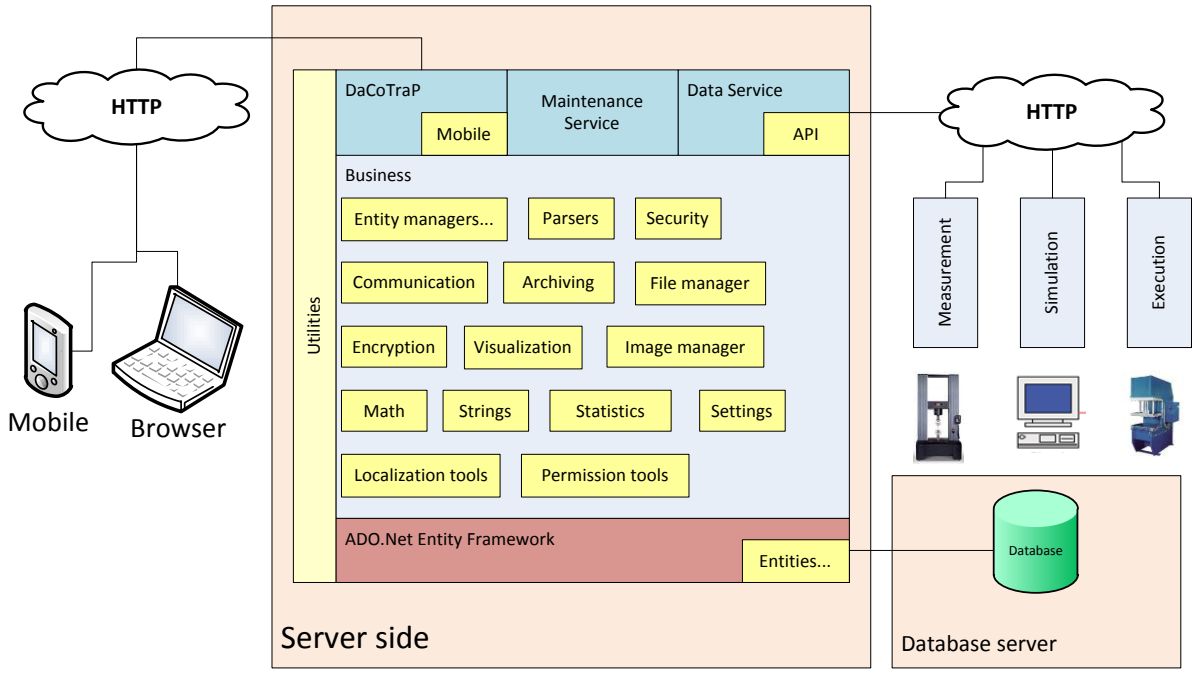

Figure 1 - Overall structure of the DaCoTraP platform

One other important quality that distinguishes this project from the other similar projects mentioned beforehand is that the area affected by the software does not have to be limited to a single factory; multiple machines from factories, laboratories from all over the world can work together by communicating, building a central knowledgebase and generating useful information.

Project/Solution structure

The .Net solution will contain the following projects: the main Web application simply called *DaCoTraP*, a Windows service called *MaintenanceService*, and the web service responsible for the communication and containing the API called *DataService*. The DaCoTraP project will also contain the mobile version of the application which does not form a standalone project but it is simply part of the same web application.

Besides the main projects which all act as *user interfaces* and *top layers* in the architecture the solution contains a project called *Business* (containing the manager classes that are collectively used by the top-level layers) and a project called *ProjectUtilities* or Utilities which will contain the common interfaces, constants, enumerations and data transfer objects used through the whole project and is referenced by every other application layer. The names of the projects are shortened for the sake of readability; the assemblies will be strongly named according to .Net standards. For logging purposes the Apache Log4Net library will be used.

The web application that allows users to manage the tasks and visualize data is going to be built in ASPX taking advantage of the *Ajax Control Toolkit* and the *JQuery* JavaScript library. The content will be completely separated from the design by using CSS. To visualize

the data Microsoft web charting will be used which is part of the .Net framework since version 4.0. The application will run on an IIS 7+ webserver in integrated mode. To be able to handle high traffic and server farms the application will take advantage of Out-Of-Process session handling using an SQL server to store users' session data.

The MaintenanceService is a Windows service that will be responsible for the management of the database and execution of asynchronous tasks. The MaintenanceService will send emails from the email queue; generate reports and notifications at end of specific periods, archive the data, issue tasks for devices, clean up, take care of user registrations and many more. All the tasks of the service will run on separate thread controlled by separate timers. The interval between the tasks can be configured using the application configuration file but the execution of a task can also be tied to a trigger. Besides Log4Net Windows event logging will be used to message the administrators about the status of the service.

The purpose of the .Net web service is the receiving of the device data packages and the providing of the responses. The service will use the *System.Xml.XmlDocument* .Net framework class to validate and process incoming messages and to construct the responses. In the future the service might be expanded to host different applications (Silverlight, Flash) and to offer these applications a RIA service façade.

The same business layer of the application is used by the primary layers and it contains the most important part of the application: the ADO.Net Entity Framework data model. All the business functionality is implemented in manager classes inside the Business layer. By using an Entity Framework data model there is no need for a separate database layer. The generated database model will host all the database entities which can be used as DTOs between the primary layers and the Business layer. In case a DTO needs to be used through more layers it should be separated and moved to the ProjectUtilities. There is usually a manager for every entity.

The managers implement the Singleton and Unit of Work patterns. Using these patterns and supplying the database connection string to all methods of the managers the Business layer is able to operate in stateless mode; this solves the problem of using the same business layer for a stateless (DaCoTraP web application) and "stateful" applications (MaintenanceService).

Communication

The key component in the communication is a .Net web service which will be called *DataService*. 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 single *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.

```
[WebMethod]
public string CollectDeviceData(string xmlData)
```

The communication protocol will consist of a very basic SOAP over HTTP. 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.

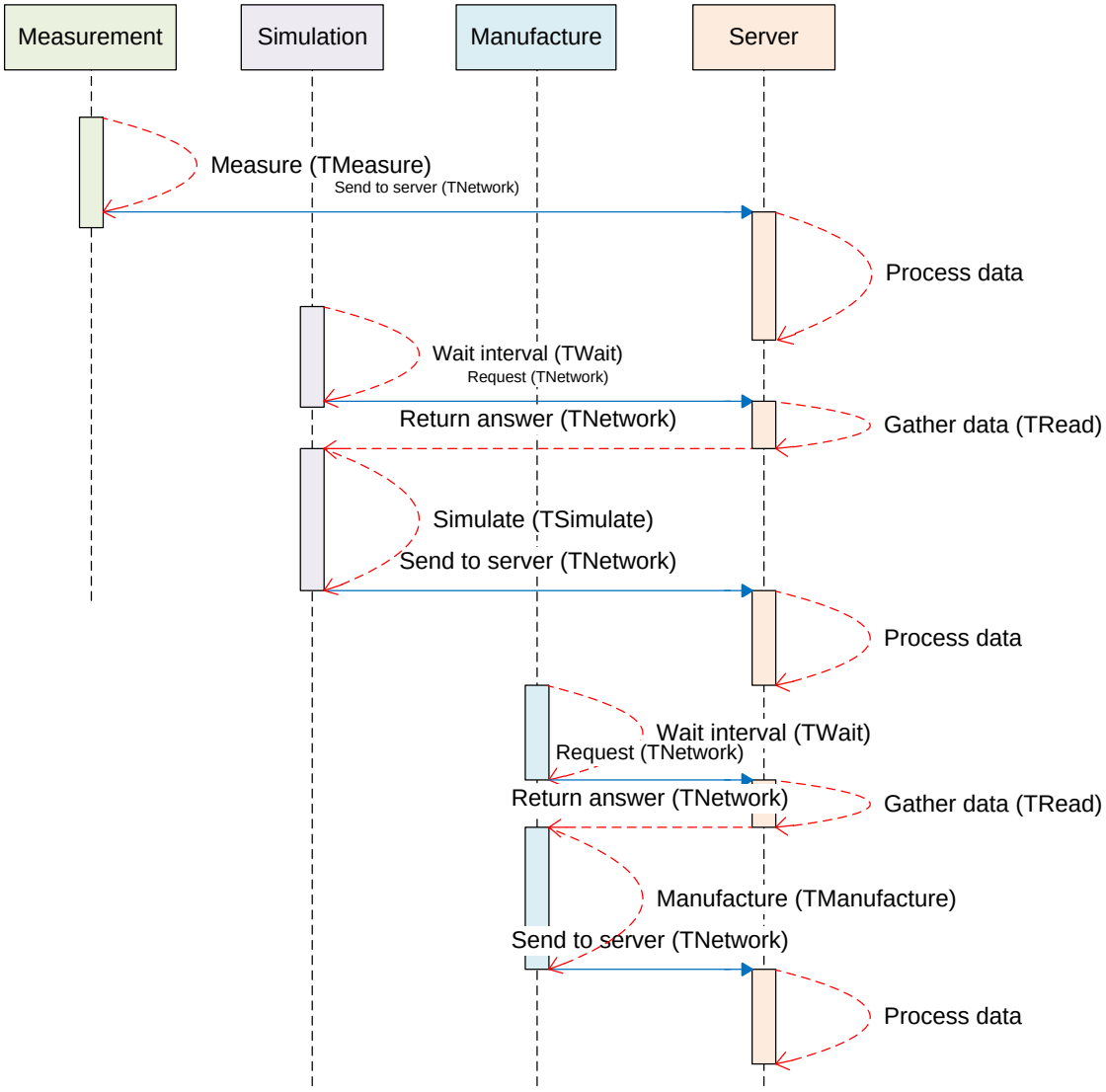

Figure 2 - Sequence diagram illustrating a full Measurement - Simulation - Manufacture cycle

The single parameter the CollectDeviceData method accepts is a string which itself is an XML document. These XMLs are constructed by the devices and contain separate formats with the exception of the authentication block that will identify the sending device. This

package will be processed by the server and a response will be sent back as the return type of the web method.

There will be two types of messages based on the purpose of the device: data sending messages that send information and do not expect any information from the servers besides an OK or NOK and data requesting messages that expect the server to send them data; these will receive a formatted XML document from the server as a response to their request.

Messages sent by the clients can have the following purposes:

- Send availability: the device will be able to communicate its status to the server. Users will be able to see if the machine is idle, restarting, in error mode or simply unreachable
- Send technical information: technical data can be sent for example by the presses setting their manufacturing year, service status, serial numbers and other data unknown at the time of the registration
- Send error messages: error logs can be uploaded to the server
- Send results: results can be uploaded to the server
- Request tasks: the machine receives tasks if there are any issued to it
- Request settings: the machines can request settings from the server like the correct time, default speed and force parameters

The packages used for communication need to be well-formed and valid Xml documents. The casing of the Xml documents needs to be respected by the clients.

```
<?xml version="1.0"
encoding="utf-8"?>
<Request type="task">
  <Auth>
    <Username>usr</Username>
    <Password>pass</Password>
    <DeviceId>21EC2020-3AEA-
1069-A2DD-08002B30309D
    </DeviceId>
  </Auth>
  <GetTasks>
    <FromDate>
      1269508261
    </FromDate>
    <ToDate>
      126950826055
    </ToDate>
  </GetTasks>
</Request>
```

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="
qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Request">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Auth">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Username" type="xs:string" />
              <xs:element name="Password" type="xs:string" />
              <xs:element name="DeviceId" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="GetTasks">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="FromDate" type="xs:unsignedInt" />
              <xs:element name="ToDate" type="xs:unsignedLong" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="type" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Code 1 - Task request package Xml document and Xml schema

Storage

The application suite will use one central Microsoft SQL Server. An Entity Framework model will be generated based on this SQL Server structure and the database will be manipulated with Linq queries. The usage of Linq queries will make the maintenance of the data access very easy. The amount of data received from the devices will be huge especially if all the status and error message history has to be kept in the system. This could end up generating tens of millions of database records and querying a table this large is problematic. To resolve this issue, layers of archiving will be introduced based on the freshness of the data. Besides archiving the freshest data record will also be kept in a separate table. In the case of error messages the arriving message will be inserted in many different tables simultaneously: the actual data table (holding only the latest record), the daily data table, the weekly data table then the monthly yearly and overall. When querying for data the system will always try to use the database table with the least records. The current status of the device or the time of the last sent record can easily be determined using the actual data table. Every non-peak hour (eg. from midnight to 3 AM) the MaintenanceService cleans up the expired data. Each of the huge tables will have well defined indexes and the simple queries will be pre-compiled. The system will rely on the System.Web.Caching namespace to store reusable data.

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.

The prototype of the DaCoTraP platform is being implemented in the Sheet Metal Forming Research Centre (CERTETA) belong the Technical University of Cluj Napoca.

References

K.-D. Bouzakis, G. Andreadisa, A. Vakali, M. Sarigiannidou, *Automating the manufacturing process under a web based framework*, Advances in Engineering Software, 40 (2009) 956–964

D. Brown, D. Leal, C. McMahon,R. Crossland, J. Devlukia, *A Web-enabled virtual repository for supporting distributed automotive component development*, Advanced Engineering Informatics, 18 (2004), 173–190

J. Byrne, C. Heavey, P. J. Byrne, *A review of Web-based simulation and supporting tools*, Simulation Modelling Practice and Theory, 18 (2010) 253–276

G. Chen, J. Zhou, W. Cai, X. Lai, Z. Lin, R. Menass, *A framework for an automotive body assembly process design system*, Computer-Aided Design, 38 (2006) 531–539

S. Kumar Parida, Framework and Implementation of a Vision Based Tele-robotic Control over Internet for an Industrial Robot, PhD Thesis, IIT Kharagpur, 2009 November

H. Lan, Web-based rapid prototyping and manufacturing systems: A review, Computers in Industry, 54 (2004) 51–67

H. Lan, Y. Ding, J. Hong, H. Huang, B. Lu, *A web-based manufacturing service system for rapid product development*, Computers in Industry, 60 (2009) 643–656

S. Millett, *Professional ASP.NET Design Patterns*, Wiley Publishing, Inc., Indiana, USA, 2010

M. Perdeck, *ASP.NET Site Performance Secrets*, Packt Publishing Ltd., Birmingham, UK, October 2010

Acknowledgments: this paper has been elaborated as part of the projects: "PhD research in the field of engineering with the purpose of developing a science-based society - SIDOC", Contract no. POSDRU/88/1.5/S/60078 and PCCE-100/2010.