TOWARDS BUILDING SOFTWARE PROJECT TELEMETRY TOOLS

Adrian ROMAN

Abstract

Software project telemetry is a new approach to software project management in which sensors are attached to development environment tools to monitor the development process and products. Data obtained is abstracted into high-level perspectives and gives to project members useful insights for local, in-process decision making. This article identifies some general requirements for building sensors, proposes a possible architecture for the telemetry system and presents an implementation of the proposed architecture.

1. Introduction

According to the *Encyclopedia Britannica*, telemetry is a *"highly automated communication process by which measurements are made and other data collected at remote or inaccessible points and transmitted to receiving equipment for monitoring, display and recording"*.

Software project telemetry is an approach to software project management in which sensors are attached to development environment tools to unobtrusively monitor the development process and products. This data is abstracted into high-level perspectives on development trends called telemetry reports, which give project members useful insights for local, in-process decision making. [1]

In software project telemetry systems data is collected automatically by tools / sensors that regularly measure various characteristics of development environment. Data consists of a stream of timestamped events and developers and managers can immediately access the data.

Sensors are attached to development environments and can be implemented as plug-ins (in certain cases called *add-ins* or *add-ons*). Plug-ins are optional components which can be used to enable the dynamic construction of flexible and complex systems, passing as much of the configuration management effort as possible to the system rather than the user, allowing graceful upgrading of systems over time without stopping and restarting. [2]

This article identifies some general requirements necessary for building sensors and proposes a possible architecture for a software project telemetry systems. Also an implementation of a Visual Studio Add-in for project line counting is introduced.

2. General Requirements for Telemetry Sensors

The basic function of a sensor is to collect data from the development environment and send it to a centralized collector application. In order to obtain usable and valuable data, more requirements have to be fulfilled by a sensor:

- it has to be completely integrated to the development environment
- it has to be able to send data a centralized warehouse (server application)

- it has to be able to be able send data automatically
- it has to gather significant project management data (e.g. count the code lines for the current open projects)
- a setup has to be provided (easy deployment)

3. A Proposed Architecture

A software project telemetry system consists from three components: the sensor, the collector (server) application and the data visualization interface. Figure 1 illustrates the system's overall architecture.

As previously stated, the sensor can be implemented as a plug-in or add-in to be attached to the development environment. Developers instrument the project development environment by installing sensors into various tools, such as their editor, build system, and configuration management system. Once installed, the sensors unobtrusively monitor development activities and send process and product data to a centralized Web service. Project members can log in to the Web server to see the collected raw data and run analyses that integrate and abstract the raw sensor data streams into telemetry.

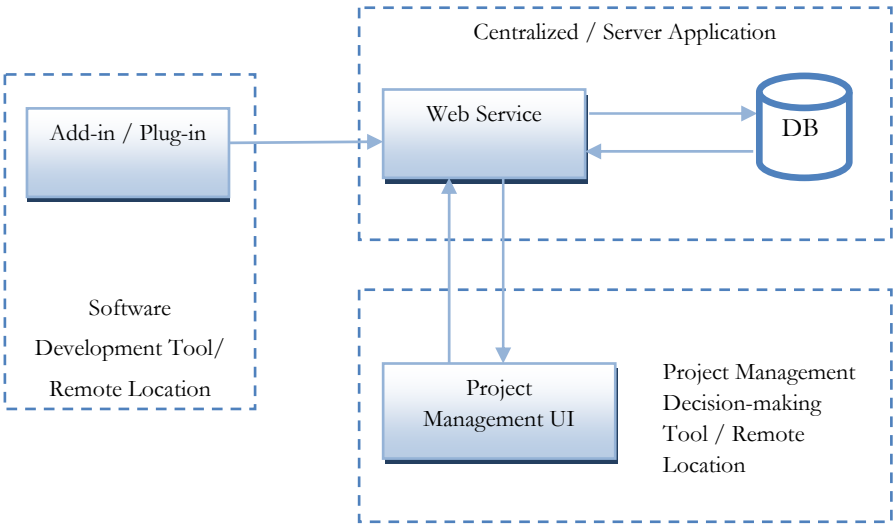

Figure 1. Proposed architecture for a software project telemetry system

The path from sensors to the telemetry report involves several steps. Sensors collect raw data by observing behavior in various client tools and then send it to the server application (Web Service), which persists the data in a database. Analysis mechanisms then abstract this raw data, which can involve synthesizing data from multiple sensors.

4. Plug-in and Architecture Implementation

As a concrete example of software project telemetry in action, I have implemented a Visual Studio Add-In that counts the lines of code of open projects and sends the data to a centralized database where it is visualized. The add-in sends to the Web Service the following raw data:

- *Computer Name* – the name of the computer the add-in is installed

- *Project Name* – the name of the project
- *Total Lines* – the number of lines contained by the project
- *Code Lines* – the number of code lines in the project (< *Total Lines*)
- *Blank Lines* – the number of blank lines in the project (< *Total Lines*)
- *Net Lines* – the difference between the number of total lines and blank lines
- *Comments* – the number of lines that contain comments (< *Total Lines*)

A web application takes the data from the database (through the Web Service) and visualize it to facilitate interpretation and decision making. This application is available to the following web address: http://phd.adrianroman.ro/pmtt (Figure 2 shows a screenshot of the application). The project manager or any other person that has access to the application can select a computer name and a project. Then a graph is built to show the evolution of code lines (total lines, code lines, blank lines, net lines, comments) for the project.

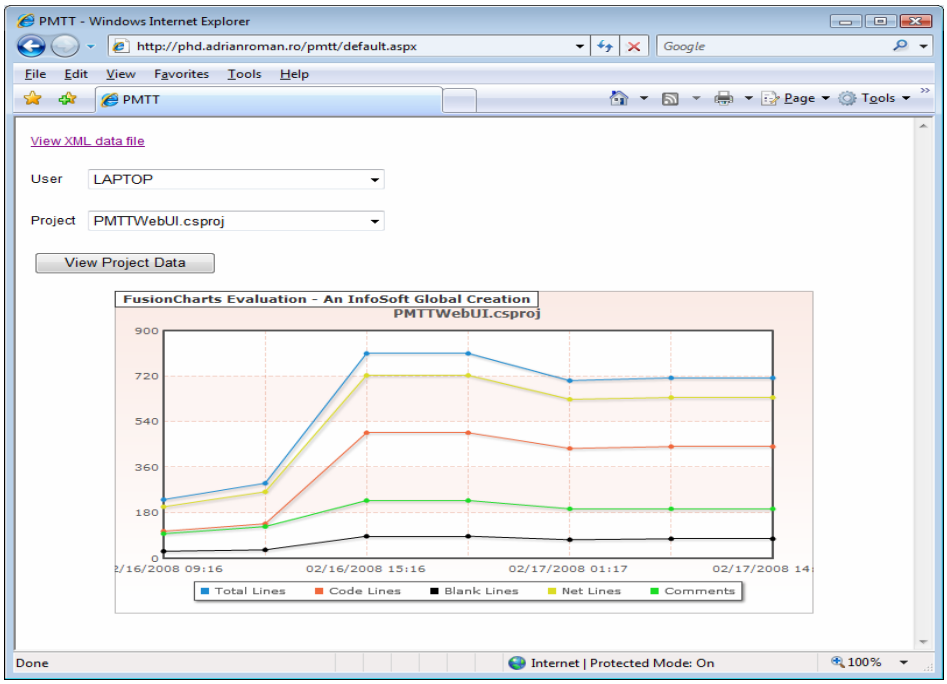

Figure 2. Web application to visualize sensors data

5. Conclusions and Future Work

This article introduces general requirements sensors to be used in software projects telemetry systems, proposes a possible system architecture and presents an implementation of the proposed architecture. These are only the first steps towards building project management telemetry tools.

The proposed architecture presents sensors for interactive development environments (Visual Studio, Eclipse, NetBeans, etc), but other types of applications can be considered to be part of a software project telemetry system:

- office productivity applications, including Excel, Word, PowerPoint, and Frontpage;
- build tools, including Ant and the Unix command line;
- testing tools, including JUnit;
- configuration management, including Concurrent Versions System
- defect tracking tools.

Telemetry data's decision-making value is only as good as the data that sensors can obtain. The proposed architecture does not refers to any metrics to be used. The presented implementation only counts the lines of codes and sends it to the server application. A more elaborate approach is necessary, questions like "What sets of sensors and sensor data types are best suited to what project development contexts?" have to be answered.

6. References

[1] Johnson, P. M., Kou, H., Paulding, M., Zhang, Q., Kagawa, A., and Yamashita, T., "*Improving software development management through software project telemetry*". Software, IEEE, 22(4):76–85, 2005

[2] Reinhard Wolfinger, Deepak Dhungana, Herbert Prähofer, Hanspeter Mössenböck, „*A Component Plug-in Architecture for the .NET Platform",* 2007, http://ase.jku.at/publications/2006/JMLC2006_Plugins.pdf

[3] E. Gamma, R. Helm, R. Johnson, John Vlissides, „*Design Patterns: Elements of Reusable Object-Oriented Software*", Addison-Wesley Pub Co, 1995.