University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

The SmartLab: Experimental and Environmental Control and Monitoring of the Chemistry Laboratory

Stephen Wilson
School of Chemistry
University of Southampton
Southampton, SO17 1BJ, United Kingdom
sw1703@soton.ac.uk

Professor Jeremy Frey
School of Chemistry
University of Southampton
Southampton, SO17 1BJ, United Kingdom
J.G.Frey@soton.ac.uk

ABSTRACT

A suite of software applications was developed in order to provide functionality to monitor and record environmental and experimental data and the associated metadata. A piece of software developed by IBM [1] called the Microbroker was used as middleware to handle the flow of messages containing the monitored data using the MQ telemetry transport [2] (MQTT) format. Methods to send control commands were also employed where allowed by the experimental set up. The software was used to monitor a range of laboratories within the School of Chemistry at the University of Southampton. A number of repository solutions were implemented to build an understanding of what is required for a scalable and interoperable system.

KEYWORDS: Data acquisition, Middleware

1. INTRODUCTION

1.1. Why Is It Important to the Chemist?

A typical laboratory notebook or journal article rarely contains all the information required to replicate the experiment reported on. Within the synthetic laboratory, basic information such as full experimental details, work-up procedures and colour changes are poorly reported, if at all. Fundamental information such as laboratory temperature, humidity and movement within the laboratory is almost never reported, although any of these can have an effect on the outcome of the experiment [3]. In analytical laboratories although machine cycle parameters can be reported, the actual values of the machine during the cycle are not known. Additionally, when making data available to other

researchers, for example publishing to a journal, this additional data will provide the readers with the extra information they require to fully understand and replicate an experiment or procedure.

Providing mechanisms to remotely control experiments has been shown to increase the productivity of the chemist and improve accuracy of results [4–8]. Recent developments in analytical equipment, such as auto-samplers, laboratory information management systems and electronic laboratory notebooks have provided automated methods for handling samples, their movement and related results. This has allowed more time to complete experiments and less time on writing up in the laboratory notebook. Additionally, automated systems allow the chemist to prepare a number of samples to be experimented on with a range of input parameters and know, with confidence, that results will be available after a given time period. Remote control systems could allow the chemist to leave the experiment environment; increasing safety, improving accuracy, providing real-time feedback and saving time.

1.2. Middleware [9]

Middleware is the name given to software that connects two components while keeping them independent. In a data acquisition system the middleware provides the connection between the sensors and the repository or output devices. This abstraction between the components makes the system much more scalable. Instead of having to re-code all the existing systems to communicate with a new component, the only requirement is to begin communications with the middleware, see Fig. 1. The use of middleware allows the developer to abstract themselves from the data transfer mechanism, allowing more time producing the functionality of the application.

In this work this means a new sensor or data acquisition card can be added to the system without modifying any other components. The middleware also allows new output systems to be added without modifying the original system. With the support of a well written web service, the system can allow a fully modular data acquisition approach with sub-sections to be switched on and off as required, all controlled remotely.

Figure 1. The Addition of a New Component With and Without Middleware

1.3. IBM's Middleware Solution [1, 10]

IBM's middleware solution is a centralised message brokering system called the Microbroker. This is a publish/subscribe application that provides a link between the source of the data and its destination via a TCP/IP socket connection. Data is published to the Microbroker in a similar way to that of an email; the message is split into two parts, the content of the message (containing the data) and a topic to which the message is related. To receive a published message an application must again create the TCP/IP connection to the broker and subscribe to a topic. When a message is published the Microbroker will determine which clients are subscribed to the message topic and forward the message to those clients. Topics are organised in a hierarchical system using '/' to denote new levels in the hierarchy. A client can subscribe to either a single topic within the hierarchy or use 'wild-cards' to subscribe to a range of topics. The Microbroker can also offer a number of quality of service priorities for each message.

1.4. Data Acquisition and the Labjack UE9 [11]

Standard sensory equipment such as off-the-shelf temperature, humidity and movement sensors were connected to a data acquisition card. The device used in this system was the Labjack UE9; a multi-channel analogue and digital data acquisition card that can also offer a number of output channels. The Labjack UE9 provides both a USB and Ethernet interface. For environmental monitoring the Ethernet interface will be used, connected directly into the Local Area Network (LAN) of the University of Southampton. In experimental set-ups where a computer with network capa-

bilities is already available the USB interface will be used. To make use of all the I/O channels of the Labjack UE9 an additional breakout board is required (CB37) and is connected to the DB37 connector of the Labjack. Several sample applications are provided for the Labjack UE9 covering programming languages such as Java, LabVIEW, MATLAB, Visual Basic and C as well as a test/control panel for Microsoft Windows.

The data captured by the data acquisition card will be published to the Microbroker, undergoing the data life-cycle before it is stored in a central repository. Where possible experimental set-ups will be connected to the system through standard data communication channels such as RS-232 [12] or through a manufactured dongle. The control system will operate through the same Microbroker, publishing messages on a separate topic. These systems will control devices such as laboratory instruments, motors or actuators via a desktop computer.

2. IMPLEMENTATION

The data acquisition and storage system was built in a modular format, each 'agent' subscribing and publishing to the Microbroker using XML formatted messages. The system was broken down into four components: data acquisition, calibration, variation monitoring and storage. The core message pathway is shown in Fig. 2.

The decomposition of the system allowed for faster coding and management of the system and provided the ability to deploy a distributed system. For example time-critical sections can run on dedicated hardware with a global calibration system running elsewhere. Although this system was chosen for data acquisition from the Labjack UE9, the process could be modified for different data sources such as smart sensors; in this case the acquisition and calibration could be combined into a single component. Variation monitoring can be by-passed when full data capture is required such as in experimental monitoring.

Each component will be referred to as an 'agent', this describes a single process running a piece of code rather than that defined by Franklin *et al.* [13]. The complete acquisition system may fall into Franklin's description of an agent; it senses its environment (the sensor data), acts on this (storage into a repository) and effects what it senses in the future (variation monitoring).

2.1. Acquisition Agent

The acquisition agent was developed to query the Labjack UE9 via a socket connection and publish the raw data to the Microbroker. The agent connects to the Microbroker and

Figure 2. The Core Message Pathway for Environmental and Experimental Data Acquisition, Assuming Minimal Bandwidth Usage Taking Less than a Second from Acquisition to Storage

enters a loop where the data is published at a user-defined sample rate. Within the loop the agent sends an eight byte request to the Labjack for the appropriate channel and retrieves the response. This value is wrapped into an XML MQ telemetry transport (MQTT) message and published on the 'raw data' message topic. A Unix time stamp is added to the message at this point so when stored the time of acquisition is used rather than time of storage.

2.2. Calibration Agent

The calibration agent was developed to retrieve raw data, perform a calibration and republish it to a new topic. The agent subscribes to a 'raw data' topic, typical for a given laboratory. When a message is received, the XML is parsed and the sensor type retrieved. This sensor type is used to dynamically look-up the calibration method and the calibration is performed. The units of calibration are also included at this point. This data, along with the sensor metadata, is wrapped back into XML and published to a 'calibrated data' channel.

2.3. Change Agent

The change agent was developed to prevent repetition of stored data. It filters any new data until it has changed by more than a specified threshold for each sensor. The agent subscribes to a 'calibrated data' topic although can be run at any point during the message life-cycle. The agent holds a threshold value for each sensor specified by the user. When a message is received, the sensor type and value are retrieved from the message. This new value is compared to the most recently changed value of that sensor. If the change between sensor readings is greater than the threshold, the message is republished on a 'change topic' otherwise the new message is ignored and the agent

awaits the next message.

2.4. Storage Agent

The storage agent was created to receive the data stream and store it in a logical and structured way. The agent can subscribe to either 'calibrated' or 'change data' channels. When a message is received, the XML is parsed and the data and metadata retrieved from the message. During the development of the project a number of repository solutions were tested and two are discussed below - a MySQL database (2.4.) and the 52° North Sensor Observation Service (2.5.). Research by Wallis *et al.* [14] suggested sensor data can only be trusted by the users if additional metadata such as deployment data, calibration techniques and sensitivities is available. As a result the system attempted to incorporate this.

2.5. MySQL Database

Originally a MySQL 5.0 [15] server running on a Debian [16] distribution of Linux was used. The storage agent stored data as a simple topic, value and date/time format through direct database queries. In an attempt to capture more of the metadata the database structure was modified to contain entries such as building number, room number, data acquisition card channel number, sensor name and units of calibration. The deployment data was included via a supporting website which contained schematics of the room showing sensor positions. Data recall was again handled via direct database queries and formatted as required (typically as a comma separated list).

2.6. 52° North Sensor Observation Service [17]

52° North have developed an implementation of the Open Geospatial Consortium (OGC) [18] Sensor Observation Service (SOS). A SOS provides an API for managing sensors and retrieving data via XML based queries to a Representational State Transfer (REST) interface [19]. There are three core operations the SOS must offer; *getCapabilities* (retrieves all operations and allowed values the SOS offers), *describeSensor* (returns the metadata of a given sensor, typically in SensorML [20] format) and *getObservation* (used to recall data, typically returned as SensorML) along with a number of non-mandatory functions. The SOS uses the concept that an observation is an event that produces a result, the result being an estimate of the observed phenomenon. Each event is classified by a time stamp, a feature of interest, the observed phenomenon and the procedure (sensor). Data is recalled from the SOS through observation offerings, these are non-overlapping groups of related observations such as all results from a given sensor.

The 52° North implementation is programmed in Java and runs on a Tomcat server. During the deployment of the service there were issues when adding sensors. It was determined that all relevant low-level database entries must already exist before the sensor can be registered. The service offers no capability to do this remotely and these must therefore be added through the machine running the server. Once a service containing the sensors and features of interest had been deployed, a second version of the storage agent was developed to capture data in the SOS; this is achieved through a call to the *insertObservation* function via the REST interface. Data recall is also handled through the REST interface using the *getObservation* function, results are returned as full XML and a number of supplementary scripts were generated to gather additional metadata stored in the SOS and link it to the results.

2.7. Data Recall and Visualisation

A system to allow graphical representations of the data was developed within a website framework. After evaluating the available plotting software it was decided that the Simile Timeplot software [21] would be used. Simile Timeplot produces a graphical representation of data and provides details against data points on mouse over, an example is shown in Fig. 3. The Simile Timeplot is generated from a number of comma separated text files and XML documents containing the sensor data.

Figure 3. A Simile Timeplot of 4 Thermocouples from a Laser Laboratory in the School of Physics

3. DEPLOYMENT

The environmental data acquisition was carried out in a number of laboratories throughout the School of Chemistry and School of Physics at the University of Southamp-

ton. During this time an XML dump of messages passing through the Microbroker was taken and converted into a format that could be loaded as a hierarchy schematic. Figure 4 shows the hierarchical approach of the system. Each laboratory is represented as a branch and each branch contains raw, calibrated and changed data for a number of data acquisition channels (and therefore sensors).

Figure 4. The Hierarchy Schematic of the Topics Published during Data Acquisition

Sensor deployments began in December 2007 in a single laboratory within the University of Southampton and several more have since been included. During this period the software has streamed and stored data with minimal down-time. There were three periods of extended down-time when sensor data was not captured. After investigation it was determined that this was due to external sources, for example loss of network during scheduled upgrades or loss of power during building works. The software was designed to handle situations such as loss of power and after rebooting the data stream resumed without intervention. After the first network outage the data stream did not restart automatically, requiring a manual start and stop message. This was due to insufficient error handling by the acquisition agent. The solution was to have a shorter time-out on the data acquisition cards connection than that of the connection to the Microbroker, allowing capture and response to this error and reporting through the Microbroker. After

modification the software is robust to network disconnections.

The acquisition agents were run at a sample rate of once per second for digital acquisition and once every three seconds for analogue acquisition. These rates were chosen as it provided sufficient sampling to capture the slow changing environmental data of the laboratory. The sampling can run faster when required, for example capturing experimental data. The system has currently been tested up to once every millisecond with no problems.

4. CONCLUSION

4.1. Middleware and the Microbroker

The use of the Microbroker allowed for faster development of the basic acquisition system. Small components could be built and tested independently to the overall solution before being integrated. This is most significantly shown by the change agent. Initially the system was developed without this component but after running the system for several days it was determined that much of the data was repeated readings. The change agent was produced to avoid this redundant data capture, on integration the only modification to the system was to change the topic to which the storage agent was subscribed.

As new laboratories were added to the system new instantiations of each agent were launched, subscribing to the appropriate topics. This approach was used to extend the modularity of the system; each data acquisition card having an independent message pathway that can be controlled as required. The Microbroker allows this as each laboratory has its own hierarchical branch through the topics. The Microbroker can also be used to send the start and stop control messages to the appropriate components.

During development a number of message formats were considered for the acquisition data. Originally it was planned to have each message containing SensorML, allowing the metadata data to be included at point of acquisition. The inclusion of full SensorML increased the message size to approximately 2500 bytes per message and therefore required more bandwidth. Messages were modified to contain only the sensor type and data reading, reducing the message size to approximately 75 bytes per message; allowing metadata be added on storage. This minimalistic approach was chosen as this is easier to scale to larger deployments or to networks with reduced bandwidth. Although this approach reduces the load on the bandwidth there is possibility the metadata may be lost or confused on storage.

4.2. Database Solutions

The MySQL database was easy to deploy and tune to the required functionality. New features could be added as required since the handler code was developed at the University of Southampton. The system did not include all the metadata recommended by Wallis et al [14] but could have been included if it was determined to be necessary. During the deployments up to seven million data entries were created over a twelve-month period and during this time no database issues were recorded. The drawback to this system is its lack of a standardised response. As no direct API was developed there were no options for another software solution to request data, dramatically reducing the systems interoperability. This could be resolved by creating a wrapper for the database to provide the required API, although careful consideration into the available functions and the format of the responses will greatly affect its success.

The 52°North SOS provides a rigid structure for handling both the sensor data and its associated metadata. When a sensor is registered with the SOS, a SensorML document is submitted containing data such as manufacturer, full sensor name, observed properties, sensitivities and operational range. This volume of metadata can allow further trust in the data provided there is software to make this available in a user-friendly way. The software runs through an API using the REST interface and uses the Open Geospatial Consortium standard. This functionality should allow external software to communicate and collect data increasing the interoperability. As this is a business solution there is limited availability to modify the software. This means specialist functionality required for this project could not be incorporated, although source code is provided. During the deployment and use of the SOS some minor bugs were found that limited its use, this software is still in development and therefore these issues can be fixed in later releases.

4.3. Further Work

The system is currently being expanded to include more complex experimental data such as where interfacing with a piece of hardware is required; the interfacing code will act as the acquisition agent. The database solution will have to be modified to make the distinction between experimental and environmental data as well as providing a linking system for all related data captured during an experiment. This data along with the associated metadata can then be included in reports and publications, providing significantly greater detail than has previously been reported. The data could also be used to identify experimental failures, for example if an oxygen sensor is attached to an air sensitive

reaction the results can be reviewed to determine if a leak had occurred.

REFERENCES

- [1] IBM United Kingdom, "IBM," [Accessed 12 January 2009]. [Online]. Available: <http://www.ibm.com/uk/>
- [2] MQ - Telemetry Transport, "MQTT.org," [Accessed 12 January 2009]. [Online]. Available: <http://mqtt.org/>
- [3] m.c. schraefel, G. Hughes, H. Mills, G. Smith, T. Payne, and J. Frey, "Breaking the book: Translating the chemistry lab book into a pervasive computing lab environment," 2004. [Online]. Available: <http://eprints.ecs.soton.ac.uk/8780/>
- [4] F. Cardoso, "A universal system for laboratory data acquisition and control," *IEEE Transactions On Nuclear Science*, vol. 47, no. 2, Part 1, pp. 154–157, APR 2000.
- [5] L. Ying, L. Jiarong, L. Guiming, Z. Yingfei, and L. Shi, "The east distributed data system," *Fusion Engineering And Design*, vol. 82, no. 4, pp. 339–343, JUN 2007.
- [6] A. Neto, H. Fernandes, A. Duarte, B. B. Carvalho, J. Sousa, D. F. Varcarel, M. Hron, and C. A. F. Varandas, "Firesignal - data acquisition and control," *Fusion Engineering And Design*, vol. 82, no. 5-14, pp. 1359–1364, OCT 2007.
- [7] J. G. Frey, "Dark lab or smart lab: The challenges for 21st century laboratory software," *Organic Process Research & Development*, vol. 8, no. 6, pp. 1024–1035, 2004.
- [8] J. Robinson, J. Frey, A. Reynolds, B. Bedi, and A. Stanford-Clark, "From the laboratory to the mobile phone: Middleware for laboratory data acquisition using the publication subscribe model," in *e-Science 2005*, 2005.
- [9] G. Banavar, T. Ch, R. Strom, and D. Sturman, "A case for message oriented middleware," in *In Proceedings of the 13th International Symposium on Distributed Computing*. Springer-Verlag, 1999, pp. 1–18.
- [10] A. J. Stanford-Clark, "Integrating monitoring and telemetry devices as part of enterprise information resources," *WebSphere MQ Integrator*, March 2002.
- [11] Labjack – USB/Ethernet based data acquisition and control, "Labjack," [Accessed 12 January 2008]. [Online]. Available: <http://www.labjack.com/>
- [12] EIA/TIA, "Interface between data terminal equipment and data communication equipment employing serial data interchange," August 1969.
- [13] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*. London, UK: Springer-Verlag, 1997, pp. 21–35.
- [14] J. C. Wallis, C. L. Borgman, M. S. Mayernik, A. Pepe, N. Ramanathan, and M. Hansen, "Know thy sensor: Trust, data quality, and data integrity in scientific digital libraries," 2007.
- [15] MySQL :: The world's most popular open source database, "MySQL," [Accessed 23 July 2008]. [Online]. Available: <http://www.mysql.com/>
- [16] Debian – The Universal Operating System, "Debian," [Accessed 12 January 2009]. [Online]. Available: <http://www.debian.org/>
- [17] Welcome to 52North, "52North.org," [Accessed 12 January 2009]. [Online]. Available: <http://52north.org/>
- [18] Welcome to the OGC website, "OGC," [Accessed 12 January 2008]. [Online]. Available: <http://www.opengeospatial.org/>
- [19] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," in *ICSE '00: Proceedings of the 22nd international conference on Software engineering*. New York, NY, USA: ACM, 2000, pp. 407–416.
- [20] Introduction to SensorML, "SensorML," [Accessed 16 February 2008]. [Online]. Available: <http://vast.uah.edu/SensorML/>
- [21] Simile — Timeplot, "Simile," [Accessed 12 January 2009]. [Online]. Available: <http://simile.mit.edu/timeplot/>