Automatization of Acceptance Test for Metrology Algorithms

B. Müller
Ostfalia – University of Applied Sciences
Faculty of Computer Science
Germany – 38302 Wolfenbüttel
bernd.mueller@ostfalia.de

Abstract— *Coordinate Measuring Machines use complex metrology algorithms to compute geometric shapes based on 3-dimensional measuring points. National metrology institutes are committed to validate that results computed by these algorithms are correct. We report on a project funded by the European Union which, beside other topics, develops criteria to assess the fitness for purpose of computational algorithms and software in metrology used by coordinate measuring machines.*

Keywords: Testing, verification, validation, TraCIM, coordinate measuring machines

1. Introduction

Coordinate Measuring Machines (CMM) are used in different manufacturing industries to ensure high accuracy of manufactured products but also high accuracy of the production run itself. The research project TraCIM (*Traceability for Computational-Intensive Metrology*), funded by the European Union, aims for the development of a coherent framework to ensuring traceability in computationally-intensive metrology, a basis for ensuring the trustworthiness and fitness for purpose of metrology software for coming decades. To reach this aim the software and therefor its underlying algorithms have to be verified and validated.

We first introduce CMMs (Coordinate Measuring Machines) and report on the current manual process to check for the correct and high precision measuring processes of such machines. Further, we classify the checking process with respect to the established software engineering concepts of verification and validation. Finally, we describe the architecture and functionality of a system which automates the whole process.

2. Coordinate Measuring Machines

CMM are devices for measuring physical geometrical characteristics of different kind of objects. Maximal permissable error is typically around $1\ \mu\mathrm{m}$. The high accuracy measuring can be achieved by optical, tactile or even computer tomography scanner based capturing of probes. CMMs are hardened against floor induced vibration and are operated in an air conditioned environment to prevent measuring errors.

The capturing of probes differ from conventional measuring. Substitution points get captured and represented as $x/y/z$ coordinates. Based on these substitution points the geometrical forms are computed. Figure 1 shows a circle and the captured substitution points in a plane.

In practice, 3-dimensional geometric bodies such as cubes or cylinders have to be measured and their surfaces or volumes have to be computed. In modern manufacturing industry high accuracy measuring is important to verify that manufactured parts are within designer-specified tolerances and to ensure that manufacturing processes stay in control during the production run.

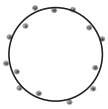

Fig. 1
SUBSTITUTION POINTS OF A CIRCLE

CMM manufacturers therefore have to implement algorithms in some programming language to compute, for example, the circle (diameter, circumference or circular area) depicted in figure 1 out of the substitution points. This can be done with different algorithms — for example least-square, Gaussian and Chebyshev algorithms — and, of course, different programming languages. For an introduction of CMM and used algorithms see [1], [2], [3].

3. Manual Certification Process

National Metrology Institutes (NMI) provide scientific and technical services for industry, science and society. For example, NMIs have to do some certification and support calibration of CMMs to support manufacturing processes of high technology industries.

At the moment the process of certification is done manually by NMIs around the world in a variety of ways. For example, some NMIs own test data sets, which represent substitution points as introduced in section 2. The test data is sent per

e-mail or ground mail (CD/DVD) to the requesting CMM manufacturer. The manufacturer uses the data as input for his metrology algorithms and sents the computed result back to the NMI, also per e-mail or ground mail. The NMI compares the computed result with the expected result and hands over a certificate if the computed and expected results match within some tolerances.

This manual certification process is lenghty, error prone and expensive because of the great portion of human work. Process automation is therefor an evident demand. However, there exist many more requirements, for example concerning traceability which we will detail on in section 5.4.

4. Verification and Validation in Software Engineering

While there is some confusion regarding the terms and definitions of verification and validation across different siences Boehm defines already 1979 in his seminal paper [4] the terms with respect to software engineering:

Verification: to establish the *truth* of the correspondence between a software product and its specification. ('Am I building the product right?')

Validation: to establish the *fitness* or *worth* of a software product for its operational mission. ('Am I building the right product?')

In the figure called *V-Chart* following the above definitions Boehm depicts that verification is done regarding formal requirements while validation ist done regarding customer expectation.

In a complete product development life cycle the transition from validation to verification and vice versa is fluent by nature. Relating to CMM the customer expectation of correct and exact measurement, the implementation of Gaussian and Chebyshev algorithms from the 19th century with some programming language and the embedding into some physical machine has to function correctly as a whole. At the very end there is some last acceptance test resulting in adoption or refusal of the product.

From a software engineering point of view component and integration tests are fully automated while system and acceptance tests are not. It is therefore helpful to look for the characteristics of component and integration tests:

- Test method knows the method to test.
- Test method calls method to test. Both are written in the same programming language.
- Test result is undoubtful.
- Test motivation, test coverage etc. are defined by project conditions.

In contrast TraCIM tests are characterized by:

- Method to test respectivly the environment knows the test data set or test data generator.
- Test is executed randomly and application specific.
- Method or algorithm to test written in some programming language has to obtain test data self-dependent.
- Test result is supposed to be correct but this is not ensured.
- Successful tests lead subseqently to some certification. Therefore test motivation, test coverage etc. are defined by public authorities.

5. Process Automation

Despite long history in formal verification research [5] only very small and simple software systems can be verified correct based on formal methods of mathematics. In practice the only valid choice to get some confidence in proper software operation is testing as introduced in section 4. Some NMIs own test data sets with corresponding test solutions. Some NMIs generate test data sets on the fly and the test solutions are computed, too.

The TraCIM Software Verification System (TraCIM SVS) is part of the TraCIM project (Traceability for computational-intensive metrology). We depict the project further in section 6. A detailed description is also available online [6].

From a software engineering point of view the requirements for TraCIM SVS are quite standard:

- Clients, humans or other software systems ask for some (test) data
- After the data is received some computation regarding the computational coordinate metrology algorithms from section 2 takes place
- The resulting new data (the test result) is send back to the system as the solution for the test data
- After verification of the submitted data there is some kind of result, either success or failure

One of the most popular environments to implement such systems is the Java Platform Enterprise Edition (Java EE) [7]. TraCIM SVS is build with Java EE 7, the most current version. Java EE includes different parts, for example JavaServer Faces (JSF) to build HTML and Web based UIs, Enterprise JavaBeans (EJB) to implement business logic, Java Persistence API (JPA) to persist data to relational databases, JAX-RS to offer REST-like APIs and Context and Dependency Injection (CDI) to glue all the parts together.

The most important technical requirement of TraCIM SVS is the ability to handle all kind of tests, not only the 3D coordinate measurements features described in section 2. Therefore, TraCIM SVS consists of a core system and an innovative extension mechanism illustrated in the next section.

5.1 Architecture and Base Functionality

Figure 2 represents the main components of TraCIM SVS together with the client applications built by the CMM manufacturer. The TraCIM Server core offers REST based services and is hosted by a NMI. Functionality and communication steps are as follows

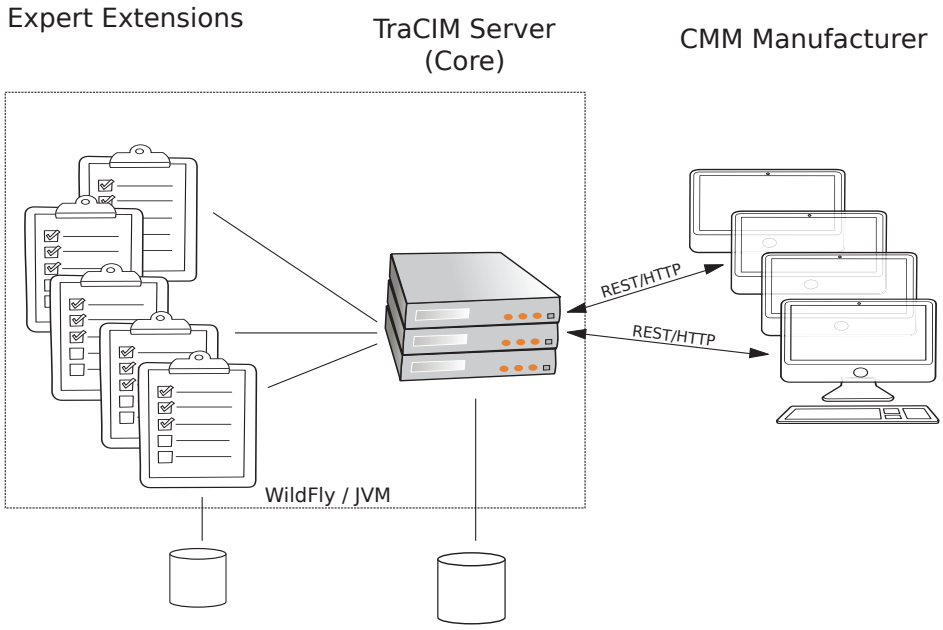

Fig. 2
TRACIM ARCHITECTURE

1) The client has payed the invoice and received some key, which enables him to request for test data. The key encodes also the type of test.
2) The client request TraCIM SVS for test data.
3) Because the request includes the key generated in step 1, TraCIM SVS is capable to identify the requested expert module. This special expert module is called and returns the test data.
4) TraCIM SVS sends back the test data as HTTP response.
5) The client computes the result for the received test data and sends the result back to the TraCIM SVS per HTTP request.
6) TraCIM SVS calls the expert module to compare the expected result for the provided test data and the actual result from the client. This comparison can succeed or fail. In both cases the result of the method call is returned to TraCIM SVS and includes a certificate in PDF in case of success.
7) TraCIM SVS returns the comparison result to the client. TraCIM envelopes the expert extension generated PDF with some administration information from the involved NMI.

As depicted in figure 2 the server stores management information in a database. The kind of management information ranges from CMM manufacturer identification, payment information and the number of remaining tests to memorandums which test data set was delivered to which client, including the time of test data delivery and the time of result submission of the client.

As mentioned earlier expert extension can generate test data on the fly but can also manage a set of static test data and expected test results stored in a database. This optional database usage is also depicted in figure 2. TraCIM SVS does not restrict in any case the inner working of expert extensions.

5.2 Implementation and Used Technologies

Java EE is a well known technology in the area of big application implementation and in widespread use. Java EE is an umbrella specification and consists of about 30 single specifications, depending on the version used. Our project started with version 6 of Java EE but was migrated to Java EE 7 in course of the project. Java EE implementations manifest themselves in so called *application servers*. There are many companies which offer application servers, for example

WebLogic® from Oracle, WebSphere® from IBM and JBoss-AS/WildFly from JBoss. We use JBoss-AS — which was renamed to WildFly in the last version — because JBoss-AS is a so called *open source* implementation of Java EE. Therefore, there are no costs of purchase as well as no annual subscription costs. If in later project stages some demand for commercial assistance will arise, Red Hat the parent company of JBoss offers a commercial licence called EAP which can be subscribed to.

TraCIM server core and expert extensions are implemented as Java EE applications. Because the server core has a JSF based UI it is deployed as a WAR (Web Archive). The expert extensions are deployed as JARs (Java Archive). Both, server as well as expert extensions use some common set of classes. To prevent code redundancy a so called extension base is the third kind of Java EE application we use and is also deployed as a JAR.

The Java EE standard dictates absolute separation of different applications to prevent negative impact from one application to another in case of malfunction. In our case we have the demand that some application modules use some other modules which is not an uncommon requirement in big applications. All application servers offer some kind of non standard mechanism to allow modules to access modules from different applications. This mechanism is usually based on Java's classloader architecture. In TraCIM SVS classes from the extension base are used by the server as well as by the expert extensions. The expert extensions are additionally used by the server.

Finally, TraCIM SVS consists of

- the extension base
- the server core
- one or more expert extensions

If a CMM manufacturer wants his software and in turn the complete CMM to get certified, he has to pay the mandatory fee for responsibilities of public administration and get the authorization to get test data sets and submit in turn test results for these data sets.

This is done by REST requests (REpresentational State Transfer), the most up-to-date interpretation of web services. The details about the communication steps are already described in section 5.1.

The most innovative aspect of the system architecture is based on the extension mechnism for expert extensions which is similar to plug-in architectures. If a new expert extension is implemented and has to be integrated into the system, no code change has to be accomplished. This is possible because of Java's concept of a *service loader* which was introduced in Java 6 and manifests itself in the class `ServiceLoader` [8]. The mechanism is based on a simple convention which results in a self publication of classes implementing a particular interface. The class `ServiceLoader` can then be asked for all known implementation of the particular interface.

5.3 Future Enhancements

Because we describe here some work in progress there will be of course future enhancements. At the moment we are working on a design enhancement to allow expert extensions to run as separate server services. If, for example, some NMI X hosts the TraCIM server core but the expert extension runs on behalf of NMI Y on a different server, probably in a different country the collaboration of TraCIM server core and expert extensions has to be revised to reflect this requirement. The generated certificate has also to reflect this separation of responsibilities. It has to contain a functional part of the NMI offering the expert extension but also a more administration part of the NMI hosting the TraCIM server core which reflects the contractual relationship between the NMI and the CMM manufacturer. This point directly passes over to some legal aspects.

5.4 Legal Aspects

Because certificates were assigned by public authorities there are some legal consequences. The performed tests and certifications have to be repeatable and traceable. Repeatable means that if a CMM manufacturer has requested a particular kind of test and has succeeded this test a consumer of the CMM can ask many years later for a further test. It has to be guaranteed that the consumer will get the same test data set as the manufacturer many years before to ensure that the outcome of the same submitted test results are the same.

Based on the same rationals and the responsibilities of public administration all test processes and test results have to be stored for decades to establish a complete chain of evidence if some disaster happens because of some earlier certification of wrong or even right test results.

6. Project and Project partners

The European Community has established the research project *Traceability for Computationally-Intensive Metrology* (TraCIM) which - beside other topics - develops criteria to assess the fitness for purpose of computational software in metrology and to verify them. The TraCIM home page [6] details objectives of this research project. The project started in 2013 and will be finished in 2015.

The national metrology institutes of the United Kingdom, Czech Republic, Italy, Germany, Slovenia and Netherlands as well as 4 CMM manufactures and 3 Universities belong to the project consortium.

Ostfalia, University of Applied Sciences, located in Germany is responsible for implementing the project supporting software and therefore TraCIM SVS. Close collaboration takes place with PTB, the German NMI.

Some NMIs are working on different expert extensions at the time to complete the bunch of possible test data sets for different aspects of CMM characteristics. At the moment

Guassian, Chebyshev and Intercomparison are available and offered by PTB.

7. Conclusion

We reported on the software system TraCIM SVS which main task is to support national metrology institutes to proof and certify the correct working of CMM. CMM are an important part of manufacturing processes in modern industry.

Workshops in fall 2013 and spring and fall 2014 with the CMM manufacturers of the TraCIM project demonstrated the capacity of the design and implementation path we have chosen. At the moment two manufacturer's CMM software was certified by PTB, the German NMI, in a fully automatized process based on TraCIM SVS.

Acknowledgment

This research was undertaken within the EMRP project NEW06-REG3 TraCIM. The EMRP is jointly funded by the EMRP participating countries within EURAMET and the European Union.

References

[1] V. Srinivasan and C. M. Shakarji and E. P. Morse, *On the Enduring Appeal of Least-Squares Fitting in Computational Coordinate Metrology*, Journal of Computing and Information Science in Engineering, March 2012, Vol 12.

[2] T. H. Hopp and M. S. Levenson, *Performance-Measures for Geometric Fitting in the NIST Algorithm Testing and Evaluation Program for Coordinate Measurement Systems*, Journal of Research of the National Institute of Standards and Technology, 9/1995.

[3] C. Shakarji, *Evaluation of one- and two-sided Geometric Fitting Algorithms in Industrial Software*, Proc. American Society of Precision Engineering Annual Meeting, 2003.

[4] B. W. Boehm, *Guidelines for verifying and validation Software requirements and design specification*, Proc. European Conference of Applied Information Technology of the International Federation for Information Processing, London, 1979.

[5] E. M. Clarke and E. A. Emerson and A. P. Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications*, ACM Transactions on Programming Languages and Systems, Vol 8, Apr. 1986.

[6] *Traceability for Computationally-Intensive Metrology*, http://www.ptb.de/ emrp/1389.html.

[7] *Java Platform, Enterprise Edition*, http://www.oracle.com/technetwork/java/javaee/overview/index.html.

[8] *ServiceLoader Documentation*, http://docs.oracle.com/javase/7/docs/api/java/util/ServiceLoader.html