Reprinted from PHARMACEUTICAL ENGINEERING®
The Official Magazine of ISPE
May/June 2010, Vol. 30 No. 3
www.ISPE.org

Guide for using Open Source Software (OSS) in Regulated Industries based on GAMP

This article demonstrates the benefits of employing Open Source Software (OSS) in compliance with GAMP for the validation of computerized pharmaceutical applications.

by Markus Kaufmann, Marcus Ciolkowski, Andreas Hengstberger, Till Jostes, Erwin Kruschitz, Thomas Makait, Karl-Heinz Menges, Stefan Münch, and Martín Soto

Introduction

Differences between Open Source Software (OSS) and Commercial-Off-The-Shelf (COTS) software are smaller than many stakeholders perceive. This article outlines the benefits of free and Open Source Software (OSS) in contrast to commercial software and the resulting approach to qualification and operation of OSS in GxP-regulated areas.

Commercial software is generally assumed to be traditionally developed and marketed by a company and often requires royalties. Typically, source code is not disclosed and the customer has no right to modify the software.

Free and open source software are terms for software which comes with certain rights or freedoms for the user. The rights vary dependent on the chosen license. They typically allow access to and modification of source code and free redistribution.

One main driver for using OSS is the reduction of business risk, due to the open nature of the software; all details of a system can be verified and reviewed in detail. On the other hand, today's validation methods need to be adapted to address aspects such as:

- Communities are not conventional (commercial) business partners. Communication and ways of working may be different, e.g., service level agreements are not available or the community might not want to work with you for some reason.
- As conventional supplier audits cannot be performed on OSS communities, other methods for vendor evaluations must be applied.

Free and open source software is already an essential part of today's software industry. Many business critical applications already exist, continuously gaining momentum. However, most installations of software systems containing OSS to a certain degree, use it as part of "Infrastructure software" (as per GAMP 5). They consist in part or in its entirety of OSS (OSS Applications) by using LINUX and MYSQL databases. Especially for such infrastructure software, there is a high potential for economic savings.

International regulatory bodies generally do not distinguish between commercial software and OSS;[2] therefore, companies are free to choose either one. All GxP regulated computerized systems must be validated prior to use to show that the system is fit for the intended purpose. Decisions on data integrity controls and the extent of validation should be based on a documented and justified risk assessment. Impact on patient safety and product quality as well as data integrity/availability should be evaluated during this process. Clear acceptance criteria should be defined based on this risk assessment and documented in the validation plan.

Specific Nature of Open Source Software

One common misconception about OSS is that it is produced exclusively by altruistic individuals and that, consequently, it should always be free of cost. However, both the development and the deployment of OSS involve significant effort investments that cannot be reasonably ignored. Understanding the business and license model, development processes, and support structure is fundamental to make appropriate business

decisions regarding OSS. Parties involved include OSS communities, commercial software firms, system integrators, consultants, users/health care industries, and regulators.

The Business Models

"How are those developers supposed to earn money?" – Just like any other software development project, an OSS project requires effort directly related to the size and complexity of the intended product. In commercial software endeavors, the work is done by paid developers with financial resources being provided by customers through a direct development contract or some sort of licensing model.

In the case of OSS projects, development effort is mainly contributed by:

- Individuals who spend part of their free time working on the project.
- Companies and other organizations that directly or indirectly commit paid programmers or other professionals to work on the project.

The motivation for individuals in the first group can vary widely and range from being excited by technical challenges over having altruistic motives (e.g., OSS may help combat poverty and inequality) to attempting to build their personal reputation as developers by making valuable contributions to a visible project. Companies and freelance professionals, on the other hand, are rather interested in developing new business opportunities around their OSS involvement. A number of business models are possible:

- Extending or improving OSS and contributing back to the corresponding project as a cost-effective make or buy alternative.
- Offering paid services for an OSS product. This may include enhancing or correcting the product on a contractual basis for customers.
- Turning a commercial product into OSS. This can increase product adoption, creating new business opportunities for the company.
- Offering a product under dual licenses. In such cases, a product is made available under a commercial license and as OSS. The OSS license is typically restrictive in some aspects (e.g., it does not allow integration of the product into commercial products) while the commercial license lifts these restrictions. This way, the company can exploit the benefits of a community (e.g., external contributions, large user base) and still provide some additional paid services.

In summary, many OSS projects are motivated by business considerations at least to a certain extent. This trend is likely to continue in the following years, underpinning the increasing interest in OSS.

Legal Aspects

OSS is free of (license) costs, but not free of any liabilities. Therefore, legal aspects need to be considered when deploying OSS. When discussing legal aspects, we need to strictly differentiate the contractual side (e.g., warranty) from the intellectual property (copyright) side of the subject.

Let us first take the perspective of a typical software user: in case the intended use of OSS is acquiring, installing, and running (typically the case for Linux, MySQL, or Apache) without changing the software, the legal situation is not much different compared to the situation of buying commercial software. Even if the source code of the software will be changed in order to adapt the software functionality to the requirements there are normally no legal implications, as long as it is not distributed to others. However, the user should be aware that with using OSS, a contractual relationship is potentially entered that may encompass contractual and copyright aspects. As with commercial software, the purchaser should ensure that the supplier grants software to be free of rights of third parties.

The perspective of a software supplier or integrator is, compared to a normal OSS user, more complex: OSS license conditions and additional country-specific legislation need to be considered when OSS software is used as a basis for developing new software which is being distributed later. Depending on the OSS license model, the distributor is subject to some OSS specific obligations. For example, the BSD-type licenses (e.g., Apache Server) allow modifications of the OSS software and commercial distribution without disclosure of the source code. On the other hand, the GNU General Public License requires the (modified) source code to be published. It also requires that the source code of software, which has been combined with OSS – Software, needs to be published. In case of doubt, professional advice should be taken into account.

Development Process

The typical OSS development process involves a group of loosely coordinated developers, who work in parallel on new features or corrections. In many cases, a versioning system (such as CVS or Subversion) is used to manage the product's source code. Usually, only a small number of trusted developers (called *maintainers*) have write access to the source code repository, being able to make changes to the main development line. However, other contributors can make a copy of the code (for example, through public, read-only access to the repository) and develop their own contributions based on those copies. They submit a file containing their changes (*patch file*) to the maintainers, who review it, and if deemed adequate, integrate it to the main code branch.

Branching may lead to separate development streams. Those may later be merged again to the main development line or may lead to a split of the project (*forking*).

Mature OSS projects often have a number of processes in place that support activities, such as requirement management, release management, issue reporting and tracking, software distribution, software testing, and as mentioned before, version and configuration management. These processes are usually enforced by a combination of software tools

and community activities. Openness and transparency of the community are key factors.

Support and Maintenance

An active community around an OSS product also increases the chance of obtaining free support for that product. In many cases, users of an OSS product can submit questions to open mailing lists or Internet forums related to the product. Also many projects provide an issue tracking system that everyone can use to report problems or suggest enhancements. However, it must be taken into account that these free support channels are not guaranteed to work. A question sent to a mailing list may not be answered or a reported problem may remain unsolved for a long time. Reasons include that knowledgeable people do not answer, nobody in the community knows the solution, or the request or problem is not even considered to be relevant. Even in a project with a large, active community, such a situation may still occur.

For organizations requiring guaranteed support, there are options, such as contracting a company or freelance developers or providing in-house support by hiring experienced developers and system administrators.

Customer–Supplier Relationship

In contrast to the well-known customer–supplier relationship of commercial software, OSS offers several ways of interacting with a community or supplier. The list below shows typical scenarios.

- Scenario (a): customer downloads, installs, and uses OSS application "as is," the application is not changed by customer or customer's IT. In this scenario, the customer is not part of the community.
- Scenario (b): application is supported and maintained (and may be customized) by a system integrator or other supplier. Depending on the license model, the integrator or supplier may be part of the community by contributing code.
- Scenario (c): as (b), but customer's IT department acts as facilitator and supports and maintains the application, customer or customer's IT may change code. Here, depending on the license model, even the customer may become part of the community.

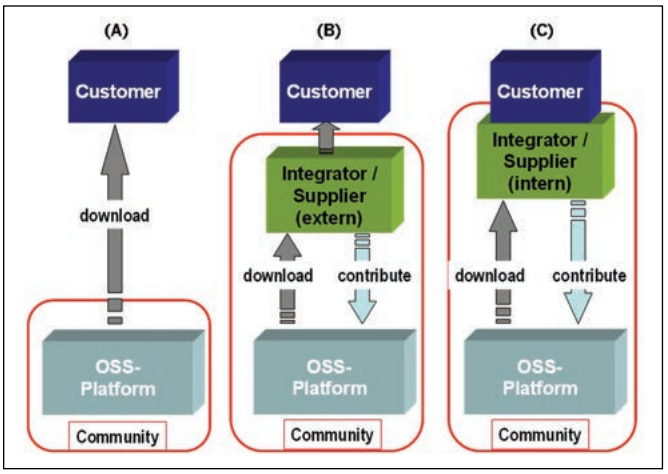

Figure 1. Typical scenarios for Customer-Supplier Relationships.

These three scenarios are prototypical; in real life, different variations and combinations can be found.

The selection and institutionalization of one of the above-mentioned scenarios is of major importance, as they have implications on the role of the OSS customer. For example, if removing defects and adding features is critical to the application of the OSS software, it may be advisable to hire an external supplier (scenario b) or contribute internal developers to the project (scenario c) in order to guarantee quick resolution of issues. Furthermore, if the modified OSS product is given to third parties (e.g., as part of an embedded system), it may even be necessary to contribute the changes to the community, depending on the OSS license. Although it may seem strange on first sight to invest into something that potentially benefits other companies, the support and product improvements received from the OSS community can still pay off.

Distribution Channels

For distribution, several channels are available to OSS developers.

The most common ones include:

- *Direct Internet download.* In some cases, OSS projects provide and manage their own internet servers (often with financial support from company contributors). Many projects, though, rely on Web sites offering generic services for OSS projects, such as SourceForge or Launchpad. Software distributed through such generic channels is often only available in source code form.
- *Operating System Distributions.* Distributions combine an operating system kernel (such as Linux or BSD) with utility and application software to produce a usable, integrated system. Since distribution developers (operating system distributions are OSS projects by themselves) normally put effort into making the various components work together, distributions are often the most convenient way to install Open Source products. Also, in most cases, distributions provide precompiled, ready-to-run code for popular hardware architectures. Examples of (Linux) system distributions are Fedora, SUSE Enterprise Linux, and Ubuntu.
- *Software Download Web sites.* Web sites specializing in (often commercial) software downloads (e.g., Tucows), are offering a continuously growing selection of OSS.
- *Software Collections.* Software collections, such as those often distributed by computer related magazines, often contain OSS.

The main risk associated with such distribution channels is that of inadvertently downloading versions of the OSS product that have been modified by malicious third parties. This risk can be minimized by using the "official" community server or platform, as the download is in many cases secured by checksums and/or digital signatures to verify its authenticity.

Lifecycle Approach for OSS Systems Based on GAMP

The same regulatory expectations apply to software in the GxP environment regardless of whether that software is commercial or OSS.

Achieving and maintaining compliance of GxP regulated systems containing OSS components can be *generally* performed according to GAMP Guidance in the same way as commercial software. The differences and the adaptation of the life cycle approach for OSS will be described in this section. The generic project life cycle described in GAMP 5 is applied as a general framework.

Table A lists GAMP 5 categories and gives examples for OSS applications. In the rightmost column, recommendations for life cycle activities are presented. Some suggestions for different life cycle approaches strategies also have been outlined in Table A.

Concept Phase

The purpose of the concept phase is to prepare the project to have the appropriate resources in place.

Project/Validation Plan

The project plan defines work products to be developed; life cycle model and approach to be used; customer requirements related to project management; tasks to be accomplished; task ownership; project resources; schedules, milestones, and target dates; estimates; and quality criteria. In addition, the plan identifies critical dependencies; required work products; project risks and risk mitigation plan; and contingency actions for non-completed tasks.

As for any validation project, the validation plan should include at least significant background information, the objectives of the project, the responsible personnel, description of SOPs to be followed, standards and criteria for the relevant processes; and predetermined acceptance criteria for drawing conclusions.

Requirements/Specifications

Depending on the development lifecycle used for the software, specifications are handled in different ways. Sufficiently defined requirement specifications should be available. Specifications should describe the process supported by the software with inputs and outcomes. Of course, the level of detail also depends on the GAMP category as well as risk, complexity, and novelty. Technical specifications like functional specifications or design specifications have to be in line with the chosen lifecycle model.

Project Phase

Vendor/Supplier Evaluation

Due to the specific nature of OSS and the different customer–supplier relationship, the vendor and supplier evaluation is probably the most challenging task with major differences to commercial software.

For low risk applications, a supplier evaluation is not required. For medium to high risk applications, scenarios (b) or (c) are recommended, because a service organization – whether internal or external – will reduce the uncertainties and risks of support and maintenance by a community.

For scenario (b), the vendor evaluation is the same as for commercial products. For scenario (c), internal quality standards apply, and the internal supplier needs to evaluate the community according to scenario (a). The rigor of evaluation should be commensurate with the risk priority.

Category	Software Type	Examples	Life Cycle Approach
1	Infrastructure	Core system and utilities of major commercial Linux distributions like embedded systems (e.g., firmware, networking software, Linux, BSD operation system, Apache HTTP Server, MySQL), and established layered software (e.g. OpenOffice, Firefox, DIA)	Standard GAMP approach, provided that a legal entity exists to maintain the software, i.e., providing services along the SDLC that can be considered a supplier in terms of the GAMP guidance.
2	N/A	N/A (Formerly firmware, obsolete since GAMP 5.)	N/A
3	Non-configured	RANDI2	Standard GAMP approach, provided that a legal entity exists to maintain the software, i.e., providing services along the SDLC that can be considered a supplier in terms of the GAMP guideline. Depending on the risk, vendor evaluation may be different, see section 3.3.
4	Configured	Alfresco (CMS), Compiere (CRM/ERP)	Standard GAMP approach, provided that a legal entity exists to maintain the software, i.e., providing services along the SDLC that can be considered a supplier in terms of the GAMP Guidance. Supplier audit against the criteria laid out for OSS. The scope and depth would be dependent on the intended use of the product.
5	Custom	The authors currently do not know of custom software directly supporting pharmaceutical processes that were released as OSS. Custom parts added in the course of a project should be regarded as custom code. If such software would be released as part of an OSS product after initial testing and verification, the category would decrease as a consequence.	The standard GAMP approach for validation of OSS applications should be applied.

Table A. Life cycle approach.

If a community is selected as a direct source (scenario (a)), suitable quality criteria to measure the stability and quality of a community have to be applied. A general assessment checklist for OSS is provided by GAMP 5, accompanying material (Example Checklists and Questionnaires).

The sustainability of an OSS community can be evaluated using the following common criteria, which shall be considered when conducting a supplier assessment.[4] The selection of criteria is case-dependent and should be documented.

- Activity of the community:
  - Number of downloads
  - Number of developers
  - Number of contributions
  - Activity of mailing lists and discussion forums

- Personal profiles:
  - Experience of key developers/maintainers (an important motivation for OSS developers is earning acceptance by the Community)

- Communication:
  - Mailing lists
  - News groups

- Organizational structures within the community,[1] such as:
  - Project management
  - Definition of a core team
  - Maintainers of subprojects

- Configuration Management:
  - Definition of the process of review of contributions and integration into main branch
  - Definition of the process for forking off stable releases
  - Version control
  - System and release documentation

Additional criteria can be found e.g., at FlossQuality.[14]

Development Standards

Development standards appear to be a major challenge for OSS, but in fact many communities have standards in place. Depending on the license model, following standards is imperative for reusing and modifying the code. This should be a criterion for the vendor evaluation.

Traceability

Relations between URS, FS, development specification, and tests should be traceable. For example, if you hire parts of a community to extend or build an application, either you or the hired contractors from the community have to provide traceability as described in GAMP 5.

Risk Assessment

Risk assessments can be performed on various levels. A high level general risk assessment may take system complexity and risk as basic inputs for a life cycle activities strategy.

In case of complex systems, a detailed risk assessment on function level that includes configuration and coding might be useful to target or reduce testing effort. As a means to a risk assessment, EN 60812 (which describes Failure Mode and Effects Analysis (FMEA) and Failure Mode, Effects, and Criticality Analysis (FMECA)), and GAMP 5, which also applies functional risk assessment approach may be applied, but fault tree analysis and other methods (e.g., refer to ICH Q9) are acceptable as well. However, remember that the overall idea is not to measure risk, but to identify and manage risk.

Also, keep in mind that the methods for the risk assessment of commercial software and OSS are the same – and so is the consequence of a failure – but the mitigation methods covered later on by the risk management process are slightly different. For example, for OSS additional code review is a mitigation method that usually cannot be applied to commercial software, whereas additional testing for highly critical functions can be used for either system.

Implementation

The purpose of the implementation phase is to build the system; therefore, this phase applies to SW of category 4 and 5 only. Here, the difference between commercial software and OSS is obvious and the different license models (see section 0) need to be taken into account:

- If you use the SW directly as provided by the community (scenario a), it will most likely not be SW category 5. With respect to accountability and liability, additional risks may apply.
- If you purchase the OSS through a supplier (scenario b), maybe including source code or configuration changes supplied by the community, you should expect the same processes and tests like for commercial software.
- If you change or configure the SW by your own (scenario c), you may be (or become) part of the community when distributing your software to others (Copyleft). If you use the SW for your own purposes only, Copyleft does not apply. In the first case, you have to follow the community's rules, in any case, you should define your own internal processes and tests for your own work. This may include raising staff awareness of OSS license issues (see section 2.2) and maintaining appropriate license documentation. It is advised to check whether a change, extension, and/or redistribution of the software is planned. This is particularly important with software components under GPL-style licenses.

Depth and rigor of testing should be commensurate with the identified risk.

Installation

The installation process comprises more activities than the Installation Qualification (IQ): you have to have a description for the installation process with prerequisites and, e.g., hard-

ware requirements in place. The IQ might follow this process and documents, proving that it is performed appropriately alike for all other software, too.

Acceptance Testing

Regardless whether you follow the V-model or any other type of development model, acceptance testing is a must for all SW categories other than 1, hence for OSS, too. The same methods to perform and document acceptance testing for commercial software can be applied to OSS as well.

Training

Like for all systems, users and system administrators have to be trained to be fit to use the system. The recommendation is to use internal or external training resources to perform the required training activities.

Validation Report

At the end of the project, the system has to be handed over to operations. Both parties have to agree about the status and the acceptance of the system as it is. The appropriate activity to document project closure is the validation report. The validation report defines the end of the project from a quality point of view and releases the system for use. Prerequisites for the release are the successfully performed activities defined in the validation plan. The validation report for OSS does not differ from validation reports of other systems.

Operation Phase

The purpose for the system operation phase is to supply a system for the users.

Service Desk and Incidents

For all applications, including OSS, a responsible person or organization as single point of contact should be defined. Similar for commercial software, this person/organization manages incidents. Since a community has not the required availability and may even disappear in the future, a community based services desk poses additional risks. Therefore, internal resources or a supplier with a binding contract need to be in place.

For systems with high GAMP categories and high risk, suppliers with a legal binding contract (SLA) should be the way of choice. Systems with very low risk may be used without a defined service from a GMP point of view. For systems in between, the service might be based on communities without legal binding contract, but with an evaluation of the community.

Contracts such as SLAs should be in line with given standards such as COBIT and not differ from those used for commercial software.

Deviations/Problem Management

A process for deviation and problem management should be defined. Although communities may provide these problem solving services, the management of tracking and classifying incidents can be done on the vendor's side only. As before, it is recommended to set up internal resources or establish a binding contract with a supplier.

When working with a community, a bug tracking system as well as a feature and support request page should be available. These communication mechanisms must be integrated in the processes of your internal or external service supplier.

Change and Configuration Management

Change and configuration management takes place at two sides:

- Customer/user: Like for other applications, the regulated company has to implement a change management system for OSS applications.
- Vendor/community: The vendor or community need to provide sufficient underpinning/detailed data to allow change management on the customer/user side and allow changes to the software to be tracked. Depending on the complexity of the system, the methods may differ, e.g., release notes on patches and/or source code commenting are sufficient. Own the change management process to manage and control changes.

The service organization, whether internal or external, has to monitor the activities of the community. Bug fixes and releases have to be evaluated and implemented if appropriate. So the general approach is not different when compared to commercial software, but OSS updates may come with more details, thereby supporting risk assessments.

Maintaining the Validated State

In general, there are two kinds of evaluation:

- After changes to the system: As integral part of the change control process, impact and risk of every change needs to be identified, evaluated, and managed.
- Periodic review to ensure the current capability of the system.

Like with all other systems, frequency and extent of periodic reviews should be determined by a risk-based approach and a review of historical data.

Retirement Phase

Again, system retirement is nothing specific to OSS. The most important requirement is to manage and provide the data throughout the retention period. For OSS, data formats and algorithms are available and may support data analysis and migration.

Summary and Future Trends

While Open Source Software is often perceived as being completely different from traditional commercial software, the consequences for regulated industries turn out to be rather minor. The differences affect the licensing models and development processes, and may in turn influence vendor selection and service processes. However, the life cycle activities remain

basically the same, and the risk-based approach promoted by GAMP 5 is still applicable.

Consequently, there is no reason to generally exclude usage of OSS in regulated industries. Increasingly, OSS applications emerge that can be an alternative to commercial software. As an example, the Open Document Format (ISO/IEC 26300) provides a format independent from applications, and is an excellent solution for documents which have to be archived or exchanged between different systems. Up to now, only Open Source applications are supporting this format.

Other regulated industries like aerospace companies are far ahead, already using OSS even in critical areas.

References

1. Dietze, S., Modell und Optimierungsansatz für Open Source-Softwareentwicklungsprozesse, 2003.
2. Bell, B.S., et al., Times "R" are A Changing, FDA Perspectives on Use of "Open Source," 2006, http://www.fda.gov/cder/Offices/Biostatistics/Bell.pdf.
3. Pharmaceutical cGMPs for the 21st Century – A Risk-Based Approach, Final Report – Fall 2004, http://www.fda.gov/cder/gmp/gmp2004/GMP_finalreport2004.htm.
4. Ripamonti, L.A., DeCindio, F., and Benassi, M., "Online Communities Sustainability: Some Economic Issues," *The Journal of Community Informatics*, Vol. 1, No. 2, 2005, p. 6378, http://ci-journal.net/index.php/ciej/article/view/221/180.
5. WHO Expert Committee on Specifications for Pharmaceutical Preparations, 2006, p.143.
6. Volume 4 – EU Guidelines to Good Manufacturing Practice – Medicinal Products for Human and Veterinary Use, 2008, p. 111 ff.
7. 21 CFR 11, Electronic Records, Electronic Signatures, 2008.
8. 21 CFR 211 Current Good Manufacturing Practice for Finished Pharmaceuticals, 2008.
9. Guidance for Industry – Part 11, Electronic Records; Electronic Signatures – Scope and Application, 2003.
10. ICH Q7, Good Manufacturing Practice Guide for Active Pharmaceutical Ingredients, 2000, http://www.ich.org.
11. PIC/S Guidance Good Practice for Computerised Systems in Regulated "GXP" Environments, 2007, http://www.picscheme.org.
12. EN 60812:2006-11, Analysis Techniques for System Reliability – Procedure for Failure Mode and Effects Analysis (FMEA).
13. Flossquality, http://www.flossquality.eu/.
14. *ISPE GAMP® 5: A Risk-Based Approach to Compliant GxP Computerized Systems*, International Society for Pharmaceutical Engineering (ISPE), Fifth Edition, February 2008, Appendices D5 and M5, http://www.ispe.org.

About the Authors

Markus Kaufmann is certified SPiCE/ISO 15504 assessor and works as Global QA/e-Compliance Manager at Novartis. He holds a degree in process engineering from the University of Karlsruhe and started his career in 1996 as a quality engineer at Nestle. Since 1998, he has been working in GMP-regulated industries for several pharmaceutical companies (e.g., Pfizer, Roche, and Bayer Healthcare) in various positions. His early experience was with scale-up; however, he continued to gain experience in automation, IT quality management, and IT project management. He can be contacted by email: markus-1.kaufmann@novartis.com.

Novartis Pharma AG – PHQA, WSJ360.15.29, Novartis Campus, 4002 Basel, Switzerland.

Marcus Ciolkowski received an MS in computer science from the University of Kaiserslautern, Germany in 1999. Since then, he has been working as a research scientist at the Fraunhofer Institute for Experimental Software Engineering (IESE) and at the Software Engineering Research Group at the University of Kaiserslautern. His research interests include empirical methods for software engineering and quality management. He has participated in several research and industrial projects in the areas of quantitative model building, quality assurance and improvement, inspections and reviews, as well as design and analysis of empirical studies. He is project manager at Fraunhofer IESE for the European QualOSS project, which aims at building a quality assessment method for Open Source projects. He can be contacted by email: Marcus.Ciolkowski@iese.fraunhofer.de.

Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany.

Andreas Hengstberger is Quality Manager with Nycomed Austria GmbH, where he is responsible for computerized systems validation and the global electronic quality management system. He started working for the pharmaceutical industry as a computerized systems specialist at Boehringer Ingelheim in 2000 being involved in qualification and validation of systems in biotech and pharmaceutical sites. Hengstberger has been working with Free and Open Source Software for many years in the educational and professional environment. He joined the GAMP Special Interest Group on Open Source Software in 2006. His background is in occupational safety, information technology, quality management, and chemistry from TU, TGM, and HTL17 Vienna. He can be contacted by telephone: +43-70-6919-4637 or by email: andreas.hengstberger@nycomed.com.

Nycomed Austria GmbH, St. Peter Strasse 25, Postfach 122, 4020 Linz, Austria.

Till Jostes studied electrical engineering and information sciences at the Ruhr-University Bochum. His first engagement was in the healthcare industry as a manager of development, sales and service of hand held computers for ambulant nursing services. Sales engineer at Xerox, Germany, focused on digital high speed printing in the pharmaceutical and finance industry. Since 2001, he has been a senior consultant and specialized in the computer system validation for the medical device and pharmaceutical industry. He has published in several German publications and conducted lectures on topics around computer system validation and electronic signatures based on biometrics. Since 2005, he has been an active member of the GAMP DACH Forum, partially leading the GAMP Special Interest Group on Open Source Software. He was involved in the review of GAMP 5 and contributed to the GAMP 5 attachments. Jostes can be contacted by email: Till.Jostes@btconsult.de.

btconsult GmbH, Europaallee 27-29, 50226 Frechen, Germany.

Erwin Kruschitz, Anapur AG, has been working in the field of control systems and IT for the past19 years. After starting as an automation engineer, he led a software integration department responsible for batch, process information management, and data integration. Since 1996, he has been involved in qualification and validation of Manufacturing Execution Systems (MES) and automation systems for the pharmaceutical industry. He has attended audits and has led several assessments himself, some of them in cooperation with the TÜV (German technical inspection authority). Anapur AG is an engineering and qualification services company serving mainly large companies in the process industries. Currently, Kruschitz is the person in charge for GMP and IT-security assessments within Anapur AG. He can be contacted by email: e.kruschitz@anapur.de.

Anapur AG, Donnersbergweg 1, 67059 Ludwigshafen Germany.

Thomas Makait has more than 19 years of chemical and pharmaceutical process industries experience. His main focus was on design, implementation, commissioning and operation of Process Control, Manufacturing Execution, Laboratory Information Management and Enterprise Resource Planning Systems, before he moved on to his current position as Quality Assurance Manager for computerized systems at a biotechnological API production site in Frankfurt am Main/ Germany. During his career he worked as Lead Engineer for a major international Engineering, Procurement and Construction company and as Management Consultant for one of the leading international management consulting/commercial auditing firms. Makait is certified ISO 31000/ONR 49000 Risk Manager, certified ISO 9001 Quality Management Systems and ISO 27001 Information-Security-Management-Systems Auditor. He can be contacted via t.makait@qpri.de.

Karl-Heinz Menges studies Pharmacy at the University of Heidelberg. He has more than 25 years of experience as a GMP inspector and has inspected pharmaceutical companies in Germany and abroad. Menges is a member of PIC/S expert circle computerized systems, GAMP DACH Steering Committee, and head of German inspector's expert group on computerized systems. He has been a contributor to the APV Guideline, PIC/S document PI-011, ISPE's GAMP Guidance, including several GAMP Good Practice Guides. He can be contacted by email: K-H.Menges@web.de.

Regierungspraesidium Darmstadt, Pharmazie, 64278 Darmstadt, Germany.

Stefan Münch has more than 10 years of working experience in leadership roles in life sciences for MES applications. He is currently the Campus Quality Manager, leading the quality and test team of Rockwell Software's campus Karlsruhe. As an active member of ISPE GAMP DACH, he was a speaker at the ISPE Copenhagen Conference in 2006. He graduated in computer science from the University of Karlsruhe, Germany. In 1997, he joined Rockwell Automation (formerly: Propack Data) and gained experience in software and test engineering as well as in quality and project management. Münch can be contacted by e-mail: smuench@ra.rockwell.com.

Rockwell Automation Solutions GmbH, Zur Giesserei 19-27, 76227 Karlsruhe, Germany.

Martín Soto received his MSc in computer science in 1995 from Los Andes University, Bogotá, Colombia. He worked as a software developer for Liikkuva System Intl., a GIS systems integrator, and later as instructor at Los Andes University, where he taught in the areas of programming and software engineering and worked as industrial consultant for software process improvement. Since June 2000, he has been a member of the Process and Measurement Group at the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany. His current research interests include software process comparison and version management and processes in Open Source software development. He can be contacted by telephone: + 49 (0)631-6800-2214 or by email: Martin.Soto@iese.fraunhofer.de.

Fraunhofer Institute for Experimental Software Engineering (IESE), Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany.