Bringing governance into distributed R&D software development - GÉANT case study

Authors
Branko Marović, Paweł Kędziora, Marek Lewandowski, Cezary Mazurek

Author affiliations
Branko Marović
RCUB - Belgrade University Computer Centre, Kumanovska 7, Beograd 126119, Serbia,
e-mail: branko.marovic@rcub.bg.ac.rs

Paweł Kędziora
Poznan Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland
e-mail: kedziora@man.poznan.pl

Marek Lewandowski
Poznan Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland
e-mail: marekl@man.poznan.pl

Cezary Mazurek
Poznan Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland
e-mail: mazurek@man.poznan.pl

Paper type
Case study

Abstract
This paper presents organisation, challenges, and effects of Software Governance activity that was introduced in a large R&D project GÉANT that had been already in development for several years. We present the background of this activity, its structure, actions taken, as well as its achieved and expected results.

Acknowledgements
This work is supported by the GÉANT project, which is in part funded by the European Commission under the Seventh Framework Programme, contract number FP7-ICT 238875.

Keywords
Software governance, software development best practices, development infrastructure, quality assurance, distributed development.

1. Introduction
Software governance activity is of crucial importance in achieving efficient and high quality development. Its importance increases with complexity of both the organisational structure behind development process and domain where the results are applied. The multi-domain GÉANT network [1] interconnects Europe’s National Research and Educational Networks (NRENs) using an innovative combination of routed IP and switched infrastructure. The GÉANT Service Portfolio, during GN3, its third four-year phase, consists of a set of network services which are grouped into three types: Connectivity Services, Network Performance and Operations Services and End User Applications [2]. These applications and services are developed by independent and distributed groups of developers. Although results of efforts of these groups would finally complement each other, the development methodologies, management procedures, used tools and delivered quality were inconsistent [3], [4].

This paper presents organisation, challenges, and effects of Software Governance activity that was introduced in a large R&D project GÉANT that had been already in development for several years. We present the background of this activity, its structure, actions taken, as well as its achieved and expected results.

1.1 Need of unified software governance within GÉANT
The key challenge in establishing Software Governance activity was how to gracefully direct already established and diverse software developments toward the common ground and avoid silent or even explicit rejection. Therefore, the baseline for the activity task was determined through a survey of all GN3 software development activities. This determined the practices, technologies, issues, and expectations from the governance activity. Results of the survey were also expected to identify common paths and approaches already established by developers. According to Forrester’s Maturity Model [5] four stages of maturity can be defined:

- Ad hoc - there are no formal IT governance processes, procedures or mechanisms. IT investments are made on a completely ad hoc basis
- Fragmented - there has been some effort to formalize IT governance practices. These formalized processes may exist in some units and IT decisions within those units may be optimised, but there is no global effort to coordinate it within whole enterprise
- Consistent - IT governance processes have been consistently applied across the enterprise
- Best Practices - IT governance processes are fully evolved and optimized across the enterprise. A strong IT portfolio management process is in place to ensure that all IT investment decisions are themselves optimized. Executives are active participants in the governance process.

Results clearly indicated that software projects, conducted within the GÉANT consortium, and built upon similar technological platform (being mostly Java server applications) but differed significantly in terms of practices and tools. All of development teams maintained and administered their own software development infrastructure: repositories, issue trackers, wikis and project sites. Thus, development infrastructure was multiplied, which caused ineffective usage of manpower and inadequate accessibility of source codes and documentation. Although teams declared use of agile methodologies, most introduced them inconsequently or selectively. The differences were significantly noticeable in frequency of releases and change management. Moreover, while development teams were subordinated to GÉANT tasks, particular developers were subordinated to their native NRENs. Frequent changes of developers in some teams were also an issue.

According to abovementioned Forrester’s Maturity Model and based on the survey results, GÉANT’s approach to software development was classified as “fragmented”: the effort of formalization of IT procedures was done by individual units (teams) and lack of coordination was visible. These differences could lead to duplication or overlapping efforts, integration problems, and uneven quality of products. Finally, there was a clear interest in concrete guidelines related to various aspects of coding, testing, deployment and support, ranging from technological pointers, calls for conventions, to lifecycle management guidelines.

1.2 Related work
The earlier works from a number of distinct areas are relevant for the subject of this paper. They include governance within individual projects, but also topics of coordination, impact of agile on software development, or communication within distributed teams.

Within the body of related literature, most of research is limited on specific scientific and technological domains, like assessing processes, platforms, and issues in developing parallel scientific code [6], or individual endeavours within commercial organisations Error! Reference source not found.. Only the papers dealing with individual projects describe the actual measures and experiences in improving and modifying software governance, using described projects as examples for application and evaluation of applied approaches, even if proposed methods have general methodological ambitions, like combining of practices from both formal and agile methodologies [8].

Some papers discuss general IT governance in large organisations [9], which is a topic primarily related to organisational management and not at the level of software project lifecycle. However, some key elements of IT governance [10], which are: organizational structures, processes, (like planning, methodologies, maturity models), and relational mechanisms (supporting participation and collaboration between stakeholders) are perfectly mapping on software projects. Similarly, observations from [11] name four challenges in overhauling IT that are also relevant for the GÉANT attempt to improve software governance. These are: resistance to changes; requirement to establish relevant metrics; alignment of IT with business (or, in our case, goals of GÉANT), and need for IT to have business management tools helping it to deal with business and people aspects. The latest challenge is going beyond the typical technical IT viewpoint, as it is related to optimisation of operations, use of resources and impact on primary organisational or business goals.

Papers dealing with a number of software projects usually rely on historical data from open source or proprietary developments that are available to researchers in order to investigate governance practices and their effects. For example, some researchers analyse the relationship of organisational management and coordination with quality and effort in a number of open source projects [12]. Others evaluate the effects of adoption of agile development methodologies Error! Reference source not found., and their impact on coordination in several projects, A related subject is adoption of software engineering standards in project management [14].
Among related papers, none describes the experience and effects of governance harmonisation effort across a number of software projects. This is obviously a kind of effort that is rarely undertaken and publicised afterwards. In fact, [14], emphasises that enforcing of consistent standards across many projects is the least well understood problem in software management.

2. GÉANT software governance

In order to consolidate the work of developers and shift maturity of the developed software firstly into “consistent” and finally towards “Best Practices” levels, the Project Office established software governance as a new activity independent from developments and responsible directly to project officers. It was structured into three closely related tasks:

- Best Practices – collecting, publishing and promoting best practices related to software architecture, its development and quality assurance [15].
- Development Infrastructure – establishing and maintaining common infrastructure for software development [16], [16].
- Quality Assurance – verification and support, if needed, of application of best practices and common software development infrastructure by means of periodical audits [4], [18].

It is stated in Error! Reference source not found. that “the organisation structure in which the agile project occurs should be a consideration because it is an important enabler of effective coordination by supporting staff availability and proximity”. In GÉANT, there is a serious limitation in that regard, as the project management and its internal organisation is overlaid on the existing organisational structure of participating NRENs. All three activities had to adapt to that fact, both in terms of its internal operation and in approaches and services offered to software developments.

Unlike various commercial and community open software, GÉANT encompasses a number of software projects and products with different histories and developer and user communities. Most of them are within the realm of open software, including those using GÉANT’s own open outward software license. Encompassed developments share other properties common to open software, like diverse and distributed community of developers, some of whom are involved in development with low level of contribution during long periods of time. The developers usually communicate through remote communication tools, although subsets of developers work in the same location and meet regularly. Most of developers working on the same project meet once or twice per year.

More formal leadership styles are always used, in which the leadership role and decision making are formalized within the organization, but with a significant space for involvement of developer community in the decision making process. These styles are classified as “formal” and “benevolent dictator” in [12] (with “community adhering to rules” and “informal community” as remaining two). Unlike the most open source developments, GÉANT software projects have clear roadmaps, deadlines, and schedules. For these reasons, and contrary to usual open source practices, initiatives for a major refactoring usually require endorsement from within the project hierarchies. Refactoring, as a method for improvement of software quality, is also important facilitator of distributed development. This led to the need to stimulate refactoring efforts through best practices, software audits performed by the software governance activity, and even development of global Project Management Framework endorsing faster propagation and acceptance of new ideas within GÉANT.

2.1 Best practices

In order to respect fragmented efforts of development teams, software governance leaders decided to prepare the initial best practices on the basis of agile methodologies recommendations [18] and provide common sense, wide-ranging but still informative and useful guidelines, covering even so basic programming and development procedures, and to deploy the support infrastructure based on mainstream development tools. Best Practices established by GÉANT Software Governance were therefore developed in order to guide the software developments towards a common path across individual developments and methodologies. Because of the range of covered issues, Best Practices were published in three separate guides: “Software Developer Best Practice Guide”, “Software Architecture Strategy Best Practices Guide”, and “Quality Assurance Best Practices Guide”. The final edition of Best Practices covered wide range of topics, including following:

- Use of GÉANT development and QA infrastructure
- Software development process
- Software design
- Coding guidelines
- Version control management
- Internal and public documentation
- Code analysis using tools
- Code reviews
- Testing
Best practices task also lead the summer schools for software developers, contributed to other training events, and conducted in-depth audits of two software projects. It also worked together with the quality assurance task to identify developer needs to be addressed in next release of Best Practices Guides (four incremental iterations were produced) and develop guidelines for regular periodic QA audits.

2.2 Software development infrastructure

GÉANT portfolio is composed of many services, underpinned by a large number of software products and components. The project consortium involves many NRENs located in different countries, what imposes a distributed nature of software development. To ensure a basis for effective and productive software development, the software governance activity, apart from unified development process guidelines, provides also a common software development infrastructure. The common infrastructure not only simplifies migration of development teams and integration of individual projects, but also delivers well-accepted mechanisms for software projects’ management. It was agreed that software governance provided comprehensive support and maintenance of the installed tools through the Help Desk facility, so developers could fully devote their time to day-to-day duties related solely to software design and development. In particular, software governance provides and maintains the following services:

- Source code management (Subversion)
- Artifacts repository
- Issue trackers (Altassian Jira)
- Collaborative web spaces (Altassian Confluence Wiki)
- Continuous integration server with tools for automated source code analysis such as PMD and Checkstyle (Hudson/Jenkins)
- Collaborative source code reviewing and commenting tools (Altassian Fisheye and Crucible)
- Authentication and authorization (Altassian Crowd)
- Maven artifacts’ repository with software license checking capabilities (JFrog Artifactory Power Pro)

The common infrastructure delivers also GÉANT Forge and GÉANT Downloads Portal, where software products and documentations are available to the public audience. Recently the infrastructure was also extended with tools supporting project management processes, namely dotProject and web2project. All services delivered by software governance activity are hosted onto the virtualized environment, what ensures good performance and improves the scalability and flexibility of the overall infrastructure.

2.3 Quality assurance

In order to verify if (and to what extend) developers take advantage of the published best practices and established software development infrastructure, the Quality Assurance task conducts periodical (every four months), independent audits. Although auditors are independent, they provide developers with suggested conclusions and recommendations, so they can comment on these and provide their comments. Additionally, auditors collect source code and software development metrics (e.g. number of lines of code, test coverage, open versus closed issues), followed by interpretation, which provide project managers with clear visualization of the progress of development.

3. Results

The software governance activity has been supporting eleven development projects. Majority of these have been developed according to agile methodologies (Best Practices recommend OpenUP [20]), which entails frequent releases and possibility of rapid changes of requirements. One of the first visible result of the software governance activity was migration to the common software development infrastructure [16]. The utilization of this infrastructure was very important from projects managers’ point of view, since it provided not only a unified platform of tools supporting development but also simplified project management and provided managers with additional tools and functionalities. For example, it was possible to collect software development metrics that reflected both quality of the software (e.g. test coverage) and pace of the development (e.g. opened versus closed issues, SVN log entries). At the end of the 3rd year of the project, nearly all of projects have been migrated to the common infrastructure (Table 1).
<table>
<thead>
<tr>
<th>Project / Tool</th>
<th>SVN</th>
<th>JIRA</th>
<th>Hudson</th>
<th>Forge</th>
<th>Fisheye / Crucible</th>
<th>REPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoBAHN</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>cNIS</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>HADES</td>
<td>NO</td>
<td>YES</td>
<td>PARTIALLY</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>i-SHARe</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Oppd</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>PS AS</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>PS E2E MON MP</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>PS LS</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>PS RRD MA</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>PS Telnet/SSH MP</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>PS SQL MA</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>No. of tool usages</td>
<td>10</td>
<td>11</td>
<td>8.5</td>
<td>11</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Adoption of common development infrastructure by 11 projects

Because of proprietary code issues, source code of HADES project could not be moved to the common SVN repository. Anyway, the source code of this project is publically available. Since three (HADES, Oppd, PS Telnet SSH MP) out of eleven projects are written in Perl language, these could not be directly deployed on Continuous Integration server (Hudson). Because of the same reason, artifacts of these projects could not be published on the repository. Majority of perfSONAR (PS) modules are not publicly available in the repository, because of the architecture of PS (individual modules are shared via SVN repository).

Figure 1 presents exemplary charts that visualize changes of software development metrics (collected during three sequential periodical audits) for cNIS project: number of lines of source code (LOC), percentage of comments in the source code (% Comments), number of JIRA open issues (Open issues), number of JIRA closed issues (Closed issues), number of compilation warnings (Compilation warnings), number of test cases (Test cases).
Although the software development metrics are being collected for all development projects in each audit cycle, exemplary graphs for 5th, 6th and 7th audit of cNIS have been select for this paper, since these metrics reflect results of intense development conducted in a relatively short time. In addition to the charts, the audit report contained following interpretation: “When the 5th and 6th audits were conducted, new functionality related to cNIS 3.0 had been developed. This is clearly visible in the software development metrics (lines of code, open and closed issues, test cases). In parallel, source code was refactored (resulting in a significant reduction of compilation warnings). While the number of lines of code increased (from approximately 182981 to 235958), the percentage of comments decreased (from 12.98% to 9.84%). This probably means that the source code documentation (Javadoc) has not been updated. Between the 6th and the 7th audit no changes in source code were observed. During the same period, the number of closed issues significantly increased (from 731 to 786) while number of open issues decreased only slightly (from 54 to 50). This indicates that intense testing was done in that time. It effected in closing numerous (solved) issues and creating new ones.”. Each audit is followed by report containing a number of recommendations and comments for developers. Post-audit recommendations are then inserted into projects’ JIRA ticketing systems, so developers can smoothly include them in their release and development plans.

Since knowledge (best practices) and tools (software development infrastructure) transferred to developers were quite substantial, thus, in order to avoid confusion and misunderstanding among developers the strategy of incremental alignment, instead of rigid proscription, was adopted. It included gradual detailing of initial recommendations, continuous addition of more advanced tools and their integration with the ones provided earlier, and mapping of recommendations on actual use of common tools, with periodic measurement of projects’ progress in terms of alignment with best practices and provided tools. More advanced or specific topics and tools related to later software lifecycle stages have been addressed through best practices updates and extension of the infrastructure. Periodical audits by Quality Assurance task resulted in numerous recommendations that were given to developers. Best Practices also promoted some low-hanging fruits in terms of low-level procedures and recommendations that could be directly adopted among developers. The intention was not to establish another methodology, but to present specific elements from existing frameworks, addressing both software development and its maturation towards operational services.

This scheme of software governance activity within GÉANT was strongly aligned with the principles of agile methodologies that were, through best practices, promoted within the scope of individual software developments. This approach strongly relied on developers’ feedback and demands and observation of issues identified through QA activities, and monitoring of development infrastructure usage. Several meetings with software development teams were organised. During such events, some actual guidelines and other aspects of software governance were presented.

4. Proactive support

Because the initial results of the software governance activity efforts were widely applied by developers’ community, it was decided to extend the activity’s initial scope with more proactive support in the domains of:

- Extension of best practices with guidelines related to the transformation from software developments into mature and reliable products
- Facilitation of knowledge and experience sharing between developers
- In-depth audits on software quality, reliability, efficiency and security
- Support in appliance of the recommendations from periodical audits
- Support to the Project Office with tools for verification of compliance with the IPR policy

These additional areas were addressed by joined effort of all three tasks.

Gaining knowledge and improving skills is crucial and should become an integral part of project. Generally speaking training attendees gain not only knowledge related to new technologies and methodologies (“hard skills”), but are able to exchange their experience on high level, that goes beyond issues related to day-to-day work. One cannot underestimate “soft skills” that are gained in trainings. Interpersonal relationships developed during these educational events have a great impact on team spirit and cooperation between teams’ and groups’ members. The attendees can there find inspiring examples from other projects and role models, and also learn about analogous or comparable experiences. Thus, a periodical event (Summer Developers School) was established, providing developers from different GÉANT projects with the opportunity to mingle, exchange experiences, learn best programming practices, exercise agile development and solve common problems and issues while using a shared development infrastructure during hands-on sessions. The topics that were covered during these events were identified from periodical audits and requested directly from developers. For example,
it was revealed that developers tended to avoid writing tests. In order to promote Test Driven Development (TDD) [21], workshop session during Summer Developers School 2011 workshop sessions were led by authors of Mockito framework [22]. These summer schools proved to have a significant influence on development teams, which often decided to adopt some new development techniques. Additionally, two security-related trainings, led by GÉANT Security Expertise Delivery (SED) and with participation of Best Practices task, were delivered. Since majority of GÉANT applications are Java server applications, trainings focused on web security and ethical hacks techniques that could be utilised by developers and testers in their daily duties.

Because not all of QA recommendations were implemented the first line of support for each development team was established by introducing Quality Assurance Personnel. The QA person, taking advantage of her / his knowledge and experience, became an advisor that constantly supported the project team, observing the development process and investigating potential issues, even before these actually arise. Such situations are further reported to lead developers and development teams. This issue is then solved internally, by development team and QA person. If the issue cannot be solved internally, QA person is responsible for setting up external communication and knowledge exchange with other development teams and available experts. The QA person was not supposed to became a member of the development team, so this person should not be expected, for example, to write production code, test cases, or project’s technical documentation. Significant involvement in such activities could diminish this person’s capability to track and scrutinize the development.

The software governance activity also coordinated several in-depth audits: on projects’ architecture, code, reliability, and efficiency for two applications under development; and security for an application in migration from pilot to operational. During these audits, the development team leaders and QA persons served as liaisons between the auditors and developers. The in-depth audits significantly differed from the periodical ones. First of all, their scope and structure were tailored to particular needs and expectations according to already noticed issues. These audits were conducted by teams (4-10 persons) of experienced developers and software engineering specialists. The range of topics covered during these audits varied from efficiency of algorithms implementation and data structures usage, through efficiency of external communication (other applications, databases) and usability of graphical user interface to effectiveness of project management mechanisms. The source code was explored both manually (code inspections) and automatically (with e.g. Checkstyle, PMD). Receivers of the audit results agreed, that achieving such detailed and sophisticated analyses and advices would not be possible by using only internal reviewing and testing procedures.

5. Discussion
Based on the experience with adoption of software development related best practices observed through regular and in-depth audits, it is evident that their promulgation must be stimulated by performing a number of promotional, educational, policy-related, and other supportive measures. For example, it is not possible to promote the most basic best practices that are already familiar to software developers as parts of common industry standards, like coding conventions, through direct educational activities during trainings. Although the occasional lack of adherence to such norms was noticed through the QA audits, the proper way to enforce them was through automation of code checking. Backing such topics by lectures would be perceived by developers as “brain washing”. Instead, lectures should focus on “hot” and attractive topics or subjects exercised during accompanying hands-on sessions and competitions.

This also emphasises importance of using software process tools supporting tools that foster adoption of some approach or methodology. For example, the availability JIRA GreenHopper agile process management and planning tool, was a great stimulus for better adhering to agile practices. However, even such a help does not guarantee adoption of the best practices, as their actual implementation must be preceded by a clear decision by the management of a software project.

5.1 Establishing common staging framework
After unifying the software development infrastructure across all distributed teams through installation and maintenance of a set of commonly used services the emphasis is also put on installation and maintenance of QA testbed environments for software development teams as well as establishing procedures and software tools supporting IPR coordination within software development lifecycle.

QA personnel conducted the survey that was supposed to verify if (and to what extent) developers applied common approach to testing and releasing the software. The general motivation for that was to define a staging framework throughout all developments. So far the staging framework has been partially addressed by development teams, which adapted software release management procedures described in Best Practice Guides.

To cover the missing part related to the usage of software testing environments, software governance activity has implemented the QA testbed infrastructure. It is based on a virtualized environment, where a number of virtual machines is dedicated to development teams for individual software testing and demonstration, whereas the remaining virtual machines are the poll of resources shared among teams on the calendar basis, mainly for
distributed and complex testing scenarios.

The fact that QA testbed is based on virtualized environment enables easy extension of the infrastructure towards more complex one, where a number of virtual machines can be increased thanks to adding new hardware resources and thus virtual nodes. In the near future, the QA Testbed is also considered to be extended by the integration with a testbed of virtual routers, in order to satisfy the requirements of distributed end-to-end testing scenarios for GÉANT bandwidth-on-demand software.

Thanks to its scalability the QA testbed infrastructure can be further developed to make test, deployment and release environment complete. To address GN3 Best Practice Guides, it could follow the structure proposed in DevOps methodology. While having the automated deployment of software artifacts to testing environments in place, such series of environments could be organized in the following way:

- Development (DEV): build and some basic functional and unit tests are done in DEV environment. There may be several instances of this environment, as it is heavily used by developers.
- System Integration Testing (SIT): integration tests and some additional functional tests are performed in a SIT environment used by Continuous Integration tools. It is also used by developers, but its usage is triggered by commits or nightly builds.
- User Acceptance Testing (UAT): User Acceptance Testing is done in a UAT environment. Several changes of a Release Candidate (RC) may be combined and made available to the testers and internal users in order to verify that the changes function as intended. Alternatively, this testing may be performed on individual feature branches.
- Staging: in highly critical missions, this testing may be followed by deployment to an optional Staging environment that almost fully emulates the production one. All changes that are to be included into a planned release are packaged into deployed here. A variation of this approach is deployment of a release candidate on the not-critical production instance serving selected representative users willing to tolerate possible usability problems.
- PROD (production): the final deployment of operational product is done into PROD environment.
- Stress and Volume Testing (SVT): there may also be an asynchronously operated SVT environment, which is fully separated from the common DEV – SIT – UAT – Staging – PROD path, but may be quite close to Staging or PROD environment.

Once the test, deployment and release environments are in place along with tools and mechanisms for automated deployment of software artifacts, such an approach enables continuous delivery and allows for extremely short release cycles. It also ensures that performed steps can be reproduced in a reasonable timeframe.

5.2 Towards coherent IPR policy

Since GÉANT services have been developed by distributed development teams hired by numerous NRENs, establishing the IPR policy and managing of its appliance in development has been crucial. The software governance was requested by the Project Office to support this process by defining procedures and deploying software tools that would automatically verify the compliance of the software with GN3 IPR Policy. To meet such requirements, Software Development Infrastructure task installed and configured software tool that, though interaction with a continuous integration system, extracts and manages licenses used by every build of developed software and its dependencies. It also helps to create and maintain a repository of allowed and forbidden licenses. The license control introduced in such a way strongly supports software release process and leverages the overall confidence of proper verification of project IPR policy.

6. Conclusions

This paper analyses the importance and impact of the Software Governance activity in distributed software development in R&D project. It also summarises an approach that goes beyond the usual methodologies and frameworks, as it was tailored for the characteristic of developments within GÉANT.

In the first year of GN3, the software governance activity focused on defining best practice guidelines as well as identifying and installing common software infrastructure. In parallel, quality assurance team initiated periodical audits that are supposed to verify if the best practices are applied and the common infrastructure is properly utilised. When these tasks were finally put into operation, the urgent need of proactive support for developers was spotted. Thus, it was decided to extend the scope of interest of the activity and provide trainings addressing the identified issues and direct, less formal support for developers. As a result, in the second year of the project a number of actions were taken (establishing Summer Developers School, defining and introducing of the QA personnel, starting a series of software in-depth audits). Taking into account the need of continuous improvement of the quality and efficiency of the development, in the third year of GN3, the software governance activity faced additional challenges. Among others the software governance team in purchased and installed the QA testbed infrastructure. Finally, when aforementioned actions were introduced, the software governance focused on refining the common staging framework. This area needs further efforts to be taken. In parallel, the GN3 Project
Office intensified its efforts on definition of the consistent IPR policy. Although a lot has been done on this issue, there is still the emerging need of clear, common and unified approach to the IPR. The success of software governance activity depends not only on the quality of documents that this activity produces. The success arises from cooperation and synchronized actions in many areas; in case of GÉANT these areas were: best practice guides and workshop initiatives, software quality assurance, common software development and test environments. Additionally, one must not underestimate the impact of personalities of auditors, QA personnel and trainers. When QA staff does not wear policemen hats but acts like guardian angels, the chances of success of the software governance activity significantly increase. Additionally, direct and immediate support of project managers and willingness of developers are key factors of the success of the whole initiative.

As a result of the software governance activity, the GÉANT software development has been shifted into “consistent” stage (according to Forrester’s Maturity Model). When all best practice recommendations are followed, when periodical audits are completed with “all correct” statuses and when all development teams fully utilize software development infrastructure, the GÉANT approach will be classified as “best practices” level. The approach taken by the GÉANT Software Governance consists of not only procedures (best practices), formal processes (audits), and support resources and activities (maintenance of development infrastructure), but also addresses more general and thus more difficult underlying areas, like identifying needs, motivating developers to applying provided guidelines, trainings provisioning, and stimulating knowledge and experience sharing. Although the described activity was defined for specific needs of the large R&D project that develops tools and services operating in the pan-European network, it can be generalized and applied in other large scale distributed projects, even in domains beyond software and service development.

References

[16] Żurowski W. et al., “Report on Development Infrastructure Usage and Adoption,” GN3 deliverable DS4.3.2
M., Mantadaki V., Mazurek C., Stamos K., Szubarga J., Zaoudis G., “QA Audit Report 2,” GN3 deliverable DS4.2.1, 2

Vitae

Branko Marović received his BS, MS and PhD degrees in computer engineering and computer science from University of Belgrade in 1993, 1998, and 2001. He is with the Belgrade University Computer Centre since 1994, currently leading its Centre for Software Development. His experience includes medical informatics, computing infrastructures, and management of software projects. He participated in several FP6/FP7 projects, often as activity or task leader. Branko is also teaching MS and PhD courses on medical informatics at the schools of electrical engineering of University of Belgrade and University of Banja Luka.

Paweł Kędziora received his M.Sc. degree in Computing Science from Poznan University of Technology in 2007, specializing in a design and development of software systems. In 2007/2008 he worked at Hewlett-Packard as a software developer in a warehousing domain. In 2008 he joined Poznan Supercomputing and Networking Center where he works till present, initially as a software engineer and now as a system analyst. He participated in FP7 QualiPSo dealing with systems interoperability and software quality. Currently he is involved in FP7 Géant3 project in a domain of software governance, as well as in homeland projects.

Marek Lewandowski received the M.Sc. degree in Computing Science from Poznan University of Technology in 2007. In the same year he joined Poznan Supercomputing and Networking Center. He is interested in database systems, data mining, software development and quality assurance of the software and development processes. He has been involved in Géant2 and Géant3 projects as a developer and a member (later on leader) of the Quality Assurance team.

Cezary Mazurek is the Head of the Network Services Department at PSNC. He received his PhD from Poznan University of Technology (2004). His expertise and experience include a variety of advanced network services including digital libraries, interactive television, telemedicine, data and information management and access to grid services. His activities are related to Future Internet technologies and experimental research infrastructures. He is a leader of Software Governance Activity and project management team member in FP7 Géant3 project and Quality Coordinator in WF4Ever project. He is the author or co-author of over 100 papers in professional journals and conference proceedings.