Scaling Agile Methods to Regulated Environments: An Industry Case Study

Brian Fitzgerald*, Klaas-Jan Stol*, Ryan O'Sullivan†, and Donal O'Brien†
*Lero—The Irish Software Engineering Research Centre, University of Limerick, Ireland
†QUMAS, Cleve Business Park, Monahan Road, Cork, Ireland
bf@ul.ie, klaas-jan.stol@lero.ie, rosullivan@qumas.com, dobrien@qumas.com

Abstract*—Agile development methods are growing in popularity with a recent survey reporting that more than 80% of organizations now following an agile approach. Agile methods were seen initially as best suited to small, co-located teams developing non-critical systems. The first two constraining characteristics (small and co-located teams) have been addressed as research has emerged describing successful agile adoption involving large teams and distributed contexts. However, the applicability of agile methods for developing safety-critical systems in regulated environments has not yet been demonstrated unequivocally, and very little rigorous research exists in this area. Some of the essential characteristics of agile approaches appear to be incompatible with the constraints imposed by regulated environments. In this study we identify these tension points and illustrate through a detailed case study how an agile approach was implemented successfully in a regulated environment. Among the interesting concepts to emerge from the research are the notions of *continuous compliance* and *living traceability.

***Index Terms*—Agile methods, regulated environments, Scrum, case study.**

I. Introduction

The widespread penetration of agile methods is readily evidenced in a large-scale industry survey which reported that 80% of respondent organizations were following an agile approach [1]. Agile methods were initially viewed as best suited to (a) small projects with (b) co-located teams and (c) non-critical projects [2][3]. The first two of these constraints (small projects and co-located teams) have been addressed: several research studies have been published of agile adoption by large teams (e.g., [4][5][6]) and in distributed environments (e.g., [7][8][9]). However, the final constraining characteristic, that of agile adoption in *regulated environments*, has yet to be addressed. In this area, there is very little rigorous evidence of successful application of agile approaches—typically short experience reports in workshops or practitioner reports (see [10] for a review). A lack of such evidence inhibits the adoption of agile methods in regulated environments. A number of key agile advocates have argued that agile software development methods are best suited to non-critical systems. For instance, Boehm [11] cited Scott Ambler [12] (originator of agile modeling) in stating that,"*I would be leery of applying agile modeling to life-critical systems.*"

Regulated environments, such as automotive, aviation, financial services, food, medical devices, nuclear, pharmaceutical and railway, pose particular challenges for software development as software has not been traditionally viewed as core in these sectors. Recent changes in the medical device sector, for instance, illustrate clearly the need for scaling of software to regulated environments, and the challenges which this presents. Traditionally, medical devices comprised primarily hardware with perhaps some embedded software. Since 2010, an amendment to the EU Medical Device Directive now classifies stand-alone software applications as active medical devices [13]. This has major implications in that software which was traditionally seen as secondary and a means to an end in the sector, has now moved center-stage.

Agile methods and regulated environments are often seen as fundamentally incompatible [14]. The reason for this can be traced to the Agile Manifesto [15] which identifies four fundamental value propositions for agile as:

1. Individuals and interactions **over** Processes and tools.
2. Working software **over** Comprehensive documentation.
3. Customer collaboration **over** Contract negotiation.
4. Responding to change **over** Following a plan.

While the agile advocates acknowledged the statements on the right as having value, they valued the statements on the left more. However, in regulated environments the statements on the right represent values which are of particular importance. Thus, an initial assessment might conclude that agile approaches and regulated environments are incommensurable.

Agile software development methods are faced with some fundamental challenges in regulated environments. Agile processes follow an empirical logic in a plan-do-check-act (PDCA) cycle [3], whereby some development is planned and done, the results are inspected, and adaptations are made to improve the process to solve any problems that have arisen. However, in regulated environments, a *defined* logic rather than *empirical* logic is more desirable. Development processes in regulated environments are typically audited by external assessors. Thus, the granularity at which development processes are expressed and adapted requires careful tailoring in a regulated environment. Furthermore, regulated environments require rigorous traceability. In the case of requirements, for example, these need to be traced from initial requirement through to final implementation in the code-base.

This study presents an in-depth account of agile method implementation in a regulated environment at QUMAS, a leading supplier of regulatory compliance management so-

978-1-4673-3076-3/13/$31.00 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA
Software Engineering in Practice

lutions for document and quality management, submissions management, and regulatory approval in the life sciences sector. The paper is laid out as follows. In the next section, the essential principles of the agile approach are presented, specifically as they relate to Scrum, by far the most commonly used agile method [1], and the basis of the agile method in use in QUMAS. Following this, we discuss the specific challenges that regulated environments pose for software development. We then discuss the case study research method adopted for the study, and provide details on QUMAS, the case company. Details of the implementation of the agile method and how it was configured to address the constraints posed by operating in a regulated environment are then presented. Finally, the implications of the study are discussed.

II. Agile Methods

The formation of the Agile Alliance in 2001 and the publication of the Agile Manifesto formally introduced agility to the field of software development. Those involved sought to "restore credibility to the word method" within the context of software development [16]. The manifesto conveyed an industry-led vision for a profound shift in the conventional software development paradigm.

Many different methods have been labeled as agile, such as eXtreme Programming (XP) [17], Scrum [18], Crystal [19], Dynamic Systems Development Method (DSDM) [20], Agile Modeling [21], Feature Driven Design [22], Lean software development [23], and perhaps even the Rational Unified Process (RUP) [24]. Notwithstanding the breadth of agile methods available, they are underpinned by the 12 fundamental principles of the Agile Manifesto [15].

One of the most popular and widely adopted agile methods is Scrum, which is also the subject of this paper. The Scrum metaphor has its origins in Takeuchi and Nonaka [25] who used it to describe an innovative approach to new product development, borrowing the rugby practice of a group of players combining to move the ball forward. The method was developed jointly by Sutherland and Schwaber [26] and has evolved to incorporate additional practices over the years [27].

The Scrum framework is reproduced in Fig. 1. As can be seen in the figure, the lifecycle of a Scrum project is largely comprised of iterations of development "sprints" with an initial planning phase and a final closure phase of sprint review and retrospective. Planning enables both architectural and scope concerns to be addressed and the closure phase incorporates release management. The initial and final phases are suggested to be predictable and may be defined.

The empirical nature of Scrum is evident in the on-going iterations or sprints that adapt to feedback and change throughout the project. A notable aspect of this approach is the inclusion of groups that were considered to be obstacles to traditional development projects (e.g. sales and marketing). The method embraces change by enabling the development team to both react and promote changes as the system evolves [29].

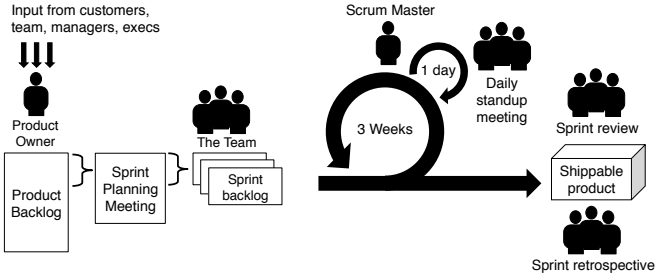

Fig. 1. Scrum Framework (adapted from Deemer and Benefield [28]).

III. Regulated Environments

Regulated domains exhibit varying levels of criticality, from safety-critical to security-critical [30]. A core characteristic of regulated environments is the necessity to comply with formal standards, regulations, directives and guidance. There is a plethora of regulations and standards which apply across different regulated domains. These are issued by a number of bodies or associations (e.g., ISO, FDA, PDA, GAMP, IEC, ISPE, RTCA). Also, some are region-specific (e.g. US or EU). (The specific regulations and standards that apply in the QUMAS case are summarised in Fig. 3 below).

Software plays an increasingly important role in regulated environments. The principles of the agile manifesto were identified earlier, and although an overarching set of principles for regulated environments does not exist, a number of core issues for software development in regulated environments may be inferred. These issues include quality assurance, safety and security, effectiveness, traceability, and verification and validation. They are summarized in Fig. 2, and are elaborated on below where a discussion of how they may conflict with agile methods is presented.

A. Quality Assurance

The IEEE [31] define Quality Assurance as: "*a planned and systematic pattern of all actions necessary to provide adequate*

Quality Assurance:
- Systematic and inherent quality management underpinning a controlled professional process
- Reliability and correctness of product

Safety and Security:
- Formal planning and risk management to mitigate safety risks for users
- Securely protect users from unintentional and malicious misuse

Effectiveness:
- Satisfying user needs, and delivering high value to users with high usability

Traceability:
- Documentation providing auditable evidence of regulatory compliance and facilitating traceability and investigation of problems

Verification and Validation:
- Embedded throughout the software development process (user requirements specification, functional specification, design specification, code review, unit tests, integration tests, requirements tests)

Fig. 2. Key concepts in regulated environments.

confidence that the item or product conforms to established technical requirements."

Pfleeger et al. performed an analysis of software standards and found that they are heavily focused on processes (rather than products) [32]. They characterized software standards as prescribing "*the recipe, the utensils, and the cooking techniques, and then assume that the pudding will taste good.*" This corresponds with Deming [33] who argued that, "*The quality of a product is directly related to the quality of the process used to create it.*"

Agile principles are inspired by concepts from complex adaptive systems (CAS), such as edge of chaos and emergence [34]. Software development is a complex activity. Thus, a defined or theoretical approach is not feasible since one cannot define in advance the necessary steps to fully accomplish the development task from initial requirements through to eventual implementation. This is the key failing of the phased development approach inherent in the traditional Waterfall lifecycle [35]. In such situations, a more experimental or empirical logic is appropriate. This is the PDCA logic whereby some activity is performed, the results are inspected, and the process is adapted to resolve any problems that have arisen. The CAS concept of 'edge of chaos' is drawn on to inspire the notion of self-managing teams that can experiment and adapt, while still retaining enough structure so as not to fall into disarray. Likewise, emergence of new scenarios is embraced as an opportunity to learn from adaptation. However, these agile principles are fundamentally at odds with the desire for a defined and repeatable process that regulated environments stipulate. While agile methods stress early involvement of test groups and rapid feedback, they pay little attention typically to the relationship between development and quality assurance groups [3].

B. Safety and Security

Regulated environments place a strong emphasis on safety and security. Though safety and security focus on different aspects, they are both related to risk management. One of the most-cited software failures is the Therac-25 radiation machine resulting in a number of fatal treatments [36]. Though these accidents cannot be attributed solely to software, it is clear that it did play an important role in those cases.

Safety and security are system-level characteristics, and as such must be built-in from the start and not considered after the fact. As Mead wrote, a "*focus on features tends to result in buggy, insecure software*" [37]. Yet, agile methods suggest an iterative approach, which is often mentioned as a problem with respect to architectural quality attributes [38], of which safety and security are two examples. Tribble argued that showing process compliance is not sufficient for building safety-critical systems but that demonstrating achievement of a product's safety requirements is also required [39].

C. Effectiveness

An important factor in regulated environments is effectiveness as it relates to development speed and cost; the additional cost of strictly following predefined processes may be at odds with agile methods, which are claimed to be lightweight and flexible. The primary emphasis in software development in regulated domains is to obtain regulatory approval rather than to improve software processes per se. Thus, software development in regulated environments differs from software development in non-critical domains in several significant ways. Characteristics such as safety, effectiveness and traceability are deemed more important than time-to-market and profitability. This leads to longer product lifecycles, often up to ten years, a timespan unheard of in an agile context. In fact, one of the promises of agile methods is an increased time-to-market. DeMarco [40] argued that "*agile methods provide a tradeoff between speed and risk.*"

D. Traceability

A key concern in regulated environments is traceability, which helps to establish compliance to standards and regulations. The IEEE define traceability as: "*The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match*" [31].

Agile projects typically have few traceable artifacts [41]. Also, in regulated environments the emphasis is on the software development processes rather than the software product. Other agile principles, such as prioritizing working software over documentation, inhibit traceability as documentation is the primary evidence of traceability, and needs to be explicitly addressed in regulated environments.

Cleland-Huang explored some of the issues, challenges and goals of traceability in agile projects, and proposed a number of specific solutions to traceability in such projects [41]. As Cleland-Huang argued in the case of safety-critical projects, an organization may be fined, or its products forcibly recalled, when the compulsory traceability and other requirements are not respected.

E. Verification and Validation

Verification and Validation (V&V) are two distinctly different concepts, but often conflated. V&V is defined as: "*The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements*" [31]. Whereas verification helps to answer the question *Are we building the product right?*, validation refers to the question, *Are we building the right product?* [42].

Another agile principle, that of frequent delivery of working software, can also cause problems in a regulated environment, where this would imply more frequent formal review and approval cycles for this software. The latter represents a major undertaking for several functions throughout the organization such as the Quality Assurance (QA) function, for example, and

as such would require significant organizational change, which is never easy to accomplish. As Rakitin [42, p. 47] points out, there is a cost associated with performing verification and validation tasks in software development.

IV. Previous Work

As pointed out above, whereas the adoption of agile in *distributed settings* and the use of agile methods in *large teams* has received considerable attention, very little research has addressed agile in *regulated environments*. Cawley et al. [10] conducted a systematic review of agile in regulated environments, and identified 21 relevant works, of which 14 were peer-reviewed papers. Of the 14, only four were empirical studies (as opposed to experience reports and expert opinion), two of which focused on embedded software rather than regulated environments. The remaining two papers present a method for process assessment in the automotive domain [43], and an examination of the applicability of agile practices in the aerospace domain [44]. This clearly indicates a lack of attention for this topic while aspects such as traceability are of increasing importance. This is further evidenced by a recently published book on software and systems traceability [45].

Gary et al. reported on a safety-critical open source project that is using agile methods [46]. Our paper differs from that work in two aspects. Firstly, Gary et al. reported on their own experiences, whereas our paper presents an in-depth *case study* approach. Secondly, while Gary et al. reported on agile methods in an *open source* setting, this paper reports on agile methods in an *industry* setting. As such, our findings are directly relevant to other organizations that wish to adopt agile methods in regulated environments.

V. Research Approach

A. Background to the Case: QUMAS

QUMAS is headquartered in Cork, Ireland with offices at five locations in Ireland, UK, Asia and the US, and its customer-base of more than 250,000 users is based in 29 countries worldwide. QUMAS delivers a compliance model that standardizes and integrates the common elements of compliance tasks across the organization. This allows convergence of all compliance programs onto a single platform, radically reducing the cost of compliance and creating competitive advantage. Founded in 1994, the company has a long and proven track record in the regulated life sciences industry, and is required to comply with a number of regulations, listed in Fig. 3. QUMAS had employed a classic Waterfall approach since the company was founded. However, this approach resulted in a long time-to-market and a large release overhead, which were seen as drawbacks in the quickly changing market that QUMAS is operating in. As a consequence, they have adopted and augmented the Scrum methodology over a period of approximately two years.

B. Research Method

The objective of this research was *to investigate how an agile development approach can meet the rigorous standards required in regulated environments*. To that end, we conducted a case study. Case studies are appropriate to answer "how" or "why" questions, and to study a contemporary phenomenon within its real-life context, in particular when the boundaries between the topic of study and its context are not clearly evident [47]. Given the fact that very few studies of agile methods in regulated environments exist, the case study can act as a useful "revelatory" case [47].

A commonly cited limitation of the case study approach is its lack of generalizability, as findings of a case study are typically specific to the case study. However, this is due to a misconception of the term "generalization," since the aim of a case study is not to seek *statistical* generalization, but rather to seek *theoretical* generalization [47][48], or even to *construct* a theory [49]. Therefore, the concept of validity, and external validity in particular in the case of generalization of case study results, is dependent on the type of research. Furthermore, the "thick descriptions" [47] provided by the case study were considered much more valuable than generalizability of results.

C. Data Collection and Analysis

Informed by the established guidelines for doing case study research [47], we developed a case study protocol following the template and recommendations offered by Brereton et al. [50]. We focused our data collection on identifying *how* the organization implemented the Scrum methodology, and to identify what changes were made to the standard Scrum framework shown in Fig. 1. Data were collected during six workshop sessions located at the organization, each lasting two to three hours and involving two to four participants. The workshop meetings were held over a time period of 15 months. Such a prolonged involvement is a recommended strategy to establish validity [51]. We collected data from various sources in order to achieve triangulation across data sources, which helps to establish reliability of the findings [51]. Sources of data were semi-structured interviews with key members of staff (including the CEO, Vice President (VP) Development & Support, VP of Quality & Customer

FDA (Food and Drugs Administration)

- FDA - 21 CFR Part 820 (Quality system regulation)
- FDA - 21 CFR Part 11 Electronic Records; Electronic Signatures

ISO (International Organization for Standardization)/IEC (International Electro-technical Commission)

- ISO 9000 Quality Management
- ISO 9001:2008 Quality Management Systems
- ISO 13485 (Quality Management System (QMS) for the design and manufacture of medical devices)
- ISO/IEC 15504 (Process assessment models for systems and software)
- ISO/IEC 12207 (Common framework for software life cycle processes)
- EudraLex Volume 4 (GMP) – Annex 11 "Computerized Systems"

ISPE International Society for Pharmaceutical Engineering)

- GAMP5 (Good automated manufacturing practice (GAMP) guide for validation of automated systems in pharmaceutical manufacture)
- GXP

Fig. 3. Regulations and Standards applicable to QUMAS.

Relations Management (CRM), Development Project Manager, Scrum Master, team leads and developers). The interviews and demonstrations were digitally recorded with the participants' consent. We also had full access to all documentation relating to software development and had access to the tools used to support the software development process in the environment.

We analyzed the data using qualitative techniques described by Seaman [52]. An audit trail was established by transcribing the interviews and written notes through memoing. This in turn helped in independent analyses and cross-comparing findings, facilitating triangulation across researchers as well as peer debriefing, which are also recommended practices to increase a study's validity [51]. After analyzing the data, we sent several draft versions of our report to key informants at QUMAS. This is a form of member checking, and is a recommended practice for qualitative studies [51].

VI. The Agile Development Approach at QUMAS

The QUMAS Scrum software development life cycle procedure is formally documented. All developers are required to read and sign an "understood" declaration. The product development process at QUMAS is directed by the Product Council which consists of the relevant senior management from development and support, quality, sales and marketing (see Table I). The purpose of the Product Council is to set overall objectives, approve key phases and make strategic decisions. They identify the personnel and resources required for the project management plan and timeline. The Council meets four times per year as standard. However, meetings can be called at any time to address major items as they arise.

Once product development is sanctioned by the Product Council, a product development team must be appointed. Each team member's name must be assigned when the team is formed. This is essential in order to identify the personnel resources required for the project management plan. The core team members are the Product Owner, the Scrum Master and the lead developer. The Scrum Master is responsible for progress and prioritization of work items on a day-to-day basis. The product development team will meet regularly to review implementation progress.

The agile development approach at QUMAS is supported by a number of products from the Atlassian (www.atlassian.com) toolset as described in Table II.

VII. Regulatory Compliance at QUMAS

The augmented Scrum implementation for regulated environments (which we label *R-Scrum*) as enacted by QUMAS is presented in Fig. 4 below. The grey-shaded features are those enhancements added to the generic Scrum method (depicted in Fig. 1 above) to meet the compliance requirements of a regulated environment. Below we discuss these enhancements as they arise according the regulatory compliance factors outlined in Fig. 2 above.

TABLE I
QUMAS Product Development Team Roles

Title	Role
Product Sponsor	Executive sponsor of product development. Make key business decisions. Report to board.
Scrum Master	Overall project management responsibility. Produce the project management plan & sprint plans. Liaise between the VP of Development & Support, VP of Quality & CRM and project team. Ensure that the project time, quality, and functionality criteria are met.
Product Owner	Represents the customer. Extensive knowledge of regulations and the business domain and expert on usage of product. Works closely together with Scrum Master to define and prioritize backlog.
VP of Quality (QA) & CRM	Verify that the output(s) from each sprint adhere to the required procedures and standards.
VP of Development & Support	Ensure that acceptable progress is gained. Act as advisor to the Scrum Master and product development team members & update Executive Management including the CEO. Overall responsibility for the development team and proxy for Product Sponsor if required.
Software Developers	Coding and debugging of software. Produce required installation and associated user documentation where required.
Quality Control	Produce system test documentation and execute system test scripts in line with required standards and product specification. Document all test results for release review.

A. Quality Assurance

QUMAS have a very strong internal quality management system and quality culture. The development workflow is formally defined in JIRA (see Table II). All development sprints are audited by QA, who are independent of the development function, to ensure compliance with the defined procedure. These audits are completed within three days of

TABLE II
Atlassian Toolset in use in QUMAS

Tool	Description
JIRA	Issue and bug tracking, project management, workflow engine
Fisheye	Source code search. It integrates JIRA with the source code repository and provides additional information on source code changes for easy interpretation. It links source code changes to JIRA issues and also supports the use of the Crucible tool.
Confluence	Enterprise wiki used by project teams to share design information
Greenhopper	Agile planning and project management
Bamboo	Continuous integration (CI) server for source code under development
Crucible	Peer code review implemented as an addition to Fisheye making it easy to review code changes, add comments and record outcomes efficiently

the end of each sprint. This mode of "continuous compliance" means that QUMAS could "*theoretically release after every sprint,*" according to the VP Development and Support, a point we will return to again in the Effectiveness section (Section VII-C) below. Under the previous, waterfall-based development process, while each output produced was subject to QA review and approval, audits to approve releases were far less frequent, no more than once per year typically.

QUMAS project estimation is based on hours per task. At the end of each day, developers record the hours spent completing their tasks for the day. Before the daily Scrum meeting each morning, the Scrum Master can assess how development is progressing and ascertain whether there are any potential delays or overruns that need to be addressed. At the end of each sprint (typically on a Friday), the Sprint Review is a half-day meeting which identifies tasks not completed or tasks that have newly arisen and these are fed back to the Product Backlog for consideration in future sprints. The Sprint Retrospective meeting is combined with the Sprint Planning meeting (typically on a Monday) at the start of the sprint and the focus is primarily on improving estimations, using the data from completed tasks in the sprint.

Development is also guided by templates which guide developers through the process. For example, a design template is automatically presented to developers on initiation of design tasks. This template identifies any related stories, a list of business rules that must be adhered to, any user interface issues and an explanation of fields within the user interface, user actions, access control, and error and exception handling. Developers are trained on the use of documents and templates as part of the induction process for new employees.

Peer code review is also practiced and formally monitored in what is termed the "dev check" process. This ensures that the up-to-date design page is in Confluence, that code is checked in, coding standards are adhered to, and unit tests are run. Dev checks are performed for each task. Code refactoring is also systematically practiced. This is generally incorporated through refactoring stories.

As already discussed above, the implications of more frequent production of software for approval and review processes can be significant. Sprint cycles at QUMAS typically follow three-week intervals. QA attend the sprint reviews and retrospectives and formally approve every sprint cycle within two to three days of the end of the sprint. This requires the integration of all the requisite information to provide evidence of regulatory compliance subsequently. QA audits typically last a half-day and identify issues of non-conformance, or lack of traceability, or tasks not fully closed in line with predefined procedures, guidelines and sprint plans. Any issues are formally identified in a non-conformance report which includes a root cause analysis of non-conformance. This is fed back to the Product Backlog for resolution in a subsequent sprint. According to the VP Quality and CRM, the final QA release process is much more efficient than when following a waterfall process: "*QA audits are done at the end of each sprint which allows for improved visibility, traceability and measurement so we have no unexpected exceptions to address at final release. We are just confirming the final release.*"

This mode of '*continuous compliance*' is greatly facilitated by the traceability afforded by the toolset—an issue considered in Subsection VII-D below.

B. Safety and Security

Risk mitigation is facilitated greatly by the transparency of being able to ascertain project status at a glance and in real-time, the *continuous compliance* phenomenon discussed earlier. QUMAS also operate a four-stage prioritization scheme for tasks and bugs, ranging from priority P1 (critical) to priority P4 (cosmetic). This allows for better prioritization of key risk factors. In terms of product security, for example, the FDA require relevant regulated industry sectors to adhere and comply with the 21 CFR Part 11 regulation. In line with this regulation QUMAS software products automatically and

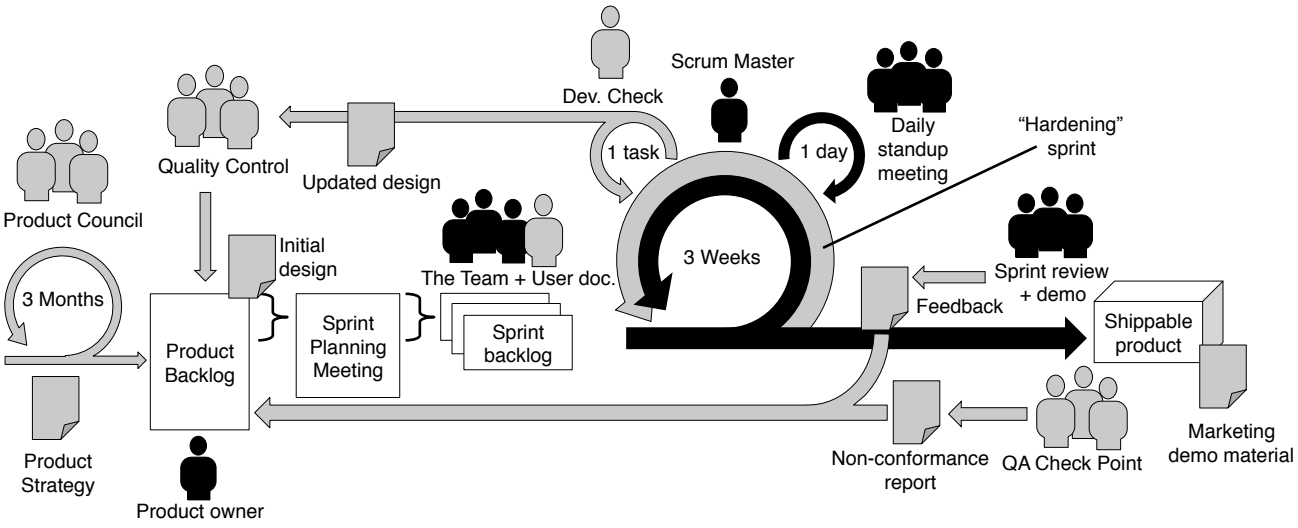

Fig. 4. R-Scrum: Regulated Scrum implementation at QUMAS.

securely binds the authenticated user's electronic signature and provides automatic required protection in the form of password expiration and unsuccessful logins. Full user audit trail capability is also provided in the product. In terms of process security, QUMAS have full audit trail visibility at all stages of the agile process, and only employees with the required security credentials can participate in the agile process.

Risk is managed through the project and is the responsibility of the Scrum master. Typically, stories of equal value are prioritized by risk, using the classification mentioned above. If difficulties arise, then the team have more time to either mitigate or avoid the risk. For example, if developing for a novel platform, early consideration of technically difficult issues allows a potentially greater number of sprints to resolve issues that arise if necessary. QUMAS also provide support to customers who adopt a risk-based approach to validation in line with regulatory guidelines, by allowing the customer to leverage the functional testing performed by QUMAS during the agile process. Customer access to this test and associated process information is managed in a controlled manner.

C. Effectiveness

Effectiveness is about satisfying user needs and delivering high value with high usability to customers. Having the Product Council direct development ensures alignment with business strategy is formally considered at least once per quarter. Also, a documentation person is a member of the Sprint team ensuring a link between development and documentation and support.

Agile methods such as XP recommend an onsite customer; that is, co-location of developers and customers with a view to directly validating and prioritizing requirements. While QUMAS do not have an onsite customer typically, the surrogate for this role is the Product Owner. The Product Owner and Scrum Master are deeply involved in sprint planning and sprint review meetings, thus affording an opportunity at three-weekly intervals for detailed feedback on desirable functionality and how it should be prioritized from the customer perspective. Under the previous waterfall process, sales and marketing were consulted about requirements at the beginning of the project, and the resulting requirements specifications were rigidly adhered to during subsequent development phases.

The frequent delivery of working software inherent to the agile development process has also had major benefits for QUMAS. Because the software can exhibit functionality which has been prioritized, this can be demonstrated to customers early. For a newly developed product, several customers purchased the new software in advance of its formal release on the basis of the interim working functionality that could be demonstrated. This would not have been possible under the previous waterfall development process according to the VP Development and Support. However, in the spirit of satisfying customer requirements, QUMAS have committed to being reactive to the specific needs of these customers. Given the cadence of three-week sprints, QUMAS believe that customer requests could be implemented and delivered in about five to six weeks if necessary under the agile development process.

The agile development process also links validated builds of the software product with the relevant demonstration package test data. Pre-sales personnel can identify features they wish to demonstrate, select the appropriate validated build containing those features and the relevant demonstration package test data to show the new software to potential customers, and be confident that the demonstration will progress smoothly. This is a major benefit over the previous process. Previously, pre-sales personnel had to manually prepare demonstration material, which was a very time-consuming process. Furthermore, because of the inevitable likelihood of a greater prevalence of bugs in newer releases of software, pre-sales personnel tended to choose more stable software releases, perhaps more than six months old, when demonstrating to customers. As a result, newer functionality tended not to be included in those demonstrations.

As QUMAS produce software products for use in regulated environments they are subject to regular customer audit. The scope of these audits includes the QUMAS agile process for product development. The feedback from customers conducting these audits is that the time involved in performing the audit is greatly reduced as a result of the automated trace process. As the verification by the auditor of functionality implemented in the product via the agile process is now more effective and efficient, as the information is immediately retrievable in electronic format.

In order to verify that the agile process defined by QUMAS was in line with the expectations of their regulated customer base, QUMAS engaged with senior members of the GAMP EU and demonstrated the process. The feedback received was that the outlined process was deemed in accordance with the expectations of the industry.

D. Traceability

End-to-end traceability is a significant overhead in regulated environments. Traceability is often accomplished using spread-sheets which are printed and subsequently manually updated. Traceability is arguably the area in which the agile development process has had the most impact. The VP Development and Support characterized it as '*living traceability*' in that there is complete transparency into the development process at any point in time. In the past, documents and artifacts were produced periodically and collated to produce traceability evidence. Now there is full end-to-end traceability established by the toolset (see Table II). Links are automatically established as developers check in code that implements a certain task. Should a developer check in code without linking it to a task, then the dev check will identify this as an error. Initial requirements can be traced to stories, and in turn to tasks and sub-tasks, to design documentation, to source code, to code reviews, to builds, to unit tests, to rework and bug-fixes, to function and system testing, to production code. The toolset can be interrogated to trace which build fixed which bugs and which build implemented which functionality.

QUMAS undergo external audits of their development process about once per month. The extra transparency afforded by the implementation of their agile development approach has engendered further confidence to the extent that audits may now take place without requiring the attendance of the Product Manager and Test Manager. According to the VP Development and Support, the absence of these managers would not have been contemplated when audits were taking place in the past. Furthermore, audits which used to take two days are now being completed in less than a day, often with no open issues to respond to, and resounding approval from audit assessors who appreciate the complete transparency and flexibility afforded by the living traceability allowing them to interrogate aspects of the development process at will.

E. Verification and Validation

Requirements are validated directly with the Product Owner at the start of each development sprint. Unit tests are generated as part of the coding tasks. The unit tests are checked in with the functional code and therefore link to the code automatically. These tests are executed during the continuous build/deployment. The build automation is done via Bamboo, which also offers the option to invoke analytical tools, such as static code analyzers. Code changers and unit tests are run and changes to test results across builds can be easily linked to problematic check-ins of code. Unit tests are done within JIRA and functional tests are the responsibility of the test team using a specific quality center testing suite. In a typical build, a regression test suite of more than a thousand unit tests are run, which take 40 to 60 minutes to execute. This regressions test suite has been written by the developers over time, and new tests are added for new functionality and defect fixes. Any failures are recorded and emails are sent to the developers and Scrum Master. Continuous integration is implemented using the Bamboo tool. Every four hours the code base is monitored and any code check-ins trigger a new build at that point. Another tool that is integrated with Bamboo is NCover. This indicates the code coverage being tested and the goal in QUMAS is to achieve 80% coverage. Actual coverage can be monitored by NCover and presented in Bamboo reports.

A feature of the R-Scrum is the '*hardening sprint*' (see Fig. 4 above) which is run to ensure release readiness before final release. This ensures the shippable product versions from prior sprints can become a releasable product. QA will not sanction a release with any open issues. User documentation, structures on FTP site for customer download, marketing material etc. must all be integrated. At this stage, 'definition of done' must also include regulatory compliance. This reinforces the view that software development in regulated environments must satisfy two customers: the end-user and the regulatory bodies [53].

VIII. Discussion

The suitability of agile methods in regulated environments has long been an assumed limitation of these methods [2]. The findings of this study, however, show that agile and regulated environments are *not* incommensurable. In fact, our findings suggest that agile is highly suitable when tailored to meet the needs of regulated environments and supported with appropriate tools. Table III summarizes the findings of our study, organized per principle of regulated environments as identified in Fig. 2.

A. Lessons Learned and Open Issues

The key lesson from this study is that agile processes can, in fact, be augmented to work very well in regulated environments. Appropriate tool support is vital. QUMAS were able to implement an integrated toolset that replaced a suite of stand-alone systems for bug-tracking, code reviews, source code repository and document compliance. This integrated toolset did much to support full end-to-end traceability – an up-to-date accurate snapshot in real-time or '*living traceability*' as it is termed in this paper.

In terms of quality, a number of significant contributions arise through the agile process. For example, there is frequent alignment with business strategy. Also, QA acceptance of the new mode of working has been key to delivering quality in a mode of '*continuous compliance*' via QA Checkpoint audits at the end of each sprint. Development is more effective through the constant validation of product and sprint backlogs based on feedback from the Product Owner, QA and customers. The frequent releases and active engagement with customers means that customer requests can be facilitated within about five weeks. Continuous integration (every four hours) ensures that sales and marketing can demonstrate the latest functionality to customers, confident that the software will be fully functional.

One issue identified by management at QUMAS had to do with the perception of 'short termism' in planning-granularity that arises from the agile process. Because the product backlog tends to only include stories that are scheduled in the next two releases, this leads to a feeling that the planning horizon is more short term. Under the previous waterfall process, long-term requirements were identified in the design document to guide development over the longer term. However, the VP Development and Support acknowledged that this long-term view was largely a perception which was not always fulfilled, and the faster cadence of the agile process ensured more flexibility to respond to market changes and more accuracy in planning estimates.

The new process also has had major implications for the QA function in terms of faster conformance review cycles as product requirements were being delivered, tested and verified in a sprint approach as opposed to being delivered in one release candidate build for verification. QA now perform audits more frequently—every three weeks at the end of each sprint, rather than at final release time as in the previous process.

However, QA are completely engaged with the new process, a factor that was considered absolutely necessary by QUMAS management. The automated traceability also better supports the impact assessment from the QA side, when applying change to existing verified functionality.

TABLE III
KEY FINDINGS OF THE STUDY.

Concern	Assumed conflict - Agile in Regulated	Study Findings
Quality Assurance	Time-to-market is a key constraint recognized by agile methods and the concept of delivering 'good enough' working software in an optimum timescale takes precedence over 'perfect' software.	Quality enhanced by: • Product, release and sprint backlogs constantly validated with developers and customers. • Continuous integration and systematic refactoring. • QA function very supportive of agile process believing benefits outweigh inconvenience of changes to traditional working practices
Safety & Security	Agile thought to lack formal planning, risk mitigation.	• *Continuous compliance* • Risk also mitigated by *risk prioritization*–tackling the most significant risks first.
Effectiveness	Adherence to regulations and standards slows down development process and delivery speed to customer	• Frequent releases enable pre-sales and early delivery to customer. • Ability to rapidly respond to customer change request within 5 weeks. • Active management allows the Scrum Master to correct course on a daily basis. • Updates are visible in real time to all team members. • Documentation person to ensure a link between development, documentation and support. • Always up-to-date sales & marketing material.
Traceability	Lack of attention to documentation in agile inhibits traceability.	• Powerful toolset providing extensive and automatic *living traceability*. • Impact assessment of changes are easier to identify via the automated traceability. • QA conducts internal audits much more often; external audits are much shorter and done without key staff.
Verification & Validation	Requirements specification is time-consuming, testing	• Continuous integration supported by powerful toolset. • Automated tests and automatic link to code facilitate easy coverage reporting.

The change to writing stories to guide development is also a challenge, especially in terms of writing stories with the right level of detail and granularity. The Scrum Master identified an example as "the product shall be scalable" as an inappropriate choice. The principle adopted at QUMAS to guide story writing is that they be 'test-driven.'

B. Limitations of this Study

We are aware of a few limitations of this study, which we discuss next, following a common classification of reliability and external and construct validity [47]. Since our study does not seek to establish any causal relationships, we do not discuss threats to internal validity.

Reliability. In order to increase this study's reliability, we developed a study protocol as well as interview guides, as mentioned previously. As outlined in Section V, we employed several practices to establish the reliability of our study, such as prolonged involvement, data triangulation, peer debriefing and member checking, as well as establishing an audit trail.

Construct validity. As pointed out in Section III, we identified a number of recurring themes based on existing literature (see Fig. 2). While these themes are closely related, each has a distinctly separate focus and as such, taken together they provide an in-depth and multi-faceted perspective on the critical aspects of this topic.

External validity. A point that is often raised in case study research is that findings are not generalizable to other settings. However, the purpose of this study was revelatory and exploratory rather than explanatory. While there are many different regulated environments, this study gives an in-depth account of the application of agile methods in one such domain. The augmented Scrum model shown in Fig. 4 can provide a starting point for other organizations in regulated domains. More research is necessary to establish how Scrum and other agile methods can be scaled to other regulated domains.

IX. CONCLUSIONS AND FUTURE WORK

This paper presents an in-depth account of how agile methods can be scaled to regulated environments. Given the successful exemplars of the use of agile methods in large teams and distributed development, the use of agile methods in regulated environments may be seen as the '*final frontier*' for agile methods.

Overall, the agile development process as it has been adopted and augmented in QUMAS has worked very well in the regulated environment. Compliance is more immediate and evident in real-time—*continuous compliance* as we have labeled it here. Also, the concept of *living traceability* has been coined to reflect the end-to-end traceability that has been facilitated by the toolset that has been implemented to support the agile development process. In summary, it seems to be the case that the assumption of incompatibility between agile methods and regulated environments is more *accidental* than *essential*. Thus, the V-lifecycle model which is frequently adopted in regulated environments appears compelling in that there is a clear sense of traceability between the levels of resting and the levels of analysis, design and coding activities. However, as pointed out by one of our practitioner interviewees: "*Agile is a lot of small Vs*" and the levels traceability can clearly be accomplished in the agile mode of development also.

Developers and QA have also embraced the agile development process very enthusiastically and can see major benefits. However, senior management are no less enthusiastic about the new process. This is an important issue as research suggests that agile methods are developer-centric and are

typically enthusiastically embraced by developers, but management require some convincing as to the actual business benefits of agile methods. Agile methods are perceived to be unsuitable, and not adopted lest they endanger the organization's processes and reputation. However, the business benefits have been very evident in QUMAS, and these findings may provide direct motivation to other organizations that wish to explore how they can benefit from adopting agile methods. To that end, the standard Scrum model that is "wrapped" with additional elements (artifacts and roles) offered in Fig. 4—that we labeled *R-Scrum*—provides a directly usable framework that can be adopted or further tailored as needed.

We plan to build further on the findings of this study as follows. Firstly, we are conducting further case studies in other regulated domains. Secondly, we will also extend our study to other agile methods, in particular XP. Furthermore, while the current study presents a qualitative account of augmenting the Scrum framework, we are also designing quantitative studies to study this topic in further detail.

ACKNOWLEDGMENTS

The authors wish to thank Joanne O'Driscoll for her input to this paper. This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero.

REFERENCES

[1] VersionOne, "6th annual state of agile survey: The state of agile development," 2011.
[2] P. Abrahamsson, K. Conboy, and X. Wang, "'lots done, more to do': the current state of agile systems development research," *European Journal of Information Systems*, vol. 18, 2009.
[3] L. Williams and A. Cockburn, "It's about feedback and change," *IEEE Comput.*, vol. 36, no. 6, 2003.
[4] L. Cao, K. Mohan, P. Xu, and B. Ramesh, "How extreme does extreme programming have to be? adapting xp practices to large-scale projects," in *37th Hawaii Int'l Conf. System Sci.*, 2004.
[5] T. Kähkönen, "Agile methods for large organisations–building communities of practice," in *Agile Development Conference*, 2004.
[6] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kähkönen, "Agile software development in large organisations," *IEEE Comput.*, vol. 37, pp. 27–34, 2004.
[7] M. Kircher, P. Jain, A. Corsaro, and D. Levine, "Distributed extreme programming," in *XP2001-Extreme Programming and Flexible Processes in Software Engineering*, 2001.
[8] D. Stotts, L. Williams, N. Nagappan, P. Baheti, D. Jen, and A. Jackson, "Virtual teaming: Experiments and experiences with distributed pair programming," in *Extreme Programming/Agile Universe*, 2003.
[9] D. Boland and B. Fitzgerald, "Transitioning from a co-located to a globally-distributed software development team: A case study at Analog Devices, Inc." in *3rd Workshop on Global Software Development*, 2004.
[10] O. Cawley, X. Wang, and I. Richardson, "Lean/agile software development methodologies in regulated environments–state of the art," in *Int'l Conf. Lean Enterprise Software and Systems, LNBIP 65*, 2010.
[11] B. Boehm, "Get ready for agile methods, with care," *IEEE Comput.*, vol. 35, pp. 64–69, 2002.
[12] S. W. Ambler, "When does(n't) agile modeling make sense?" 2001, www.agilemodeling.com/essays/whenDoesAMWork.htm.
[13] M. Mc Hugh, F. Mc Caffery, and V. Casey, "Standalone software as an active medical device," in *11th Int'l SPICE Conference*, 2011.
[14] D. Turk, R. France, and B. Rumpe, "Assumptions underlying agile software-development processes," *J. Datab. Manage.*, vol. 16, 2005.
[15] "Agile Manifesto," http://agilemanifesto.org/.
[16] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, pp. 28–32, 2001.
[17] K. Beck, *Extreme Programming Explained*. Addison-Wesley, 1999.
[18] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, Upper Saddle River, NJ, 2002.
[19] A. Cockburn, *Crystal Clear: A Human-Powered Software Development Methodology for Small Teams*. Addison-Wesley, Reading, MA, 2001.
[20] J. Stapleton, *DSDM: Dynamic Systems Development Method*. Addison-Wesley, Harlow, England, 1997.
[21] S. W. Ambler, *Agile Modeling: Best Practices for the Unified Process and Extreme Programming*. Wiley, 2002.
[22] P. Coad, E. Lefebre, and J. De Luca, *Java Modelling in Color*. Prentice Hall, Englewood Cliffs, NJ, 1999.
[23] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional, 2003.
[24] P. Kruchten, *The Rational Unified Process: An Introduction*, 2nd ed. Addison-Wesley Professional, 2000.
[25] H. Takeuchi and I. Nonaka, "The new new product development game," *Harvard Business Review (January-February)*, 1986.
[26] J. Sutherland and K. Schwaber, "Business object design and implementation," in *OOPSLA '95 Workshop Proceedings*, 1995.
[27] ——, "The scrum guide," 2011, www.scrum.org/Scrum-Guides.
[28] P. Deemer and G. Benefield, "The scrum primer: An introduction to agile project management with scrum," 2007, version 1.04.
[29] J. Sutherland and K. Schwaber, "The scrum papers: Nuts, bolts and origins of an agile method," 2007.
[30] L. Coyle, M. Hinchey, B. Nuseibeh, and J. Fiadeiro, "Guest editors' introduction: Evolving critical systems," *IEEE Comput.*, vol. 43, 2010.
[31] "IEEE Standard Glossary of Software Engineering Terminology," 1990, IEEE 610.12.
[32] S. L. Pfleeger, N. Fenton, and N. Page, "Evaluating software engineering standards," *IEEE Comput.*, vol. 27, no. 9, pp. 71–79, 1994.
[33] W. Deming, *Out of the Crisis*. MIT Center for Advanced Engineering Study, Cambridge, MA, 1982.
[34] R. Vidgen and X. Wang, "Coevolving systems and the organization of agile software development," *Inf. Sys. Res.*, vol. 20, no. 3, 2009.
[35] B. Fitzgerald, "Formalized systems development methodologies: A critical perspective," *Information Systems Journal*, vol. 6, 1996.
[36] N. Leveson and C. Turner, "An Investigation of the Therac-25 Accidents," *IEEE Comput.*, vol. 26, no. 7, pp. 18–41, 1993.
[37] N. Mead, "Who is Liable for Insecure Systems," *IEEE Comput.*, vol. 37, no. 7, pp. 27–34, 2004.
[38] P. Abrahamsson, M. Ali Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Softw.*, vol. 27, no. 2, 2010.
[39] A. Tribble, "Software Safety," *IEEE Softw.*, vol. 16, no. 1, 2002.
[40] T. DeMarco and B. Boehm, "The Agile Methods Fray," *IEEE Comput.*, vol. 35, no. 6, pp. 90–92, 2002.
[41] J. Cleland-Huang, *Traceability in Agile Projects*. Springer, London, 2012, pp. 265–275.
[42] S. Rakitin, *Software Verification and Validation for Practitioners and Managers*, 2nd ed. Artech House, 2001.
[43] F. Mc Caffery, M. Pikkarainen, and I. Richardson, "AHAA - Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs," in *Int'l Conf. Software Engineering*, 2008.
[44] S. VanderLeest and A. Buter, "Escape the waterfall: Agile for aerospace," in *28th Digital Avionics Systems Conference*, 2009.
[45] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability*. Springer, London, 2012.
[46] K. Gary, A. Enquobahrie, L. Ibanez, P. Cheng, Z. Yaniv, K. Cleary, S. Kokoori, B. Muffih, and J. Heidenreich, "Agile methods for open source safety-critical software," *Softw. Pract. Exper.*, vol. 41, no. 9, 2011.
[47] R. Yin, *Case Study Research: Design and Methods*, 3rd ed. SAGE Publications, Thousand Oaks, CA, USA, 2003.
[48] A. Lee and R. Baskerville, "Generalising generalisability in information systems research," *Information Systems Research*, vol. 14, 2003.
[49] K. Eisenhardt, "Building theories from case study research," *The Academy of Management Review*, vol. 14, no. 4, 1989.
[50] P. Brereton, B. Kitchenham, D. Budgen, and Z. Li, "Using a protocol template for case study planning," in *EASE*, 2008.
[51] J. Creswell and D. Miller, "Determining validity in qualitative inquiry," *Theory into Practice*, vol. 39, no. 3, 2000.
[52] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Trans. Softw. Engineer.*, vol. 24, no. 4, 1999.
[53] M. Mc Hugh, F. Mc Caffery, B. Fitzgerald, K. Stol, V. Casey, and G. Coady, "Balancing agility and discipline in a medical device software organization," in *13th Int'l SPICE Conference*, 2013.