2020-03-02

INCORPORATING DIGITAL ETHICS THROUGHOUT THE SOFTWARE DEVELOPMENT PROCESS

Michael Collins
Technological University Dublin, michael.collins@tudublin.ie

Damian Gordon
Technological University Dublin

Anna Becevel
Technological University Dublin, anna.becevel@tudublin.ie

See next page for additional authors

Recommended Citation

Gordon, D., Collins, M., Becevel, A., & O'Mahony, W. (2022). INCORPORATING DIGITAL ETHICS THROUGHOUT THE SOFTWARE DEVELOPMENT PROCESS. Technological University Dublin. DOI: 10.21427/QN74-9E40

INCORPORATING DIGITAL ETHICS THROUGHOUT THE SOFTWARE DEVELOPMENT PROCESS

D. Gordon[1], M. Collins[2], A. Becevel[3], W. O'Mahony[4]

[1,2,3,4]*Technological University Dublin (IRELAND)*

Abstract

The media is reporting scandals associated with computer companies with increasing regularity; whether it is the misuse of user data, breach of privacy concerns, the use of biased artificial intelligence, or the problems of automated vehicles. Because of these complex issues, there is a growing need to equip computer science students with a deep appreciation of ethics, and to ensure that in the future they will develop computer systems that are ethically-based. One particularly useful strand of their education to incorporate ethics into is when teaching them about the formal approaches to developing computer systems.

There are a number of specific processes and methodologies that incorporate these stages in different ways into their approaches. Some take a linear approach to these stages, whereas others take a more iterative and/or incremental approach. These models include the Waterfall Model, the V-Model, the Spiral Model, and the Agile family of models. For each of these models this paper will present a way to include ethics in the Specifying stage, and well as threaded throughout the model, and as an explicit stage in a final review process at the end of the implementation stage.

These formal models are understood (and used) by computer companies all over the world, and therefore are a natural means of incorporating ethics into software development in a manner that would not seem overly arduous or unwieldy to developers. These techniques are also taught in the computer science departments of universities all over the world, it is therefore vitally important that lecturers incorporate an ethical dimension into their systems development teaching, and we believe that these newly refined models provide them with a simple means of achieving this task, and this will make a new generation of software developers ethically-aware.

Keywords: Ethics, Behaviour, Software, Methodology, Process

1 INTRODUCTION

An appreciation of computer science (CS) ethics has never been more important than today. There are an abundance of examples of challenges associated with CS ethics, including social network companies trying to influence the emotions of their users, and their data used to try to influence the outcomes of democratic processes, from the challenges of driverless cars to issues around tracking and privacy. The Ethics4EU ("ethics for you") project is a transnational project, supported by the Erasmus+ programme of the European Union, that proposes to explore the issues of teaching ethics in computer science. To this end, the project will undertake:

- a desk-based research process to explore the teaching of CS ethics in a European context
- an accompanying survey of computer science departments to get more detail on content being delivered, as well as the teaching methods being used
- the engagement of a community of practice of computer science lecturers to share their perspectives and experience on teaching CS ethics
- the development of a number of training events to share perspectives and experience
- the authoring of an easy-to-use instructor guide to aid lecturers in teaching ethics
- the development of teaching content that will include content both for the delivery of a module focused solely on ethics in computer science. It will also create a number of lessons that readily incorporated in pre-existing modules, e.g. a lesson on ethics in databases, another one that could be used in an artificial intelligence class, another one in a robotics class, and one in a computer security class.

Software applications often encompass different disciplines such as business, pharmaceutical, military and agriculture. Therefore, the decisions made by the software architects and engineers in the creation of these applications have a profound effect on all stakeholders. These software architects and

engineers need to consider the dichotomy of technical and non-technical elements in the software development process, paying particular attention to ethical implications. Throughout the software development process, key ethical decisions are made that the developers must be held accountable even when such decisions may conflict with the interests of the stakeholders concerned. Even though there are general codes of ethics in software engineering (e.g. the ACM Code of Ethics and Professional Conduct[1]), it is often the case that these are deemed too generic and that specific ethical decisions need to be made in the development of each software application [12]. Therefore, all software engineering practitioners have a responsibility to ensure, as best as possible, that ethical consideration is included in their decision making.

Throughout the software lifecycle in many methodologies, a validation and verification process is facilitated at various stages. This tests the quality of the software, e.g., how it is built and how it behaves in different situations. This process often involves large sets of data furnished either by the company developing the software itself or a third-party [2]. This means that companies have to trust those responsible with a variety of data that can be vital to the company's interest. Examples include defect data, product source code and even the morale of the software developers. Therefore, there are reasonable expectations that ethical behaviour is employed at each stage in the lifecycle when using this data. However, ethical behaviour can be subjective with contrasting opinions, meaning it is very difficult to legislate. Therefore, it is often the case that safeguards are implemented throughout the software lifecycle to protect the interests and limiting the liabilities of the various stakeholders.

In [2], the authors point out some limitations in the safeguards utilised to enforce ethical behaviour throughout the software development process. These involve (i) trust with information, (ii) conflict of interest, and (iii) regulatory compliance. However, the authors state that there can be any number of safeguards, guidelines, policies and procedures employed to manage situations and encourage ethical behaviour. Unfortunately, none of these will guarantee what would be deemed “correct” ethical behaviour in developing software, thus meaning a reliance on mutual trust between all stakeholders.

In [7], the authors discuss the notion of Ethical-Driven Software Development (EDSD), a framework to raise awareness of ethics prior to, and during, the product’s development cycle. The EDSD framework is not a software development process, but rather a companion process that has questions addressing each of the phases in the development process. It provides a set of index cards as a tool to help facilitate the development process.

2 SOFTWARE DEVELOPMENT MODELS

Typically a computer system is developed when a customer (or user) approaches a computer company and asks them to write that system for them, to address some specific requirements. Developing the computer system so that it both works and satisfies the needs of its intended users is a reasonably complex process, and it usually consists of a number of standard phases, including specifying, designing, implementing, deploying and maintaining the system. In detail, these stages can be described as follows:

- **Specifying**: This stage involves understanding the requirements of the intended users fully, and articulating them in a coherent and unambiguous way. This is often composed of two substages; “System Specifying” which concerns the functional requirements, and “Software Specifying” which concern the technical requirements.
- **Designing**: This stage is analogous to an architect creating a blueprint for a house, it involves software designers taking the specification from the previous stage and describing the potential way(s) that a system can be built to satisfy those requirements.
- **Implementing**: This is the stage where the system is built, based on the previous stage, and includes decisions related to the type of operating system(s), programming language(s), and database(s) that will be used to build the system.
- **Deploying**: This stage means that the implemented software is taken from the computer company and transferred onto the users’ computer system, which may require some alterations to the software. This stage may also require the development of some support documentation or training.

[1] https://www.acm.org/code-of-ethics

- **Maintaining**: This stage occurs after the system is up-and-running for the users, the users may require the development of additional features once they become familiar with the existing features.

There are a number of specific processes and methodologies that incorporate these stages in different ways into their approaches. Some take a linear approach to these stages, whereas others take a more iterative and/or incremental approach. As these methodologies already exist, it is worth exploring how easy it is to incorporate digital ethics into them.

2.1 The Waterfall Model

One of the oldest methodologies is the so-called "Waterfall model" which was first formally described in [9]. His paper "*Managing the Development of Large Software Systems*", consists of the stages outlined above (and further stages), but with a check at the end of each stage to ensure that the stage meets the needs of the user ("Validation"), and also that the stage satisfies constraints of the previous stage ("Verification").

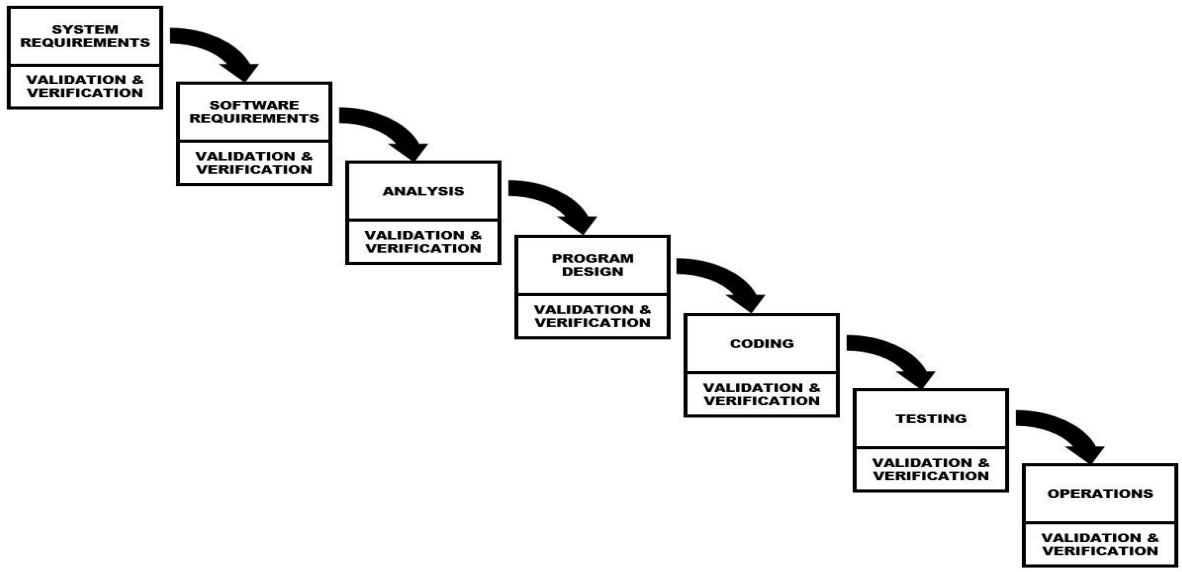

Figure 1. The Waterfall Model [9]

The model is defined as a linear model, meaning that it represents a method as to how software can be developed where each stage cannot commence until the previous stage is completed. It typically involves detailed documentation of all of the decisions and actions taken during the development. The main stages of the model are as follows:

- *System Requirements*: Identify, select and document functional, scheduling and financial requirements.
- *Software Requirements*: Identify, select and document the software features necessary to satisfy the system requirements.
- *Analysis*: Methodically work through the details of each requirement.
- *Program Design*: Use programming techniques to design software and hardware within the constraints and objectives set in the earlier stages.
- *Coding*: Implement the program as designed in the earlier stages.
- *Testing*: Test the software and record the results.
- *Operations*: Deliver, install and configure the completed software.

It is worth noting that the purpose of Royce's original paper [9] was to point out that this model is a risky one, as it leaves the testing of the software until late in the process, nonetheless, the Waterfall model is recognised as working well for smaller projects where requirements are very well understood, and because of the high level of documentation, new developers can easily join the process and understand the objectives of a new project.

Incorporating ethics into the Waterfall Model could be done as follows:

- An ethical statement could be explicitly incorporated into the System Requirements stage of the process, for example, a requirement could be added to state that "*the system will be built in an ethical manner, and it will function in such a way that it will be extremely difficult to use in an unethical way*".
- Ethics could also easily be incorporated into each of the end-of-stage "validation and verification" checks (using the EDSD framework), which are already exploring whether or not the system is meeting the requirements, therefore an additional ethical check at the end of each stage would be trivial.

2.2 The V-Model (or Vee Model)

Another approach, entitled the "V-Model" (or "Vee model") was detailed in [6], and also incorporates all of the above stages outlined in Section 2.1. However, between the *Deploying* and *Maintaining* stages are four more stages that explicitly test the previous four stages in reverse order, i.e., after the *Deploying* stage there is a *Test Deploying* stage, a *Test Implementing* stage, a *Test Designing* stage, and a *Test Specifying* stages. These stages verify that the system meets the users' requirements, in terms of meeting the specifications, in terms of system performance, and in terms of documentation requirements.

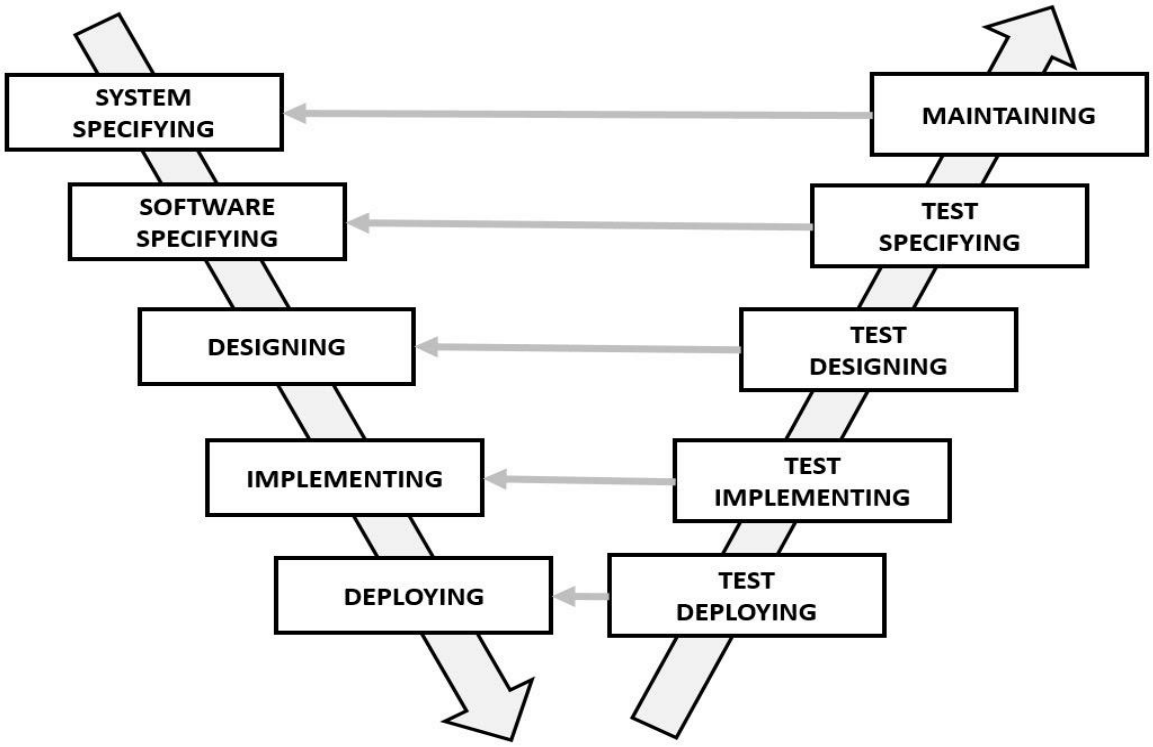

Figure 2. The V-Model (or Vee Model) [6]

This model is also defined as a linear model, meaning that it represents a method as to how software can be developed where each stage cannot commence until the previous stage is completed. It typically involves detailed documentation of all of the decisions and actions taken during the development. The main stages of the model are as follows:

- *System Specifying*: Identify, select and document functional, scheduling and financial requirements.
- *Software Specifying*: Identify, select and document the software features necessary to satisfy the system requirements.
- *Designing*: Use programming techniques to design software and hardware within the constraints and objectives set in the earlier stages.

- *Implementing*: Build the program as designed in the earlier stages.
- *Deploying*: Program is taken from the computer company and transferred onto the users' computer system.
- *Test Deploying*: Undertake Unit Tests and record the results.
- *Test Implementing*: Undertake Integration Tests and record the results.
- *Test Designing*: Undertake System Tests and record the results.
- *Test Specifying*: Undertake Acceptance Tests and record the results.
- *Maintaining*: The upkeep of the completed software.

The V-Model should be used for small to medium sized projects where requirements are clearly defined and fixed, and should be chosen when ample technical resources are available with needed technical expertise. The design of the testing happens well before the coding, which saves a lot of time, and hence there is a higher chance of success over the waterfall model.

Incorporating ethics into the V-Model could be done as follows:

- An ethical statement could be explicitly incorporated into the System Specifying stage of the process, for example, a requirement could be added to state that "*the system will be built in an ethical manner, and it will function in such a way that it will be extremely difficult to use in an unethical way*".

- Ethics could also easily be incorporated into each of the four testing stages (using the EDSD framework), which are already exploring whether or not the system is meeting the requirements, therefore an additional ethical check at the end of each stage would be trivial.

2.3 The Spiral Model

An iterative model called the "Spiral model" [5], is where the five stages outlined above are present, but instead of trying to build a complete system, on the first iteration the developers aim to create a basic prototype with only some of the required features, and then following this another iteration is undertaken where all five stages are again revisited until a new prototype with additional features are developed. This process of prototype development will continue until the final prototype is produced that is sufficient for the users. To help in the determination of when to stop, at the end of each iteration, a new stage, a *Risk* stage, is included to determine how far away the prototype is from the required system, and exploring if the cost of continuing with another iteration is higher than the benefits accrued on completing it.

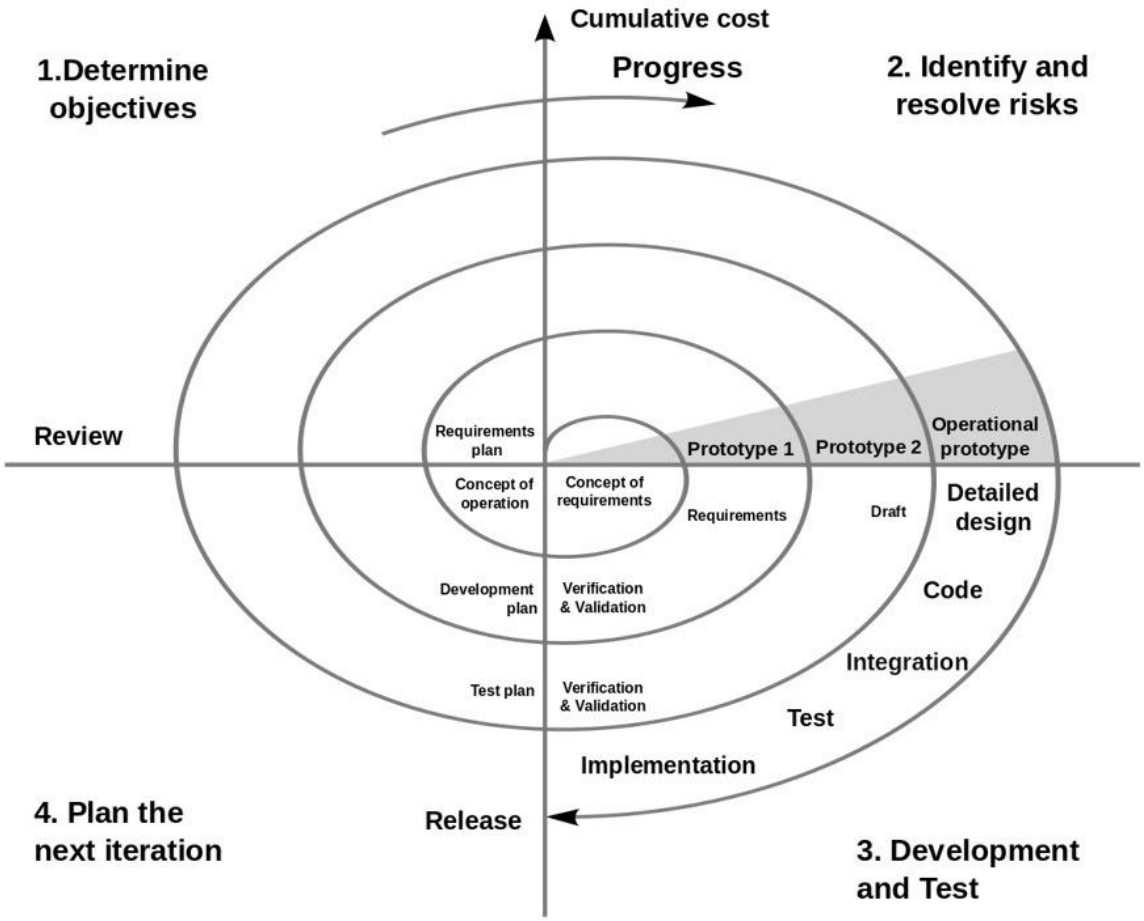

Figure 3. The Spiral Model [5]

This model is also defined as an iterative model, meaning that a prototype is quickly developed using prototyping tools, and features are added onto to the prototype until an operational system is completed. It typically involves detailed documentation of all of the decisions and actions taken during the development. The main stages are as follows:

1. *Determine Objectives*: This involves defining functionality, performance, hardware/software interface, critical success factors, etc. As well as exploring alternatives and constraints.
2. *Identify and Resolve Risks*: Identify risks such as lack of experience, new technology, tight schedules, etc. Resolve the identified risks evaluating their impact on the project/
3. *Develop and Test*: Create and review a design, develop and test code, and test the overall development.
4. *Plan the Next Iteration*: Develop the project plan, and configuration management plan. Also develop a test plan, and develop an installation plan.

The Spiral Model is very useful for medium to high-risk projects, where the users are unsure of their needs. As a result of the high level of risk analysis, the avoidance of risk is significantly enhanced.

Incorporating ethics into the Spiral Model could be done as follows:

- An ethical statement could be explicitly incorporated into the initial *Concept of Requirements* stage of the process, for example, a requirement could be added to state that "*the system will be built in an ethical manner, and it will function in such a way that it will be extremely difficult to use in an unethical way*".

- For each iteration of the Spiral mode, ethics could also be incorporated into the *Risk* stage of the process (using the EDSD framework), which are already weighing costs and benefits, therefore if any ethical breaches were considered to be part of the "Costs" an ethical check could be done in a reasonably straightforward way.

2.4 The Agile Models

A final set of models worth mentioning at the "Agile" family of software development methodologies, initially developed in 2001 on a skiing trip by seventeen of the leading software designers (including Kent Beck, Jim Highsmith, Steve Mellor, Arie van Bennekum, Ken Schwaber, Alistair Cockburn, Jeff Sutherland, Ward Cunningham, and Martin Fowler), the Agile approach favours responding to change instead of following a plan, it favours customer collaboration over contract negotiations, and it favours developing working software over developing comprehensive documentation. These approaches usually have stages equivalent to the above five stages, but the cycle through the stages is much quicker, and the system is typically divided into a series of elementary features, that should take no more than two weeks to develop (a "sprint"). Once a feature is completed, a quick retrospective process is held. We'll look at a specific Agile model called the SCRUM model [10] (based on work by [11]).

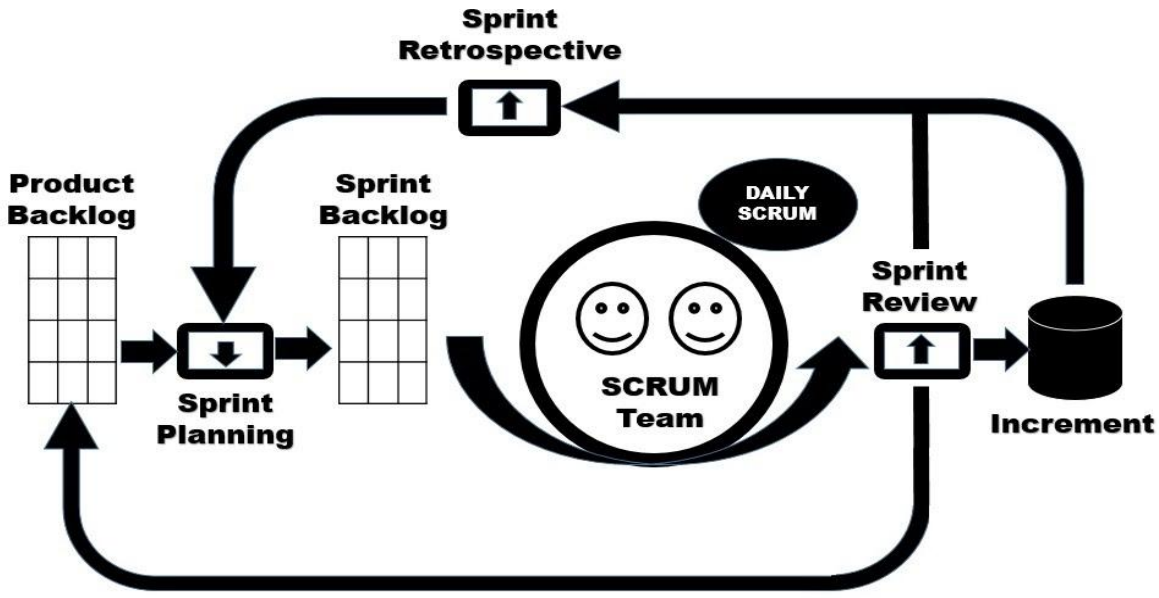

Figure 4. The SCRUM Model [4]

The SCRUM model works as follows:

- *Sprints*: The project is divided into a series of mini-projects (called "sprints"), which are timeboxed to no more than a month long, most commonly two weeks.
- *Sprint Planning*: A planning meeting at the start of the sprint, where team members figure out how many items they can commit to, and then create a sprint backlog – a list of tasks to perform during the sprint.
- *Daily SCRUM*: At the end of each day of the sprint, all team members attend a daily Scrum meeting. This meeting is timeboxed to no more than 15 minutes.
- *Sprint Review*: At the end of a sprint, the team conducts a sprint review during which the team demonstrates the new functionality.
- *Sprint Retrospective*: At the end of each sprint the whole team participates in this meeting, which is an opportunity to reflect on the sprint that has ended, and identify opportunities to improve.

Scrum is a great solution to support rapid project progress of almost any type of project. It is extremely effective in creating agility for any organization. It also easily adapts to feedback from customers and stakeholders. Another benefit is that the individual effort of each team member is recognised during daily scrum meetings.

Incorporating ethics into the SCRUM Model is slightly more complex, because of the granular nature of the SCRUM approach, an overall ethical perspective on the system is more difficult to track, but it could be done as follows:

- An ethical statement could be explicitly incorporated into the initial *Concept of Requirements* stage of the process, for example, a requirement could be added to state that "*the system will*

be built in an ethical manner, and it will function in such a way that it will be extremely difficult to use in an unethical way".

- For each *Sprint Retrospective*, an ethical review could be mandatory at this point (using the EDSD framework).

- To ensure an overall ethical perspective is maintained, a new explicit *Ethics Review* stage is needed at the end of each development, to ensure that the system developed meets the ethical requirements (also using the EDSD framework).

It is also worth noting that since the agile methods do not focus on documentation as much as the previous models, and explicitly state the need for "just enough" documentation; this may present an additional challenge for ethical monitoring.

3 DISCUSSION

The next generation of software developers will have to have a greater ethical awareness, as is evident by the myriad of software scandals caused by unethical choices made by developers. To ensure that this occurs, a central philosophical perspective of the Ethics4EU project is that ethics should be imbued throughout the curriculum, and not simply taught as a stand-alone module. Therefore there is a need to incorporate ethics into all aspects of the development process, including into software development models. By incorporating ethics into these models, software developers will be considering ethics as a default stance.

This paper reviewed four key models of software development models, ranging from linear models to iterative models to agile models, and we showed that it is possible to incorporate ethics checks into this wide range of models.

The analysis in this paper has raised some interesting issues, for example, it is worth noting that it is relatively trivial to incorporate an ethics check into the older, more monolithic models of development, whereas with the more modern, agile approaches, because of the granularity of the sprints, typically 1-2 weeks, it is more complex to include an ethical perspective into the process. As pointed out in Section 2.4, it might be difficult to appreciate the cumulative ethical implications of the interactions of the individual components, particularly when faced with the challenges of *concept drift*, whereby small, and seemingly insignificant, development deviations from the initial design specification cumulatively result in a significantly distinct system than was planned. Therefore a final, new stage is required, which consists of an explicit ethics check.

Interestingly, [1] compared two large software development projects, both using SCRUM-like methodologies, one of which incorporated the EDSD framework in its development cycle, and the other did not. The authors found that the inclusion of ethical tools in the process did improve the quality of that project, specifically in the areas of schedule, product functionality and cost. This outcome suggests that ethics is not only the right thing to do, but also extremely beneficial to the success of a project.

ACKNOWLEDGEMENTS

The authors of this paper and the participants of the Ethics4EU project gratefully acknowledge the support of the Erasmus+ programme of the European Union. The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

REFERENCES

[1] Abdulhalim, H., Lurie, Y., Mark, S. (2018) "Ethics as a Quality Driver in Agile Software Projects", *Journal of Service Science and Management*, p. 11, 13-25.

[2] Andrews, Anneliese Amschler., Pradhan, Arundeep S., "Ethical Issues in Empirical Software Engineering: The Limits of Policy", *Published in Empirical Software Engineering, Dordrecht*, June 2001, pp. 105-110, Springer-Nature Scholarly Journal.

[3] Barnard, A., de Ridder, C., Pretorius, L. and Cohen, E. (2003) “Integrating Computer Ethics into the Computing Curriculum: A Framework for Implementation”, *Proceedings of IS2003, Informing Science+ Information Technology Education Joint Conference*, pp. 265-279.

[4] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001) “Manifesto for Agile Software Development”, *Agile Alliance*, http://agilemanifesto.org.

[5] Boehm, B. (1998) "A Spiral Model of Software Development and Enhancement", *Computer* 21(5), pp. 61-72.

[6] Forsberg, K., Mooz, H. (1991) "The Relationship of System Engineering to the Project Cycle", *Proceedings of the National Council for Systems Engineering (NCOSE) Conference*, pp. 57-65.

[7] Lurie, Y., Mark, S. (2015) “Professional Ethics of Software Engineers: An Ethical Framework”, *Science and Engineering Ethics*, p22, 417-434.

[8] Oriogun, Peter., Akinbule, Olatunji., Ibecheozor, Chinwe., Nyako, Zayyad., “Software Engineering Ethical Decision Making and Professional Responsibility”, *Proceedings of the 2012 African Conference for Software Engineering and Applied Computing (ACSEAC’12)*, September 2012, pp. 7-14.

[9] Royce, W.W. (1970) "Managing the Development of Large Software Systems", *Proceedings of IEEE WESCON* 26 (August), pp. 1-9.

[10] Sutherland, J.V., Schwaber, K. (1995) “Business object design and implementation”, *OOPSLA '95 Workshop Proceedings*. ISBN 978-3-540-76096-2.

[11] Takeuchi, H., Nonaka, I. (1986) "The New New Product Development Game". *Harvard Business Review*, 64, pp. 137-146.

[12] Webley, S., Werner, A. (2008) “Corporate Codes of Ethics: Necessary but not Sufficient”, *Business Ethics: A European Review*, 17(4), pp. 405-415.