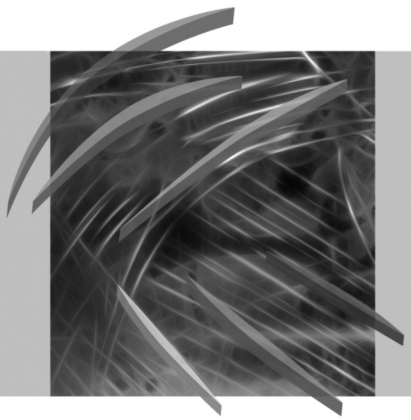
PRACE NAUKOWE
Uniwersytetu Ekonomicznego we Wrocławiu
RESEARCH PAPERS
of Wrocław University of Economics

205

Advanced Information Technologies for Management – AITM 2011

Information Systems in Business



edited by
Jerzy Korczak, Helena Dudycz, Mirosław Dyczkowski

Publishing House of Wrocław University of Economics
Wrocław 2011

Reviewers: Frederic Andres, Witold Chmielarz, Jacek Cypryjański, Beata Czarnacka-Chrobot, Bernard F. Kubiak, Wojciech Olejniczak, Celina M. Olszak, Marcin Sikorski, Ewa Ziemba

Copy-editing: Agnieszka Flasińska

Layout: Barbara Łopusiewicz

Proof-reading: Marcin Orszulak

Typesetting: Adam Dębski

Cover design: Beata Dębska

This publication is available at www.ibuk.pl

Abstracts of published papers are available in the international database The Central European Journal of Social Sciences and Humanities http://cejsh.icm.edu.pl
and in The Central and Eastern European Online Library www.ceeol.com

Information on submitting and reviewing papers is available on the Publishing House's website www.wydawnictwo.ue.wroc.pl

ISSN 1899-3192
ISBN 978-83-7695-178-2

The original version: printed

Printing: Printing House TOTEM

Contents

Streszczenia

PRACE NAUKOWE UNIWERSYTETU EKONOMICZNEGO WE WROCŁAWIU nr 205
RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS

Advanced Information Technologies for Management – AITM 2011 ISSN 1899-3192
Information Systems in Business

Payam Homayounfar
Wrocław University of Economics, Wrocław, Poland
International Program Manager, Braunschweig, Germany
e-mail: p.homayounfar@gmail.com

LIMITATIONS OF AGILE SOFTWARE DEVELOPMENT METHOD IN HEALTH CARE

Abstract: It became very popular in the last years to use Agile Software Development Methods (ASDM) to develop software in different areas of economy. Each new developed method aims to face new situations and solve current issues of problems for software development and project management. Each project has to be treated individually as the frame and environment conditions are different in every project and every industry. The agile approach can be useful if it is applied in the right situation, but affects the opposite in the wrong environment. Especially the health care sector has its own specifications which lead to the limitations of agile software development methods. This paper aims to show the limitations of agile software development methods in general and health care.

Keywords: agile software development, eXtreme Programming, SCRUM, Agile Manifesto.

1. Introduction

In 2001 the Agile Alliance with 21 members developed the manifesto for agile software development [Fowler, Highsmith 2001; Highsmith 2009] to provide a solution for new requirements in software development. These requirements have been evolved from the limitations of the waterfall or spiral model based approaches. The old models seem to be not sufficient enough to cover today's volatile requirements. An increasing number of software development projects struggle in achieving the time, budget and quality target triangle. Scientific literature documents the missing flexibility of waterfall based software development methods and leads to the agile methods. The agile software development methods have their existing legitimacy in terms of solving specific issues. The limitation of the agile software development methods will be formulated and explained in this paper.

Firstly, it is necessary to explain the idea and the body of agile software development methods to build the foundation of the following formulations. Based on the agile approach, different underlying software development methodologies have been developed and inherited the characteristics of the agile approach. These methods

like SCRUM will be named and very briefly described. The limitations of ASDM in health care do not need to be discussed based on a specific method. All methods are based on the same agile axioms which are the source of limitations if they are used in the wrong environment.

Health care is one of the most complex industries with specific needs for software development and implementation projects. Many different parties have to be involved in the process of communication for definition of the requirements before the implementation can begin. Software development in health care should reflect medical knowledge, organizational information, processes in health care, strategic topics, financial and economical effects, and technological aspects. The technological aspects cover information technology as well as the underlying medical technologies like health standards and different modalities, e.g. picture retrieving, processing, and storing technologies. A key success factor for software development in health care is the situation-related choice of the method of approach as a tool [Homayounfar, Owoc 2011b]. The dangers and limitation factors of ASDM will be described. The general and undisputed advantages of ASDM for particular areas can become risks and disadvantages in health care because specific needs are addressed to software development methods in the health care industry.

The paper is based on a normative scientific approach with deductive reasoning. The hypothesis discussed in this paper is H0: Agile software development methods are not sufficient for being applied in health care. The conclusion will show that a new holistic method is required for health care, which has to be adapted in each individual case.

2. Specification of Agile Software Development Methods (ASDM)

ASDM aim to support early and rapid production of working software code. This is achieved by structuring the development process into iterations. This code-focused approach of ASDM is based on the technical background of the developers of this method. In the past, the software development had to face the change of requirements after the customer had finished the waterfall phase of requirement definition. This is a natural process, as the business and technical knowledge gain of customer's raises increasingly during a project. In ASDM it is not necessary to know all requirements to begin with the coding of applications [Ambler 2007]. In an iterative process, the customer works together with IT developers to realize the programs and application [Collyer 2009].

In the last two decades many software development methods have been evolved which claim to be agile. The most important methods eXtreme Programming and SCRUM will be described with their characteristics. Some other methodologies will be briefly mentioned to show that there are many similarities of ASDM. Finally, the Agile Manifesto will be described which links the methods together and builds a value and approach framework over all agile methods.

2.1. eXtreme Programming (XP)

XP is an approach introduced by Kent Beck, for the development of a software which takes a code-centric view on the activities with the core values communication, simplicity, feedback and courage [Beck 1999]. Communication is perceived as an important means to retrieve tacit knowledge concerning the requirements of customers. The value of simplicity follows the goal to reduce the complexity of functions to the maximum to achieve speed and agility. Feedbacks of customers help to identify mislead development. Courage expects from the participants to try new directions in the development process. XP has the following characteristics [Hunt 2006; Cockburn 2002; Wolak 2001; Beck 1999]:

– comprehensive unit tests,
– short release cycles by adding only features needed for the current task in the order of importance,
– collective code ownership,
– continual improvement.

The development environment in an organization which uses XP is characterized by these procedures [Cockburn 2002; Beck 1999]:

– customer lists the must-have features of the software,
– programmers break the features down into stand-alone tasks and estimate the work needed to complete each task,
– customer chooses the most important tasks that can be completed by the next release,
– programmers choose tasks, and work in pairs,
– programmers write unit tests and add features to pass unit tests,
– programmers fix features/tests as necessary, until all tests are passed,
– programmers integrate code and produce a released version,
– customer runs acceptance tests and correct version is transported into production,
– programmers update effort estimation based on the experience in past release cycle.

2.2. SCRUM

The SCRUM methodology has been firstly mentioned by Ken Swaber in 1995 [Paasivaara, Lassenius 2010]. It was used before the announcement of the Agile Manifesto. Because of the same underlying concepts and rules it has been later integrated in the agile methodology. Scrum is an agile, lightweight process to manage and control software development work. The development team determine themselves the tasks for the next development iteration. The high team iteration stands over intensive documentation. The team-based approach of iteratively and incrementally development aims to solve the issue of rapidly changing requirements. The main elements of SCRUM are the product backlog, the sprints and roles in the Process [Paasivaara, Lassenius 2010; Hunt 2006, pp. 25–30].

The customers are involved in defining the product roadmap and requirements by integrating their input and feedback into a prioritized repository. This is the list of all product features and enhancements of the future application. The development team brings the features of the product backlog into the sprint backlog for each sprint. A sprint is usually a short cycle of two to four weeks. The features developed in a sprint can be tested after the sprint from the customer. This implies that the requirements in a sprint have to be chosen in units that generate working software.

All team members are at the same level, so there is no project manager who decides in case of conflicts. The team works autonomously during a sprint. The SCRUM Master is a role of one team member who is in charge of updating the product and sprint backlog. During very short daily SCRUM meetings, the developers report about their progress, their planned work for the day, about problems and they update the sprint backlog. The progress and the burn-down list can be monitored on a daily basis. Effort estimations are not easy in the SCRUM approach, as requirements are not defined in detail at the beginning of the development of a feature. All open tasks of the sprint backlog at the end of a sprint will be shifted to the next sprint.

2.3. Excerpt of further Agile Software Development Methods

Crystal: The crystal family of lightweight software development life cycle methodologies was created by Alistair Cockburn. Crystal is a human-powered and adaptive, ultralight, shrink-to-fit software management & development methodology. Human-powered means that the focus is on achieving project success through enhancing the work of the people involved. The characterization ultralight means that for whatever the project size and priorities, a Crystal-family methodology for the project will work to reduce the paperwork, overhead and bureaucracy to the least that is practical for the parameters of that project. Shrink-to-fit means that a project starts with possibly small enough features, and work to make it smaller and better fitting. Crystal is non-jealous, meaning that the Crystal methodology permits substitution of similar elements from other methodologies [Cockburn 2002].

Dynamic Systems Development Method (DSDM) is a lightweight software methodology which has its origins in the U.K. In traditional approaches the focus has been on satisfying the contents of a requirements document and conforming to previous deliverables, even though the requirements are often inaccurate. The previous deliverables may be flawed and the business needs may have changed since the start of the project. In addition, time and resources are often allowed to vary during development. In DSDM, the exact opposite is true, time is fixed for the life of a project, and resources are fixed as far as possible. This means that the requirements that will be satisfied are allowed to change. DSDM has underlying principles that include active user interaction, frequent deliveries, empowered teams, testing throughout the cycle. Like other agile methods they use short time boxed cycles of between two and six weeks. The emphasis is on high quality and adaptiveness towards changing requirements [Fowler, Highsmith 2001].

Whitewater Interactive System Development with Object Models (WISDOM) addresses the needs of small development teams who are required to build and maintain the highest quality interactive systems [Nunes, Cunha 2000]. The Wisdom methodology has the following three key components [Wolak 2001]:

1) software process based on user-centered, evolutionary, and rapid-prototyping;

2) set of conceptual notations for modeling of functional and nonfunctional components;

3) project management philosophy based on tool usage standards and open documentation.

Rapid Contextual Design (RCD). Customer-centred or user-centred methods encompass a broad class of techniques to define systems by creating an in-depth understanding of the customers needs. RCD has the following particularities [Beyer et al. 2004]:

- separation of design from engineering,
- making the user the expert,
- keeping up-front planning to a minimum,
- working in quick iterations,
- building a new process (→ Set project focus → Contextual inquiry with potential customers → Build an affinity showing the scope of issues from all customers → Introduce the larger team → Identify issues → Build user stories → Run planning games → Design detailed user interfaces → Test UI with users → Deliver to development → Continue iterations in parallel).

2.4. The Agile Manifesto

In 2001 seventeen software development methodologists held a meeting to discuss the future trends in the software development. In consequence to this meeting, the Agile Alliance emerged with the Agile Manifesto [Highsmith 2009; Ambler 2007; Fowler, Highsmith 2001]. The Agile Manifesto has following twelve principles:

1. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity – the art of maximizing the amount of work not done – is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The key features of the Agile Manifesto are that individuals and interactions stand over processes and tools. The outcome of working software stands over comprehensive documentation and customer collaboration stands over contract negotiation. The response to changes is here more important than following a plan [Highsmith 2009; Cockburn 2002]. Common for all described agile methods is the iterative development of rapidly working software features in self-organizing teams with the focus on interaction and communication.

3. Requirements of software development in health care

The health care sector with the need of integrated systems for health care institutions, like hospitals, has a much higher complexity compared to other industries. This chapter describes the reasons for the complexity that have to be evaluated and integrated when developing software systems in e.g. hospitals. This description leads to the requirements of software development in health care.

The systems in health care became more and more complex and the degree of usability for medical doctors and clinical staff is decreasing continuously. The complexity is based on the integration of heterogeneous systems ranging from computerized patient records over dosage planning applications to financial reimbursement applications. The following reasons make it very complex to develop software in health care institutions:

– Heterogeneous character of processes and systems. Health care organizations have not only one integrated system. In most cases hospital information systems are based on heterogeneous landscapes. Hospital information systems are built as an assembly of three different system types: clinical information systems, hospital management systems and central systems [Homayounfar, Owoc 2011b]. The clinical information systems consist of systems for clinical order management, radiology, laboratory, medical findings, diagnosis, surgical planning and documentation, treatment and care, anesthesiology, medical controlling, and knowledge data base. The hospital information systems consist of following underlying systems for: patient management, administrative order management, invoice management general ledger accounting, controlling, material management, statistical reporting, human resource management, and facility management. Further systems are central systems for computerized patient records, op-

erational systems, authorization management, archiving, master data management, physical IT management, communication systems, and general reporting systems. By only naming these parts of hospital information systems it becomes obvious how complex this domain is. All components have to be interlinked in a framework to be able to communicate with each other. Software manufacturers have developed different parts of hospital information systems as described above, but there is not one system that covers all system elements sufficiently. This is barely possible, as e.g. the system for medical findings can become very complex itself.

– Autonomously working individuals in different departments using the systems with different needs. Every medical doctor has different ways to write reports and to describe the disease and the treatment plan of a patient. If two different doctors wrote a report for a single patient, the probability of having two different reports would be very likely based on the experience and the economical pressure of the health care institution and the available systems.
– Increasing complexity in classification of diseases and treatments due to official reimbursement rules.
– The increased functionalities offered by the new generation of IT systems are handled differently from department to department in health care organizations. Many configuration options and operating modes are supported by computer based systems. Software manufacturers often offer more and more software features to retain market position. These factors create usability problems that have had a direct impact on patient outcomes as well as a number of indirect effects like the costs of replacing and upgrading inadequate computer systems.
– The definitions of many different standards for interface communication in health care have caused a higher complexity. E.g. the Healthcare Information Systems Architecture (HISA) with its layer structure, as middleware of common services that should exist in a hospital information system, Health Level Seven (HL7), Distributed Healthcare Environment (DHE), Standardization of Communication between Information Systems in Physician Offices and Hospitals using XML (SCIPHOX).

Based on the heterogeneous and complex situation in health care organizations the requirements can be summarized in the following list:

– With the growing complexity it becomes more important to choose in the beginning of a project the best fitting method and approach for software development.
– Evaluation of existing landscape to choose the optimal systems portfolio for the future. Health care organizations do not start from scratch in a software development project. Many system elements already exist and have to be integrated in the new system complex. The requirement is to integrate not only the interfaces, but go a step further and integrate the data processes.

- Linking together all available information from all different involved internal and external parties in a health care organization:
  - Internal: Medical doctors, laboratory doctors, radiologists, students, nurses, care personnel, technicians for devices, computer scientists and software developers, hospital directors, financial administrators, controllers…
  - External: Other interlinked hospitals, government officials (for statistics), legal reimbursement departments, insurances, external laboratories…
- Current workflow systems do not provide an integration environment. The different involved parties of functional groups in healthcare organizations require different types of applications to support their needs and processes. For example, integrating Picture Archiving and Communication Systems (PACS) with hospital or radiology information systems will allow radiologists to be presented with collateral patient information. This allows health care organizations the development and integration of patient history, clinical information, symptoms, and the previous examination history to be presented to the physician along with images retrieved from the PACS, which is highly aiding in the interpretation of images.
- Patient-oriented approach with considerations of data security.
- Integration of information at different level like business level, medical level.
- Holistic approach of analyzing the needs of all internal as well as external participants in the health care institution. Besides the medical knowledge domain that has to be integrated in software development, it is also necessary to integrate the particularities of the organizational structure of the hospital and the processes of the hospital. The optimum would be to analyze and to amend organizational structure and processes according to the overall strategy of the hospital. Afterwards the software development can be set up on a clean structure, which influences also the need for and the content of the interfaces between integrated department systems.
- Individualization of monitoring for different stakeholders. Monitoring is not a trivial task, as it is also here necessary to evaluate the participants who are receiving the information that have to be monitored. After the definition of the participants it is necessary to evaluate the questions that have to be answered by monitoring specific values. Those questions are as different as the participants in the process. For example, the medical doctors are interested in monitoring the disease progress and the trend for one single patient or on the opposite site the evaluation of characteristics of given values leading to a special disease, e.g. the evaluations of data of all available patients for analyzing risk factors to predict the probability of liver cancer. The director of the hospital has the interest to monitor and evaluate process efficiency or financial values to control the strategy of the hospital. This data mining and knowledge management tasks are also depending on the available underlying data quality [Homayounfar, Owoc 2011a, b].
- Performance of application with a very high rate of stored data.

4. Description of limitations of agile methods in health care

The limitations of agile methods for software development in health care are the gap of provided content of agile methods and the requirements of software development in health care. This chapter will combine the elements of the previous chapters and describe the limitations of agile software development methods in health care.

Agile methods do not handle large teams well. The approach only works for small and medium-sized teams. They are useful in many smaller projects for developing different functionalities, but it is not per se possible to handle the previously described complexity in health care. The agile method is focused on the development of working code, this approach cannot cover all other surrounding factors that are important to be included into the approach to develop holistically valid software. The processes, the strategy, the technical constraints, the distribution of information would not be an influencing part of the software development. Hence, the outcome would not be integrated into the health care organization. The software would solve some problems and help some parties involved in the health process, but the important requirement for a holistic and integrated health care solution would be still missing.

Together with the complexity and in combination with the many different involved parties the team size grows immanently. The structure of the project can surely be divided in sub-teams, but this implies higher communication efforts. Communication is a key success factor in software development in general, but communication is most efficient if communication happens face-to-face. The requirement of having the whole team, including the customer and all involved parties at one location is not realistic in practice of health care. In health care software development it often happens because of the large size of the projects in globally distributed teams with special domain knowledge. Agile methods are only useable in small development projects where the team is located at the same place. The wide range of involved participants in the health care sector shows that it is in practice not possible to bring together all the parties in one team and in one location. Even by using video conferencing systems there is a huge loss in efficiency and the risk of losing important information. This is even worse if using the agile approach without sufficient documentation.

The aim of the agile approach is to face the problem that customer requirements are changing often, especially after the software is finalized. This is in the nature of human beings and the agile method fails, particularly in health care, as the project would never end if the requirements and the system design are not accepted, frozen, and released from the customer at a certain point of time. Furthermore, the project can easily get taken off track if the customer is not sure what final outcome he wants. The risk of development of software with the view of developers and not including the domain knowledge of the health care institution is very high. It is likely with the

agile approach to have a nice piece of software which does not meet the requirements and the expectations of the customer.

Due to the integrative approach of agile methods, it is hard to understand exactly where the project stands. In a typical environment, upper management wants to know when each phase, such as design, code, or test, is completed. Thus, due to the various iteration steps, it can be hard to understand if the project is on track. Effort estimation of agile approach fails in health care. In the large health care projects for software development it is necessary to know in advance of the project roughly the required efforts and costs of the project. Setting up a holistic health information system including the processes, the organizational structures and all other required levels, the project can easily take two years to have the first stable version. It is not realistic to work on such a set of systems without an effort and prepared earlier cost plan. Cost planning is not static, it has to be amended continuously, and therefore it is useful to label each plan with a maturity level to show the probability of the planning. Agile methods with the code centric view do not take into account the other relevant components of the holistic health care requirements as processes or organizational requirements.

The general success of using agile methods depends strongly on the best breed of persons in the project team. Agile development requires highly skilled and highly motivated individuals which may not always be available. Agile places a premium on having premium people. Not every team can be motivated, experienced, and skilled enough to work using self-organized approachesand to come up with lightweight processes, and collaborate seamlessly to achieve the complex requirements in health care. In this environment it is hardly possible for the team at a micro level to choose the best way to reach the desired result, without any close supervision.

5. Conclusions

ASDM have evolved in the last decades to provide a solution for a fast changing requirement and to provide a light way of handling software projects. ASDM have many advantages and legitimation in particular industries. In health care the limitations show that ASDM are not practicable due to the complexity, the missing holisticity of ASDM, necessity of the large and often globally distributed team structures, and the need for extensive documentation.

Other approaches also do not cover all requirements of software development in health care. Therefore, the conclusion is that ASDM is not applicable everywhere but there is a need for a new holistic approach for development of software in health care. Some of the features provided by ASDM can be used in the new methodology at a micro level.

For health care institutions like hospitals it is necessary to see the software development not as the core tool. In IT and for the software developers a change of thinking is necessary to see IT and software development as supporting tool and not

as the core process in an industry. This would help to develop a holistic method that combines the knowledge of different domains, e.g. medical doctors, nurses, radiologists, and software developers. A specific level of structure and documentation is necessary in the new methodology. But also the advantages of ASDM in terms of agility and rapid development of working software could be integrated. Also the flexibility of the approach could be inherited. The new methodology has to be adaptable to each situation and to the customer's requirements to prevent the methodology from becoming a dogma with the risk of being used in other areas where it would not fit in – like ASDM in health care.

References

Ambler S.W. (2007), Agile software development at scale, [in:] B. Meyer, J.R. Nawrocki (Eds.), *Balancing Agility and Formalism in Software Engineering*, Springer, pp. 1–12.

Beck K. (1999), Embracing change with extreme programming, *IEEE Computer*, Vol. 32, pp.70–77.

Beyer H., Holtzblatt K., Baker L. (2004), An agile customer-centered method: rapid contextual design, [in:] C. Zannier, H. Erdogmus, L Lindstrum. (Eds.), *Extreme Programming and Agile Methods – XP/Agile Universe 2004*, Springer, Berlin Heidelberg, pp. 105–116.

Cockburn A. (2002), *Agile Software Development*, Addison-Wesley, Boston.

Collyer S. (2009), Project management approaches for dynamic environments, *International Journal of Project Management*, Vol. 27, No. 4, pp. 355–364.

Fowler M., Highsmith J. (2001), *The Agile Manifesto*, http://www.agilemanifesto.org (accessed 15.06.2011).

Highsmith J. (2009), *Agile Project Management: Creating Innovative Products*, Addison-Wesley, Boston.

Homayounfar P, Owoc M.L. (2011a) *Data Mining Research Trends in Computerized Patient Records*. Publication in Preparation for FedCSIS in Szczecin: 1st International Workshop on Interoperable Healthcare Systems (HIS'2011).

Homayounfar P, Owoc M.L. (2011b), *Data Mining Methods as an Essential Part of Hospital Information Systems*, publication in preparation for 2011 Conference "Knowledge Technologies in Public Management" in Katowice.

Hunt J. (2006), *Agile Software Construction*, Springer, London.

Nunes, N., Cunha, J. (2000), Wisdom: A software engineering method for small software development companies, *IEEE Software*, September/October, pp. 113–119.

Paasivaara M., Lassenius C. (2010), Using Scrum practices in GSD projects, [in:] D. Šmite, N.B. Moe, P.J. Ågerfalk (Eds.), *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*, Springer, Heidelberg, pp. 259–278.

Wolak C.M. (2001), *Extreme Programming (XP) uncovered*, http://www.scisstudyguides.addr.com/papers/cwdiss725paper4.pdf (accessed 14.06.2011).

OGRANICZENIA METOD *AGILE* TWORZENIA OPROGRAMOWANIA W SEKTORZE ZDROWIA

Streszczenie: W ostatnich latach metody tworzenia oprogramowania Agile (ASDM) stały się bardzo popularne i są stosowane w różnych dziedzinach gospodarki. W każdej z nowych metod stajemy w obliczu nowych sytuacji związanych z rozwojem oprogramowania czy zarządzania projektem. Każdy projekt powinien być traktowany w sposób indywidualny ze względu na zróżnicowanie warunków jego realizacji w odmiennych obszarach zastosowań. Podejście Agile (zwinne) może być korzystne, jeśli zastosujemy je w odpowiedniej sytuacji, ale może być nawet szkodliwe, jeśli będzie stosowane w nieodpowiednim środowisku. Szczególnie sektor zdrowia ma swoje specyficzne uwarunkowania, które prowadzą do pewnych ograniczeń w stosowaniu metod ASDM. Celem artykułu jest pokazanie ogólnych ograniczeń metod, a także tych, które odnoszą się do sektora zdrowia.