ISSN 2518-7325 (Online), ISSN 2306-1677 (Print)
Land Management, Cadastre and Land Monitoring
Received: 27.04.2026; Accepted: 25.05.2026; Published: 30.06.2026;
http://dx.doi.org/10.31548/zemleustriy2026.02.02

UDC 004.9:911.5/.9:528.94

STANDARDIZATION OF MSF AGILE METHODOLOGY WITH THE USE OF ISO/IEC 24744

V. Chabaniuk, *Ph.D. in Physics and Mathematics,*
Senior Researcher
E-mail: chab3@i.ua
ORCID: 0000-0002-4731-7895
Institute of Geography of the National Academy of Sciences of Ukraine
O. Dyshlyk, *Executive Director,*
E-mail: dyshlyk@geomatica.kiev.ua
ORCID: 0000-0002-8066-3925
"Geomatic solutions" LLC

Annotation

The paper proposes the standardization of the software development methodology MSF Agile. It made by using the metamodel of software development methodologies from the ISO/IEC 24744 standard. Original of MSF Agile named MSF for Agile Software Development and denoted MSF4ASD. MSF4ASD implemented in MSF version 4.0 in 2005 p. in the tool/technology Visual Studio Team System. Therefore, here it is called an example of a specific software development methodology. Without implementation, the methodology exists, but it is better to call a specified generalized MSF methodology with the threat of losing practicality.

Given the topicality of the Agile methodologies, MSF Agile can be a practically useful implementation of the generalized MSF methodology. The specific MSF Agile

methodology can be obtained in two ways: 1) two-step specification or 2) standardization with subsequent specification. This article chooses the second way - first, MSF Agile standardized using the development methodologies metamodel from the ISO/IEC 24744 standard. After that, it is easier to perform practically useful specification, since the task becomes typical. Implementation will be possible using various information technologies (IT), including Microsoft IT. The specific methodology can already be directly used in practice.

In the three main sections of this paper: 1) a standardization tool is introduced – the necessary elements of the software development methodologies metamodel from the ISO/IEC 24744 standard are described; 2) information about MSF4ASD is reminded, here simplified to MSF Agile; 3) MSF Agile is presented using the elements of the ISO/IEC 24744 metamodel. This shows how to make MSF Agile a standardized methodology. Then, by one-step reduction, it is possible to obtain a specific methodology that can constructively satisfy the Framework approach and facilitate the transition to Pattern-Based Spatial Engineering.

Keywords: *Microsoft Solutions Framework (MSF) Agile methodology, ISO/IEC 24744 MSF Agile standardization*

Introduction

Topicality

In article [1] the methodology named close by sense, but more practical comparing with approach. To explain the meaning of practicality, a second interpretation of the Framework Approach - as a generalized methodology - is used here. It explains how to perform a two-step reduction from a generalized to a specific MSF methodology. As an example considered the MSF Agile methodology. It originally in 2005 called MSF for Agile Software Development and denoted MSF4ASD. The result of the first reduction step can be called a specified generalized MSF Agile methodology. The second reduction step allows to obtain a specific MSF Agile methodology. And specific methodology can already be directly used in practice.

This paper investigates the class of “Methodologies of the Framework Approach to Research and Design of SpIS (Spatial Information Systems)”. Currently, several instances of this class are being created in parallel. Earlier in our works, one such instance was already mentioned - the methodology of SpIS extension. Another instance is the MSF Agile methodology. It is important to demonstrate the application of the metamodel of software development methodologies from the ISO/IEC 24744 standard to MSF Agile. Thanks to this example, we can talk about the properties of all instances of this class of methodologies. And with them - about the methodology of Pattern-Based Spatial Engineering (PBSpE). The future methodology of PBSpE will be part of Model-Based Systems Engineering [4]. It will include Model-Based Software Engineering in the interpretation of [5].

Analysis of recent research and publications

Elements of both MSF and Agile methodologies described in many sources. To better understand MSF Agile, we have dealt separately with the MSF and Agile software development methodologies. Thus, the MSF 2.0 methodology described in the monograph [6], the MSF 4.0 methodology - in the monograph [7]. In the article [2], the MSF methodology is called a generalized or meta-methodology. Its specification in 2005 achieved using the current version of Visual Studio Team System. Included into MSF 4.0 MSF for Agile Software Development denoted MSF4ASD. Without this implementation, it can be called *specified generalized* methodology. *Generalized* - due to belonging to MSF 4.0. *Specified* - due to the fact that today there is only their description with possibility of several implementations, and with them – specifications (as a process). Real example of a description of MSF Agile technology is [8], where described the 2005 version of MSF4ASD. To make the MSF4ASD technology relevant today, a modern implementation needed.

Agile software development methodologies are even more difficult to understand than MSF. Many authors are very loose with terminology and call Agile methodologies Agile methodics and methods. Questions arise immediately when trying to understand what a “software development methodology” is. For example, let

us remind the translation of the article “Methodologies, Techniques, Methods, Frameworks – What’s the Difference?”. It is included in the article [9] as an example the viewpoint of an IT specialist. The example does not provide clarity of understanding. After all, its author suggests understanding the terms listed in the question depending on the context. But the context may be different each time.

From the cited source, it follows that the first problem is the instability of terminology regarding the Agile methodology, which is different for different authors. The ambiguity also applies to the “possible” components of the methodology. Instead, we call Agile a methodics that is justified by a reference to a classic monograph [10]. There “Agile” united several Agile methods. In 2025, the subjects of these concepts were called components of the echelons of entities of the Framework approach to handling SpIS (including their creation). “Methodics” was “between” the process entities “methodology” and “method”. According to this hierarchy, methods can “enter” the methodics and only through it – into the methodology.

For an initial understanding of the “Agile” phenomenon in the context of software development methodologies, we recommend the source [11]. It is not scientific, but it is a relevant introduction to the problems of Agile, covering most practical issues. After it, it is possible to move on to more serious publications. For example, to monographs [10] or [12]; the latter has been translated into Russian. This article directly uses a simple explanatory article [13], the Agile manifesto [16], and the archive from [8].

We already have experience in applying the “practice” of creating a complex hierarchical SpIS. Therefore, we can describe both the practice itself (answers the question what?) and the process (answers the question how?). De facto, they clearly exist. And if both practice and process exist, then there must be a methodology. We believe that by the beginning of 2026 it only needs to be further defined and described. This is enough to use the term “approach” with justification and call it constructive (like the strategy).

At the moment, our approach can also be called the “Framework approach of extension (of Atlas and Geographic Information Systems)”. It is based on framework methods from two groups, each homogeneous with respect to (a variant of) the subject: 1) Conceptual Frameworks X, 2) Solutions Frameworks X[Y], where square brackets [Y] report that Y may not exist. Although the approach itself cannot be named a set of homogeneous methods, since in the first case the subject is X, and in the second – X[Y]. In general, framework are solutions obtained by applying one of the two or both frameworks (and an arbitrary number of times). The concept of X has been explained several times before. The concept of X[Y] is divided into two concepts with variable values. Y, if present, shows a dependence on the value of the (hierarchical) stratum, therefore the X[Y] SoFr can be named as follows: [General (γ) | Conceptual (β) | Application (α) | Operational (ω)] Framework of Stratum Y= γ, β, α, ω, Solutions of Subject X.

It should be noted that the results of this article show the ambiguity of the future methodology of the Framework Approach to the handling of SpIS. It is necessary to take into account at least three viewpoints: the two-step reduction (specialization/specification/concretization) of the Microsoft MSF meta-methodology, "our" extension methodology and the "direct" application of Model-Based Software Engineering (MBSE), which is described in the monograph [5].

Since the beginning of the century, we have used instances of both Frameworks (CoFr and SoFr) in many projects for the creation or exploitation of objects X and/or XY. In fact, immediately after the discovery of the first SoFr, generalizations began, which can be combined into two directions, homogeneous according to some criterion: 1) subject-oriented, or 2) process-oriented.

In order not to use the term “approach” to MSF, we use the term “meta-methodology”. Both terms are applicable to MSF, but the second term allows us to keep the focus on the methodology. Remind that in the article [1] three viewpoints (interpretations) on the Framework approach were mentioned. This article is devoted to the development of the 2nd viewpoint on the Framework approach, but we deliberately narrow the context to the MSF meta-methodology.

Purpose

The purpose of the work is to standardize the MSF Agile software development methodology using the ISO/IEC 24744 standard [3]. It should complement the previously introduced concept of a “generalized methodology for handling SpIS” and will be useful in creating a Framework Approach methodology for the research and/or design of SpIS from the view point of its use in the Pattern-Based Spatial Engineering (PBSpE) we are creating.

Materials and research methods

This work uses the methods of Conceptual Frameworks (CoFr), Solutions Frameworks (SoFr) and Metamodel of development methodologies from the ISO/IEC 24744 standard [3]. The methods are called systemic because of their applicability to the study and/or design of the Spatial Information Systems (SpIS). SpIS in this article is understood very broadly. Classical and non-classical SpIS are distinguished. Well-known examples of classical SpIS are Electronic Atlases such as the Electronic version of the National Atlas of Ukraine (ElNAU). Examples of non-classical SpIS are Spatial Information Activity Systems (SpIAS) and Atlas GeoInformation Systems (AGIS), which have already been studied in our works.

Metamodels are useful for defining the concepts, rules, and relations used to define methodologies. Although a methodology can be described without an explicit metamodel, formalizing the core ideas of a given methodology is valuable when checking its consistency or when planning extensions or modifications. A good metamodel should consider the process to be followed, the work products to be created, and those responsible for making it all happen. In turn, defining the work products to be developed involves defining the basic modeling building blocks from which they are built.

Metamodels are often used by methodological engineers to build or modify methodologies. In turn, methodologies are used by developers to create products or provide services within the context of a particular endeavour.

Metamodel, *methodology* and *endeavor* (originally “endeavour”, we also translate this term as “interversion”) in the standard [3] refer to three different areas of expertise, which at the same time correspond to three different levels of abstraction and three different sets of fundamental concepts. An alternative translation of the “endeavour domain” could be “intervention domain”, if we use the monograph [14] and its “metamodeling”. Since the work done by developers at the endeavour level is constrained and guided by the methodology used, the work done by the methodological engineer at the methodology level is constrained and guided by the chosen metamodel. Traditionally, these relations between "modeling layers" (in our case, strata or echelons), which are called "domains" here, are viewed as "*instance of something*" relations, in which elements in one layer (strata or echelon) or domain are instances of some element in the layer (strata/echelon) or domain below.

We placed the above on the general scheme of using γ | β | α | ωKaRi and obtained Fig. 1. Three domains [3; Figure 1] are shown on the left of Fig. 1.

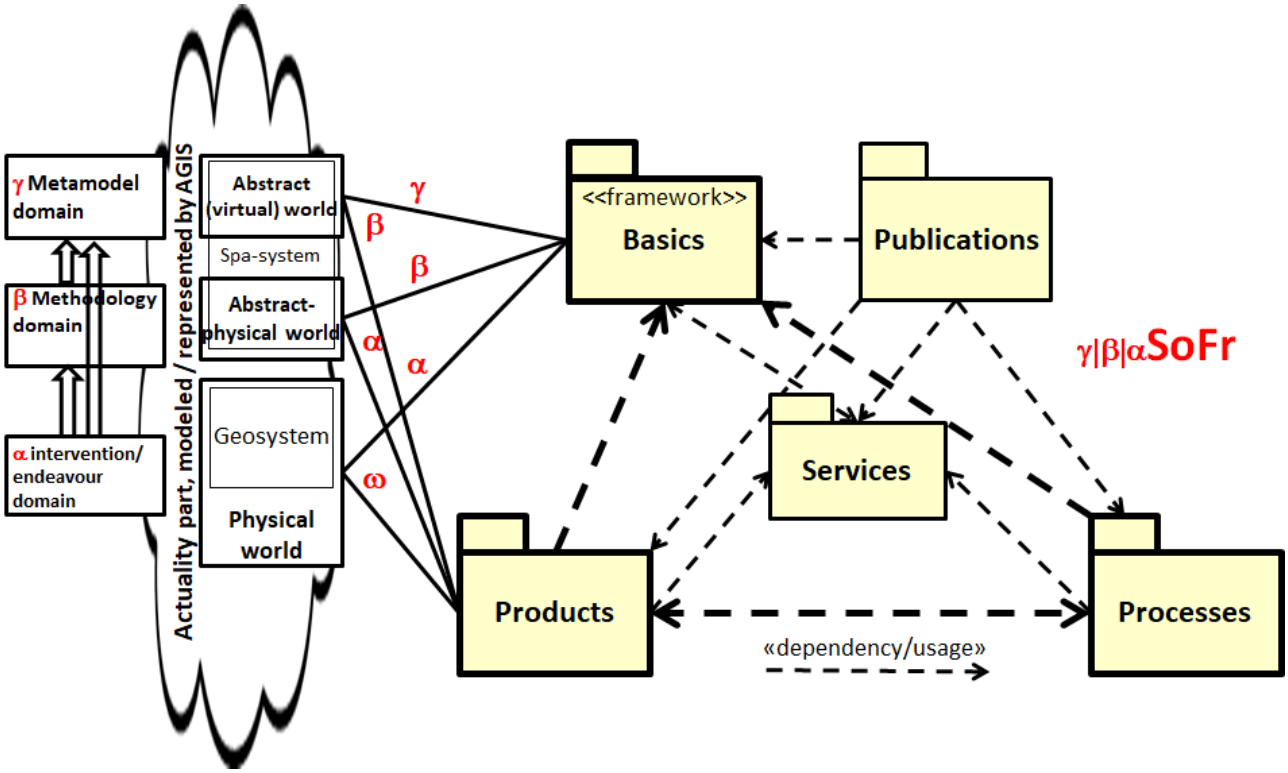

Fig. 1. [3; Figure 1] added to the general scheme of γ | β | α | ωSoFr use

The [3; Figure 1] shown on the left of Fig. 1 was called “*The three areas of expertise, or domains, which act as a context for SEMDM. Arrows mean "is*

represented by". The standard itself explains Fig. 2. In the original it was [3; Figure 4], whose name we did not change.

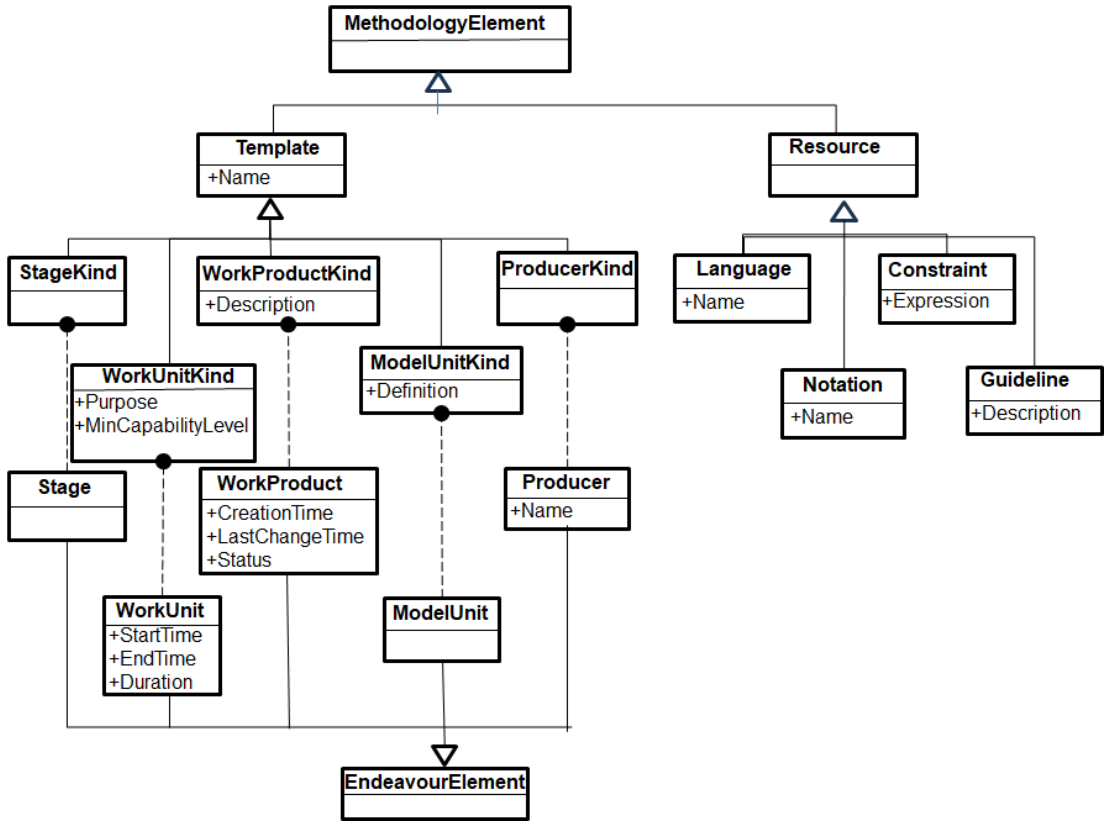

Fig. 2. Abstract view of SEMDM showing the main classes in the metamodel from [3; Figure 4]

To show the correspondence of the SoFr model structures with the classes of the ISO/IEC 24744 metamodel, we used the diagram from [15]. On it, we showed the correspondence with the SoFr Processes and Products packages and obtained Fig. 3, without detailing the comtent of the specified packages. Here we note that we never forgot about SoFr users. For example, the description of the very first ProSF (Project Solutions Framework) SoFr contained the purpose of the packages: 1) Publications package was intended for communication with the external environment of the project; 2) Products package was intended for developers of project products; 3) Processes package was intended for project management; 4) Services package was intended for the administrator, documenter and/or user training specialist; 5) Basics package was intended (element by element) for all project participants. It was supported by the architect and project management.

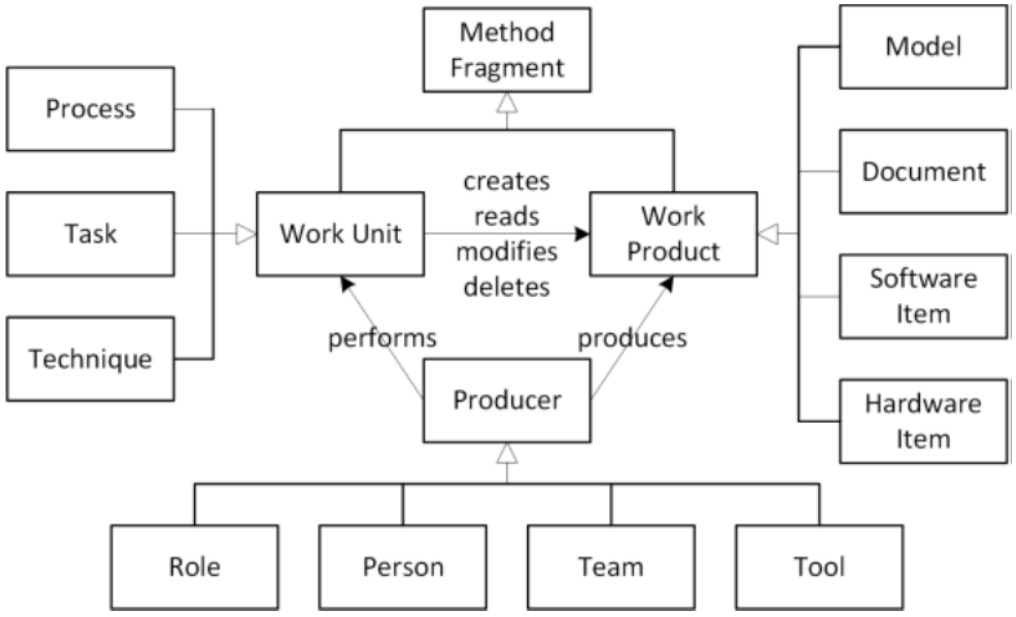

Fig. 3. Partial conceptual diagram [3] [15] and its correspondence to the Processes and Products packages

WorkUnit (Work Item below) is a task to be performed, a process to be carried out, or a technique (as a process) to be used within a project. WorkProduct is an intermediate or final result created during a project, which can be a document, a model, etc. Producer is responsible for performing the WorkUnits and can be represented by a role (a set of responsibilities of the Producer), a tool (a means that enables producers to perform their responsibilities), and a team (an organized group of producers who share a common purpose).

If we remind the correspondence with the SoFr again, it should be noted that in the last few articles we have been considering the Framework Approach to the handling of the SpIS, in which the content of the so-called “echelons” of SpIS users is essential. The echelons include the so-called “process” entities such as “methodology”, “methodics”, “method”, etc. Without going into details, we note the following main statements: 1) the echelons of users include the Producers from Fig. 3; 2) the echelons and their entities correspond to the strata and components of a particular SpIS; 3) when constructing the components of the strata of a particular SpIS, the corresponding SoFr ($\gamma \mid \beta \mid \alpha \mid \omega$) is used.

Uniting Process and Product (5.4 Uniting Process and Product)

When using SoFr models, one should never forget about their important characteristic – process-product dualism. The standard [3] has a separate subsection 5.4, considering this dualism.

Namely, most existing approaches to metamodeling focus either on the process or on the modeling (i.e. product) side of methodologies. However, most of these approaches offer connection points to “plug in” an additional, as yet undefined, component of a full-fledged methodology. SEMDM (Software engineering — Metamodel for development methodologies) goes further, proposing a complete metamodel that equally covers the process and modeling aspects of methodologies. Not doing so would be like trying to define the actions to be performed without defining the concepts on which these actions should act (process focus), or the concepts to be used without knowing what to do with them (modeling focus). This approach has the advantage of allowing for a deep definition at the methodology level of the interactions between the process and the products generated by it.

Finally, the Resource in Fig. 2 is specified (specialized) by a Language (a structure of model unit types that focuses on a particular modeling perspective), a Notation (a specific syntax, usually graphical, that can be used to represent models created using certain languages), a Guideline (a specification of how some elements of the methodology can be used), and a Constraint (a condition that is met or must be met at a particular point in time).

Research results - MSF for Agile Software Development

The section takes into account the monographs [6], [7], archive from [8], article [13], and the Agile Manifesto[16].

What is MSF Agile?

MSF Agile is a software development methodology created by Microsoft to help teams manage software projects using an agile approach. It provides a framework of workflows, roles, and key activities to help teams move through the various stages of the software development life cycle. The full name of the

phenomenon discussed in this article is MSF for Agile Software Development (MSF4ASD), which, together with the phenomenon MSF for CMMI Process Improvement (MSF4CMMI), was "included" in MSF version 4.0 in 2005.

For a more complete introduction to MSF4ASD, we recommend reading the materials in the reference [8]. There in the file ReadMe.txt recommended go to the Process Guidance directory and open ProcessGuidance.html (Fig. 4). Known limitations: 1) The Project Portal link does not work when the Process Guidance is used offline. 2) The “getting started” tasks will not be displayed in the offline version. 3) The mpp and xls work products are not automatically populated if they are not used with Team Foundation Server.

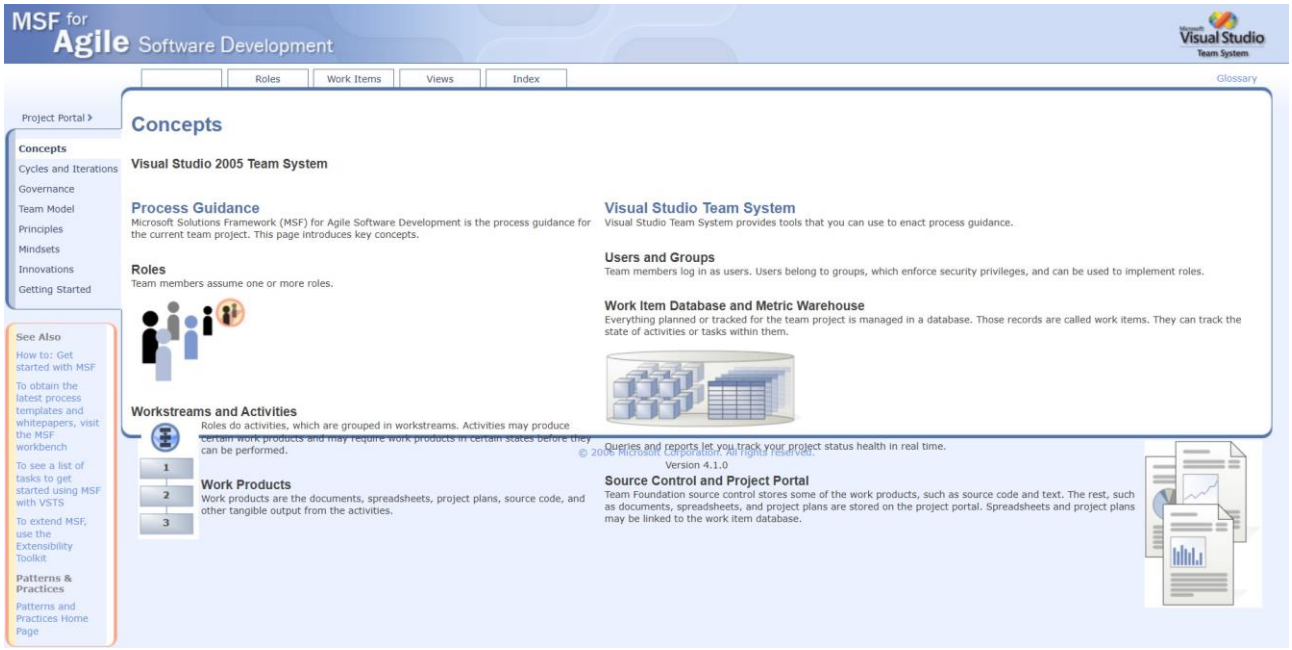

Fig. 4. ProcessGuidance.html page [8]

Immediately after opening the ProcessGuidance.html page (Fig. 4), we see the definition and access to the main Concepts of MSF Agile in Visual Studio 2005 Team System realization. The Process Guidance section (left part of page) states even that Microsoft Solutions Framework (MSF) for Agile Software Development is the process guidance for the current team project.

In Visual Studio Team System section (right part of page) said Visual Studio Team System provides tools that you can use to enact process guidance.

Section includes:

Users and Groups

Team members log in as users. Users belong to groups, which enforce security privileges, and can be used to implement roles.

Work Item Database and Metric Warehouse

Everything planned or tracked for the team project is managed in a database. Those records are called work items. They can track the state of activities or tasks within them.

Source Control and Project Portal

Team Foundation source control stores some of the work products, such as source code and text. The rest, such as documents, spreadsheets, and project plans are stored on the project portal. Spreadsheets and project plans may be linked to the work item database.

In addition to the Process Guidance directory, the archive contains template directories (examples are given in brackets ()): Project Management (Development Project Plan.mpp, Project Checklist.xls), Requirements (Scenario Description.doc), Security (Template Sample - Web Application Threat Model.doc), Test (Test Approach.doc). We cannot pay more attention to these templates. In the future, we will use only the elements of Fig. 4.

MSF Agile takes into account the main ideas of Agile development. They formulated according to the values of Agile Manifesto for Software Development and its principles [16]. Where appropriate, we will also use the elements, access to which shown in Fig. 4.

Core values, principles, mindset and leadership of Agile development

Agile Manifesto Values and Principles

The values of the famous Agile Manifesto [16] are Individuals and interactions, Working software, Customer collaboration, and Responding to change. There are twelve principles in total, but they often change slightly. Thus, in [13] seven principles are formulated: Collaboration with customers, Foster open communication, Work towards a shared vision, Quality is everyone’s daily work, Stay flexible, Adapt

to change; Make deployment a habit, and Flow of values. In [8] nine principles formulated. Access to them shown in Fig. 4 in the left menu (see Principles). To convey the essence of virtually all twelve principles, we present the first paragraph and the headings of the MSF for Agile Software Development principles.

MSF for Agile Software Development is a scenario-driven, context-based agile software development process for building .NET and other object-oriented applications. It directly incorporates practices for handling quality of service requirements, such as performance and security. It is also context-based and uses a context-driven approach to defining how to work on a project. This approach helps create an adaptive process that overcomes the limitations of most agile software development processes while achieving the goals defined in the project vision.

The MSF for Agile Software Development principles are: 1) Partner with Customers, 2) Work Toward a Shared Vision, 3) Deliver Incremental Value, 4) Invest in Quality, 5) Empower Team Members, 6) Establish Clear Accountability, 7) Learn from all experiences, 8) Foster open communications, 9) Stay agile, adapt to change.

Mindsets and Governance

Mindset: Quality is determined by the customer, Pride in one's work, Team of colleagues, Frequent deliveries, Willingness to learn, Early clarification of details, Quality of service, Civic stance.

Governance: concerns to the control of time and money relative to the flow of value: Are we doing it right? Can we do it on time and budget? Are we ready for integration? Are we ready for deployment? Do we realize the value?

In addition to values, principles, mindset, and Governance, MSF Agile is defined by the triad of processes, products, and performers (producers). Below is a brief summary of these.

Processes

The MSF process model well described in the monograph [6]. One of the options for version 2.0 shown in Fig. 5. The phases consisted of stages. The models

of subsequent MSF versions were variations of version 2.0. The following is the evidence for MSF Agile.

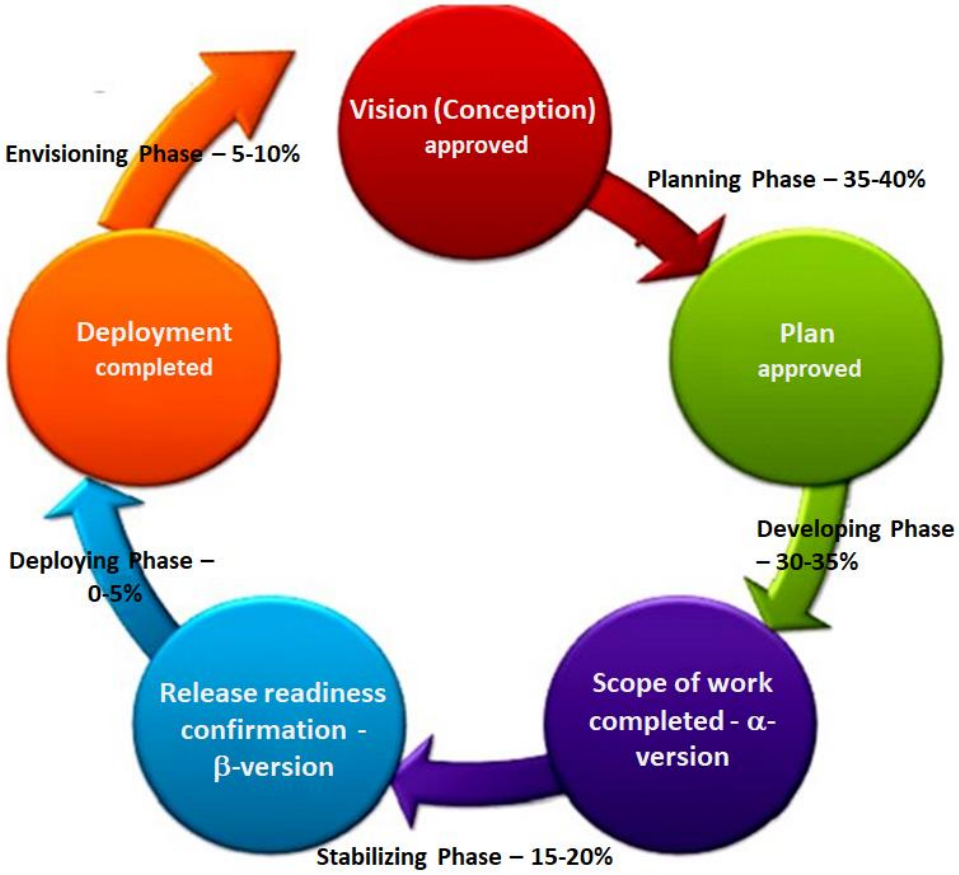

Fig. 5. MSF 2.0 process model

In MSF 4.0 (MSF Agile) process model described through “workstreams”. Namely, according to [8] roles do activities, grouped in workstreams. Activities may produce certain work products and may require work products in certain states before they can be performed. From the further description of the workstreams, it is clear that for MSF 4.0 (MSF Agile) the MSF 2.0 process model, shown in Fig. 5, is valid.

Workstreams

Capture the project vision, Create a service quality requirement, Create a scenario, Manage the project, Plan an iteration, Manage an iteration, Create a solution architecture, Create a product, Fix a bug, Implement development tasks, Close a bug, Test the service quality requirement, Test the scenario, Release the product.

Activities By Workstream

Capture the vision of the project: Write the vision, Identify the characters, Refine the characters.

Creating Quality of Service Requirements: Brainstorming on Quality of Service Requirements, Developing a Lifestyle Overview, Prioritizing the List of Quality of Service Requirements, Writing Quality of Service Requirements, Defining Security Goals.

Script creation: brainstorming scripts, developing a lifestyle overview, prioritizing a list of scripts, writing a script description, creating a script storyboard.

Project Management: Reviewing Goals, Assessing Progress, Assessing Test Metric Thresholds, Sorting Errors, Identifying Risks.

Iteration planning: Determining the iteration duration, Scenario evaluation, Quality of service requirements evaluation, Scenario planning, Quality of service requirements planning, error fix distribution planning, Allocation of scenarios to tasks, Allocation of quality of service requirements to tasks.

Iteration Management: Iteration Monitoring, Risk Mitigation, Retrospectives.

Creating a solution architecture: System decomposition, Interface definition, Threat model development, Performance model development, Architectural prototype creation, Infrastructure architecture creation.

Product build: Start build, Check build, Fix build, Accept build.

Error Fixing: Reproducing the error, Determining the cause of the error, Repurposing the error, Choosing a error fix strategy, Writing code to fix the error, Creating or updating a unit test, Running the unit test, Refactoring the code, Reviewing the code.

Implementing a development task: Determining the cost of a development task, Creating or updating a unit test, Writing code for the development task, Performing code analysis, Running a unit test, Refactoring the code, Testing the code, Integrating code changes.

Close error: Check for corrections, Close error.

Testing a Quality of Service Requirement: Defining a testing approach, Writing performance tests, Writing security tests, Writing stress tests, Writing load tests, Selecting and running a test case, Discovering a error, Conducting exploratory testing.

Scenario Testing: Defining a testing approach, Writing validation tests, Selecting and running a test case, Discovering a error, Conducting exploratory testing.

Product Release: Executing the Release Plan, Reviewing the Release, Creating Release Notes, Deploying the Product.

Disciplines related to the MSF Process Model (Fig. 5): Requirements, Planning, Architecture, Development, Testing.

Cycles and Iterations

The seamless integration of MSF for Agile into Visual Studio Team System supports rapid iterative development with continuous learning and improvement.

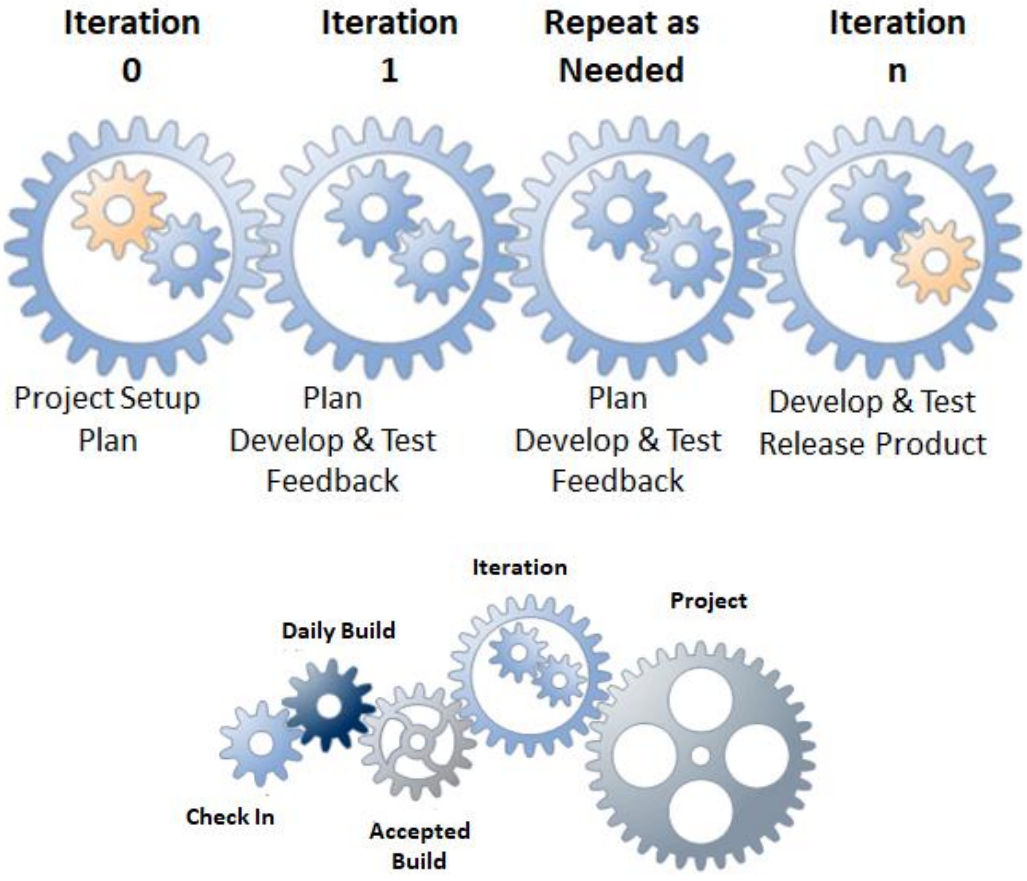

Fig. 6. Cycles and iterations [8]

- Product definition, development, and testing are performed in overlapping iterations and this leads to gradual completion of the project.
- Different iterations have different focuses as the project approaches completion.
- Small iterations allow you to reduce the margin of error in estimates and provide quick feedback on the accuracy of project plans.
- Each iteration should result in a stable part of the entire system.

  Iteration 0: Project Start, Planning

  Iteration 1: Planning, Development and Testing, Feedback

  Repeat as needed: Planning, Development and Testing, Feedback

  Iteration n: Development and Testing, Product Release.

Products

There is a distinction between Work Items and Work Products. The former are related to Processes, the latter to Products. We call this fact “dualism,” which makes the distinction between Processes and Products clearer. It is valid in every context. Here are a few examples.

Work Item

A Work Item is a database record that Visual Studio Team Foundation uses to track the assignment and status of work. The MSF process for agile software development defines five Work Items for assigning and tracking work. A shared Work Item database and Metrics Repository allowing to answer questions about the status of a project in real time.

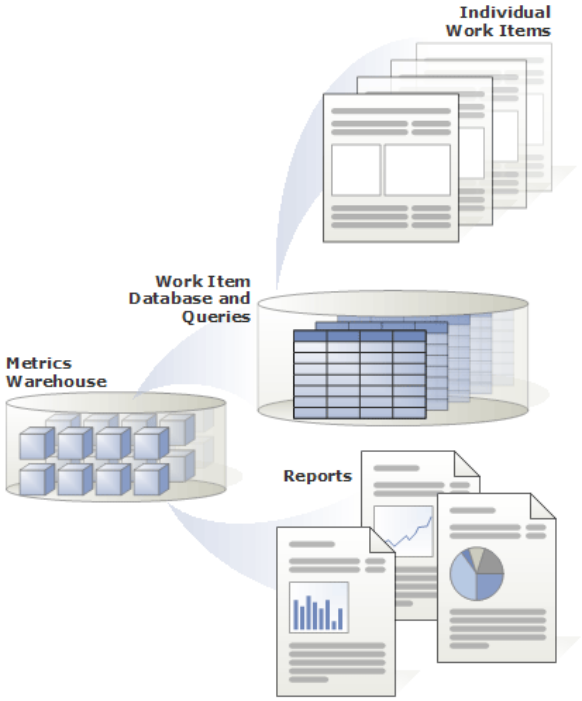

Fig. 7. Work Element [8]

Artifacts (Work Products)

Vision Statement, System Architecture, Test Plan, Personas, Scenarios, Storyboards, Development Tasks, Interface Models, Architectural Prototypes, Unit Tests, Classes, Change Sets, Log Notes, Test Scenarios, Bug Reports.

Work Products according to [8]. Work products are documents, spreadsheets, project plans, source code, and other tangible results of activities.

Performers

The evolution of the MSF Team Model shown on Fig. 8.

MSF Team Model

Fig. 8 – MSF team model: a) version 2.0 [6], b) [8], c) version 4.0 [7]

Support groups: Architecture, Development, Product Management, Program Management, Release Management, Testing, User Training.

Who advocates for what: Architecture – advocates for the System as a whole, Development – advocates for the Technical Solution, Product Management – advocates for the Customer's Business, Program Management – advocates for the Delivery of Solutions, Release Management – advocates for the Uninterrupted Delivery and Deployment of the Solution in the Appropriate Infrastructure, Testing –

advocates for the Quality of the Solution from the Customer's perspective, User Training – advocates for the Most Effective Solution in the eyes of the Target Users.

Team principles

A team of colleagues with clear accountability, shared responsibility, and open communication. Each role is responsible for a certain share of the quality of the overall solution.

Protecting all key stakeholders that need to be represented in a successful software development project. Each point of view is represented to provide checks and balances that prevent errors, omissions, and one-sided decisions.

Stretch to fit the scale required for a particular project. Components can be combined into small teams or further enhanced as teams scale for larger projects.

Who does what?

Business Analyst: Captures the project vision, Creates service quality requirements, Creates scenarios.

Project Manager: Manages the project, Plans iterations, Manages iterations.

Architect: Creates the solution architecture.

Developer: Creates the product, Fixes bugs, Implements development tasks.

Tester: Closes the bug, Tests the quality of service requirement, Tests the scenarios.

Release Manager: Releases the product.

Conclusions

The article examines the MSF for Agile software development methodology in considerable detail. On the one hand, it shows the correlation of its constructs with the metamodel from the standard [3]. On the other hand, it shows the correspondence of these same constructs with the constructs of the corresponding SoFr.

Combining both methodologies – MSF and Agile – is not necessary. However, their knowledge and understanding will reduce the number of errors in software development projects, and will also allow you to choose a practically more economical and efficient way of their implementation.

The authors believe that the material presented is sufficient for an interested company to create its own methodology based on MSF Agile. However, the use of Visual Studio Team System is not mandatory.

An important difference of the article is the combined description of the elements of the methodology and technology of software development. This significantly differs from many sources that present only individual facts about the methodologies (technologies) of software development. As a rule, little attention is paid to the hierarchy of these facts, although it is important. An obvious example is the confusion even in the understanding of the terms methodology and technology.

References

1. Dyshlyk, O., & Chabaniuk, V. (2025). Karkasnyi pidkhid yak stratehiia doslidzhennia proektuvannia skladnykh prostorovykh informatsiinykh system (na prykladi NIHD) [Framework approach as a strategy for designing complex spatial information systems (case of NSDI)]. *Zemleustrii, kadastr i monitorynh zemel*, *1*, 104–130. DOI: https://doi.org/10.31548/zemleustriy2025.01.09

2. Chabaniuk, V., & Dyshlyk, O. (2025). Karkas rishen Microsoft (KaRi M) yak uzahalnena metodolohiia karkasnoho pidkhodu povodzhennia z prostorovymy informatsiinymy systemamy [Microsoft Solutions Framework as a generalized methodology of the framework approach to spatial information systems]. *Zemleustrii, kadastr i monitorynh zemel*, *2*, 88–105. DOI: https://doi.org/10.31548/zemleustriy2025.02.08

3. International Organization for Standardization. (2014). *ISO/IEC 24744: Software engineering — Metamodel for development methodologies* (2nd ed.).

4. Holt, J. (2023). *Systems engineering demystified: Apply modern, model-based systems engineering techniques to build complex systems* (2nd ed.). Packt Publishing.

5. Brambilla, M., Cabot, J., & Wimmer, M. (2017). *Model-driven software engineering in practice* (2nd ed.). Morgan & Claypool Publishers. DOI: https://doi.org/10.2200/S00441ED1V01Y201208SWE001

6. Wilson, S. F., Maples, B., & Landgrave, T. (1999). *Analyzing requirements and defining solutions architecture*. Microsoft Press.

7. Turner, M. S. V. (2006). *Microsoft solutions framework essentials: Building successful technology solutions*. Microsoft Press.

8. Microsoft. (2024). *MSF for agile software development process guidance* (Version 4.1.61114). Available at: https://www.microsoft.com/en-us/download/details.aspx?id=5365

9. Dyshlyk, O., & Chabaniuk, V. (2026). Pro metodyku karkasnoho pidkhodu do stvorennia prostorovykh informatsiinykh system z vykorystanniam suchasnykh tekhnolohii Microsoft [On the methodology of the framework approach to spatial information systems using modern Microsoft technologies]. In *Suchasni dosiahnennia heodezychnoi nauky ta vyrobnytstva, 51*. Lviv: Vydavnytstvo Lvivskoi politekhniky.

10. Pressman, R. S., & Maxim, B. R. (2019). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

11. Abrams, S. (2024). *Agile software development for beginners: Mastering flexibility and efficiency in modern software projects*. SamIzdat.

12. McConnell, S. (2019). *More effective agile: A roadmap for software leaders*. Construx Press.

13. Meier, J. D. (2005). *MSF agile at a glance*. Available at: https://jdmeier.com/msf-agile-at-a-glance/

14. van Gigch, J. P. (1991). *System design modeling and metamodeling*. Springer.

15. Tekinerdogan, B. (2018). Situational method engineering for constructing Internet of Things development methods. In B. Shishkov (Ed.), *Business modeling and software design* (pp. 221–239). Springer. DOI: https://doi.org/10.1007/978-3-319-94214-8_14

16. Tykhokhod, V. O., Hurin, A. L., & Bespala, O. M. (Comp.). (2024). *Software development technologies: Lecture course* [Electronic resource]. Kyiv: Igor

Sikorsky Kyiv Polytechnic Institute. Available at: https://agilemanifesto.org/iso/uk/manifesto.html.

В. С. Чабанюк, О. П. Дишлик

СТАНДАРТИЗАЦІЯ МЕТОДОЛОГІЇ MSF AGILE З ДОПОМОГОЮ ISO/IEC 24744

У роботі пропонується стандартизація методології розроблення програмного забезпечення MSF Agile. Вона здійснюється з допомогою метамоделі методологій із стандарта ISO/IEC 24744. Оригінал MSF Agile називається MSF for Agile Software Development і позначається MSF4ASD. MSF4ASD реалізована в MSF версії 4.0 у 2005 р. в інструменті/технології Visual Studio Team System. Тому тут вона називається прикладом конкретної методології розроблення програмного забезпечення. Без реалізації методологія існує, але її краще називати конкретизованою узагальненою методологією MSF з загрозою втрати практичності.

З огляду на актуальність методологій Agile, MSF Agile може бути практично корисною реалізацією узагальненої методології MSF. Конкретну методологію MSF Agile можливо отримати двома способами: 1) двохранговою специфікацією або 2) стандартизацією з наступною специфікацією. У цій статті вибрано другий спосіб – спочатку виконується стандартизація MSF Agile з допомогою метамоделі методологій розроблення із стандарта ISO/IEC 24744. Після цього простіше виконувати практично корисну специфікацію, оскільки задача стає типовою. Реалізація буде можливою з використанням різних інформаційних технологій (IT), включаючи IT Microsoft. Конкретна методологія може вже безпосередньо використовуватися на практиці.

У трьох основних розділах цієї роботи: 1) вводиться засіб стандартизації – описуються потрібні елементи метамоделі методологій розроблення ПЗ із стандарту ISO/IEC 24744; 2) нагадується інформація про MSF4ASD, яка тут спрощена до MSF Agile; 3) MSF Agile представляється з допомогою елементів метамоделі із стандарта ISO/IEC 24744. Цим самим

показується, як зробити MSF Agile стандартизованою методологією. Потім з неї однокроковою редукцією можливо отримати конкретну методологію, яка може конструктивно задовольнити Каркасний підхід і полегшити перехід до Базованої на Патернах Просторової Інженерії.

Ключові слова: *методологія Microsoft Solutions Framework (MSF) Agile, ISO/IEC 24744 стандартизація MSF Agile*