ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Том 111

ANNUAL OF SOFIA UNIVERSITY “ST. KLIMENT OHRIDSKI”
FACULTY OF MATHEMATICS AND INFORMATICS
Volume 111

EXPLORING SOFTWARE ENGINEERING KNOWLEDGE DOMAINS

DILYAN GEORGIEV

Software engineering, as the primary value-based process, is influenced by the culture a given company establishes, formal and informal procedures, and technological improvements introduced during the implementation. Having this in mind it could be articulated further how these innovative methods could result in best practices, communication flows, strong-bonded teams, and successful projects. By adding an organisational domain to the model, the relationships between management's organizational aspects and technological development approaches are discussed.
This paper aims to explore and classify the software engineering domains in order to facilitate the process of managing knowledge within ICT companies. Based on a literature overview, a model is proposed which combines the multiple perspectives for adopting knowledge management practices more efficiently.

Keywords: software engineering, knowledge management, software process improvement, knowledge domains, overview

2020 Mathematics Subject Classification: 68N01, 68N30

CCS Concepts:
• Software and its engineering~Software organization and properties~Contextual software domains

1. Introduction

Software engineering is a relatively new knowledge domain that has been through many changes during the last few decades. Considering the multiple perspectives having its influence, it emerges out of the ICT, also recognized as an industry about other industries because of the role it has in their growth and evolution. Furthermore, software engineering as a “knowledge-intensive activity” [6] results in solutions, expected to play an even more critical role in society and the economy, for example,

DOI: 10.60063/GSU.FMI.111.35-53

by implementing artificial intelligence technologies in new cyber-physical solutions such as robotised and autonomous systems – self-driving cars, unmanned aerial vehicle (UAV), intelligent systems, and many more. Enabling software companies to improve the way they manage their knowledge processes is increasingly important in this specific moment when hardware-based infrastructures and systems are being upgraded with digital functionalities. Suppose this is put through a more global perspective. In that case, digital solutions aim to tackle complex socio-economic and ecological problems that, when solved, will cause a radical improvement in the way our environment is being constructed. Concepts like smart cities, green economies, and meta worlds could provide a new way of handling those problems and set up a new course of the way software engineering is progressing.

Knowledge management, on second thought, introduces many instruments, methods, practices, and approaches for value creation, especially in knowledge-intensive industries which "has spurred an exponential increase in publications covering a broad spectrum of diverse and overlapping research areas" [43]. The processes of codification and personalisation lay in the basis of those instruments since the company should support knowledge generation, sharing, and transferring among the employees. This is crucial, especially in software processes that require high expertise since much of the knowledge used for a given solution is wrapped as intangible assets. However, stimulating these processes could provide a valuable setup for the project's growth.

Furthermore, new software engineering methods and concepts are introduced to validate requirements beforehand and to apply methodologies at the team level. Hence, a value co-creation system could be established, involving different stakeholders, and coordinating between complex business models. The user participates more actively and provides feedback that helps developers work on well-defined functionalities. On the other hand, new solutions require better cohesion between the architecture of a given product, communication channels, and handling changes through time. Considering how dynamic and sometimes unpredictable the software industry is about technologies and paradigms, additional expertise is needed regarding specific solutions and management on different levels for processes, experiences, and integrations with other systems.

All this demonstrates the need for improved KM processes in software engineering. Thus, a new paradigm is proposed that could provide bigger clarity on how critical it is to create relations between the different management aspects, on the one hand, and the software development, on the other hand. Through that perspective, knowledge seems even more valuable highlighting how important it is to introduce a new understanding of how it can be efficiently managed within the software engineering aspect and the organizational and socio-economical contexts.

The main goal of this research is to overview the latest methods, approaches, and trends in software process improvement, focusing on the domain and organisational concepts of management. To do that, two critical questions are answered:

- How the organisation handles its intellectual capital to achieve its purposes and goals?

- How is different knowledge being handled in different domains?

After analysing how knowledge evolves within these two aspects, the idea is developed further by inspecting previous systematic literature reviews. The concepts that lay the foundations of current ICT evolution are highlighted and a simple model is introduced, which represents how knowledge can be categorised through the organisational and domain perspectives (Section 2). This will introduce an advanced view of areas in which studies have been focused. Then, after overviewing papers retrieved from different resources, many methods, architectures, and methodologies are identified that are used to handle the issues related to knowledge management (Section 3). A short discussion is presented in Section 4, and conclusions and predictions are made in Section 5.

2. Theoretical background foundations

There have been many changes in how knowledge is perceived in the software industry. At first, knowledge was considered a possession, "something that could be captured" [6]. However, currently, it is much more important to have the knowledge applied than to possess it – it is called the knowledge-in-action concept, and it is even more critical if not involved in the software process. Being context-dependent, applying the knowledge is an activity that needs additional preparation, as internal stakeholders must be aware of the available explicit and tacit knowledge [39]. "Tailoring software development" by applying agile concepts draws the development team's attention, strengthens the decision-making process, increases inter-team coordination, and builds collective-code ownership [31]. The creation, storage, transfer, and retrieval activities must be introduced to generate new ideas that could support the product, expand its scope, and implement its functionalities according to the stakeholders' expectations. Regarding product delivery, there are frameworks like DevOps that could guarantee product quality and simplify and fasten integration processes with the production environment [10]. This corresponds with the goals regarding customer expectations – building trust, bringing satisfaction, and establishing the belief that "working software is the primary measure of progress" [34].

To define the SE Knowledge domain, a simple concept representation is made, which includes 16 systematic reviews and their main topics (Figure 1). By comparing different aspects, an analysis of the connections between domains is made, and subdomains are identified and highlighted.

First, two significant distinctions must be made between KM and SE concepts. When discussing organisational topics, the organisational culture and structure should be considered, as well as the mission, vision, learning, and growth. The organisational structure supports transferring tacit knowledge through its levels of management [3]. The culture company supports, on the other hand, is critical for sharing between teams and departments. The mission and vision dictate the overall motivation among the employees, and learning and growth could help to introduce new opportunities and challenges.

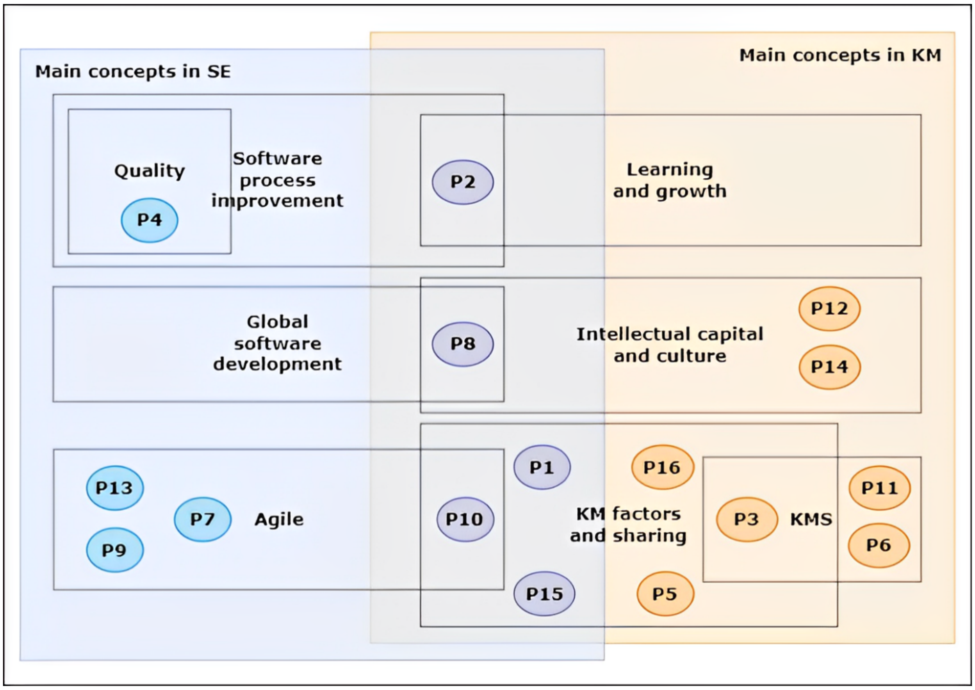

Figure 1. Main SE and KM concepts found in research data

In terms of domains, the focus is on the knowledge that comes from the specific environment and requires expertise that could provide solutions to a problem or set of problems. In that manner, Knowledge management is used to drive at least one of the five directions of development [43]:

- Ontology of Knowledge;
- Knowledge Management Systems (KMS);
- Role of Information Technologies;
- Managerial and Social issues;
- Knowledge Measurement.

All these directions aim to support the codification and personalisation processes – previously defined in the SECI model, which describes the different types of transformations between tacit and explicit knowledge – "organisational knowledge creation involves continual interaction" between those two types of knowledge [28]. Those models can be added to the models defined by Nonaka and Takeuchi, Davenport and Prusak, and Stewart, who define it as the "foundation of Knowledge management and Intellectual Capital field" [50] and later try to set up a starting point for the current overview.

2.1. Domain Knowledge in SE

The term domain is considered abstract since it expands the development field by adding processes supporting a given product in its early-stage development. These processes are critical in software engineering since they contribute to the scope definition, architecture styles, product development, and delivery.

Approaches that establish concepts with higher agility coefficients affirm the influence of the domain significantly when the scope is constantly changing, or the software is planned to be supported in long-term exploitation. In both cases, all decisions must be consistent within the context, technology trends, and other factors.

Most of the papers focus on how different requirements – functional and non-functional affect the decisions regarding the software structure and the technology stack used for its implementation. It is also necessary to distinguish between methodologies focused on the development process and design patterns related to the product's overall performance. Having this in mind, Requirements engineering is defined as the main driver in SE, on one hand, and Software architecture on the other hand.

To clarify the approaches applied when generating knowledge, the domain knowledge is split into Product and Process knowledge fields of research. Thus, it will be acknowledged how a product could be described and normed according to the client's expectations and how the process defined during the project's implementation could contribute to the team's efficiency by establishing a stable and easy-to-follow workflow.

2.1.1. Product knowledge

When a product is described, it is essential to define it as a documented set of components with their functionalities and relations. Descriptive languages often provide a machine-readable domain specification that could be enriched by introducing meta-data and using conflict-detecting mechanisms based on a three-step analytic process for validation, verification, and performance metrics.

In terms of the state KM is being introduced, this approach is used to transform, for example, legacy systems "into easily accessible, well described, and interoperable, modular services" [27]. Later, these descriptions could allow the team to apply changes more quickly, especially within the agile software delivery process. Thus, descriptive languages like UML, VEL [5], and WADL have a supportive role in different development designs such as Quality-driven Architecture Design (QDAD), Quality-Aware Rapid Software Development Design (Q-Rapids) [16], Transformation models in dynamic environments [56] and older pipeline dev models like PLUSS (Product Line Use case modelling for Systems and Software engineering) [1].

Meta-based models represent another perspective – focusing on the software design, the business process, and its persistence, different data objects could be categorised into relationships and correspondence models depending on the type, usability, and consistency [11], prioritization [36], effort estimation, predictions, and size measurement [42]. Compliance modelling, on the other hand, is focused on the abstract level of requirements engineering and identifies challenges in system modelling, such as “missing linkage with the business use cases” [57] or difficulties in understanding the connection between compliance requirements and design rules. This adds new ways of understanding and could be developed further into theories that could provide formalization and focus on stakeholders, opportunities, human resources, working practices, and the software system itself [14]. All these models and methods could be monitored via classification methods such as Gamification, which, when applied during the development process, improves the team’s overall performance [18], Prototyping, which focuses on data science projects and its three main elements: Experimentation, Development, and self-discipline [2], or CASE which considers Technical Debt when trying to reconstruct already existing solutions [24].

In product-centric companies, knowledge networks could provide critical information via internal and external communication. For example, PISA models are based on feedback and use different rating systems so the company can “identify improvement indications for next releases” [12]. Including the user in the development process is also applied in “User-Centred Designs”, used for software products whose end users have different expertise. Using the software is defined as part of their working routine [60]. Another way to establish effective product management is the product road mapping approach [35] or the Visual Milestone Planning (VMP) [15] approach, which supports the product discovery process and the agile transformation, including the stakeholders in features prioritization. This applies in industries such as Video Gaming, where virtual environment simulation is a crucial activity when validating new ideas.

2.1.2. Process knowledge

Capturing the knowledge that a given software process generates is a challenge for every team leader. Different methods that support this knowledge-gathering activity are identified by separating the Software Development Lifecycle (SDLC). There are various solutions, some focusing on how data is being handled, others – on monitoring changes and product engineering.

In the early stages of the product, the main goal is to clarify requirements and come up with a solution based on previous experiences. The portfolio-driven development model is an advanced model of APLE (Agile Product Line Engineering) and demands a more agile and rapid approach using “a collection of projects, programs, and the other operations for achieving the business goals” [25]. Based on that, collection managers can provide more accurate estimations, especially when working with fixed resources.

During the implementation phase, the focus is on the value provided and how the development process is executed – Waterfall, Iterative, V-Shaped, or Spiral. There are many models with different priorities. However, when it comes to continuous improvement, there is a need for a transition model like LACE (Lean-Agile Centre of Excellence) [8], which could remove impediments in large-scale projects.

Later during the quality assessment phase, non-functional requirements like security, maintainability, transparency, and endurance are considered. The result is a set of maturity models and concepts that target the SDLC, like the Capability Maturity Model, which aims to "evaluate and assess security engineering practices", Cybersecurity Capability Maturity Model, which is designed to help organisations "to improve their cybersecurity programs", and Software Assurance Maturity Model, which is an open framework for practice evaluation [37] which targets self-adaptive systems and their business goals. Other concepts focus on user feedback and how it is introduced to the requirements evolution [29], how the agile process's efficiency is measured [44], what metrication should be applied in order to increase the transparency in customer-relationships-based platforms [40], microservice applications [7], business intelligence systems [13] and industrial software products [45], how cybersecurity vulnerabilities could be identified via different communication models [26], and even how QA could be evaluated statistically via algorithms like Majority voting, ZenCrowd and Naïve Bayes [17].

2.2. Organisational knowledge in SE

One of the fundamental research projects is a state-of-the-art report which identifies the need of investigating the connection between how knowledge is managed and how software processes are coordinated [46]:

Needs regarding behavioural KM:

- The need for domain knowledge;
- The need for knowledge capture and share;
- The need for knowledge about local policies;
- The need for knowledge about who knows what;

Needs regarding technocratic KM:

- The need for knowledge about new technologies;
- The need for knowledge about distance collaboration;
- The need for knowledge about new challenges and opportunities;

Having that in mind, different kinds of research could be further categorised into two significant types of knowledge management. Similar classifications are discussed through the years by adding more and more details regarding relationships and possible outcomes.

2.2.1. Technocratic knowledge

Major studies point out the need for a more technocratic approach toward KM. This includes cognitive analysis, a closer look at communication within a virtual reality, event modelling, and data clustering based on common features [38]. This is essential in project management since codification and documentation are vital activities when monitoring [4]. Knowledge modelling focused on applying the conceptual and computational models when extracting functions or processing clustered data – perceived complexity and managed risks contribute to using knowledge artifacts. So, several aspects of knowledge management can be classified within this technocratic approach.

First, starting on a more global level, knowledge management systems should be focused on software products like audit tools, HR management tools, and collaboration software. For example, audit tools help the organisation with crucial activities via assets mapping, landscape mapping, flowcharts, competitive analysis, diagnostics, critical function analysis, and benefit assessment [20]. The whole audit modelling is fundamental if the company wants to improve its culture, establish an effective working environment, and use its knowledge beneficially. From a technological perspective, the KMS could be centralized and decentralized and apply techniques like knowledge discovery using surveys and audits, introducing knowledge inventorying and mapping, broadcasting knowledge, competitive analysis, and diagnostics [21].

Second, going further into info structures and data analysis, the main drivers are the knowledge itself, characterised by a specific domain, a business process with a given goal, and a context contributing to the process to be executed [22]. The system's design should be considered according to the scope, developing methodology, human resources, additional assets, KM practices, and infrastructure elements [23]. Another critical issue that should be discussed is the monitoring process which should provide abstract, meta-oriented views reflecting the organisational structure [55].

2.2.2. Behavioural knowledge

Regarding the behavioural approach, two main directions should be analysed, one wrapped around team modelling and finding patterns within larger-scale projects. The other is based on the Agile framework and how it could transform more prominent companies into strongly connected knowledge communities.

Team patterns. Team patterns usually try to identify specific behaviour types within a team that could be mapped with improved activities during work. Some of the papers observed are focused on teamwork in educational institutions where students are grouped and work in similar environments. Thus, the researchers could identify different approaches in work by prioritizing the main drivers in software development. For example, some of the process patterns include engineering-driven development, where the main drivers are planning and design, code-driven development, where coming up with prototypes is more efficient, and ad-hoc development, where the priority is dynamic and could go towards engineering, coding or verification and validation [19].

Research shows that human factors are under deep analysis, and according to literature overviews, the main clusters describe how they are introduced effectively in the SE process and how they improve it via education and motivation [32]. A similar analysis applied on a large scale identifies the need to audit existing methods and practices so teams can be more prepared and adaptable to any context. For example, Global Software Development focuses on more abstract principles defining software development as a "human-centric and socio-technical activity" and taking cultural context into account when managing larger projects [33].

Agile on a larger scale. Since agile was successfully integrated into smaller companies, managing to build compact and effective teams that could quickly deliver value according to clients' expectations, bigger enterprises try to adapt this simple framework on a larger scale. Methods like Large Scale Scrum (LeSS), which could be applied to up to "10 Scrum teams (of seven people)", and LeSS Huge, which could be applied to a "few thousand people working on one product" [48, 52] are influential, especially when companies do not pay attention at early stages and grow up rapidly. Scaled Agile Framework (SAFe) and Disciplined Agile Delivery (DAD), on the other hand, define four levels: Team level, Program level, Portfolio level, and Value stream level [41]. Some measurements identified via surveys are lead cycle and release time, the value provided, the number of defects found, velocity, automation, and predictability [30]. Others focus on maturity modelling, where levelling criteria are strongly related to the agile process's performance [49].

However, the main driver of the agile transformation is the agile coach himself – his responsibilities include "building teams by providing realistic support during implementation of agile processes, leading the team towards self-organisation" and "creating guidelines, setting goals and roadmaps" [53]. Lastly, agile transformation must be evaluated via success factors like shared product vision, shared responsibilities, shared knowledge, feedback, and ownership, and failure factors like lack of middle management support, barriers to the production environment, excessive control by the higher management, and lack of understanding from the stakeholders [51].

3. Model representation

Considering the different paradigms and aspects described in the previous section, four groups can be formed depending on the context (Table 1), and key concepts that include these knowledge domains can be identified.

The first group combines everything related to the product – studies are focused on different approaches in the development setup that correspond to the context,

Table 1. Knowledge domain groups

Domain		**Organisation**	
Product	**Process**	**Behavioural**	**Technocratic**
Design	Change	Team	Knowledge
Architecture	Improvement	Agile	Systems
Engineering	Model	Approach	Projects
Systems	Metrics	Development	Model
Software	Quality attributes	Methodologies	Practices
Data	Analysis	Scrum	Management

from architecture-leading technologies like the Internet of things and Artificial Intelligence, through domain-based concepts like Automobile industries and eLearning, to fundamental processes of software engineering like Requirements elicitation and Design patterns. This forms a wide field of discoveries about the domain that could be patterned.

The second group is formed from the idea of Software Process Improvement. These studies centre the process as a systematically arranged set of activities, each contributing to the product's development. This sets up requirements regarding quality and introduces metrics that help validate and verify the software. Deployment and Maintenance are also very critical phases of every project – approaches like DevOps and Meta-Modelling seriously impact how people can manage the product after its release.

The third group is concentrated on human resources. The need to manage intellectual capital triggers another narrative in scientific discoveries, which includes experiments on all types of projects and their teams. These experiments investigated how teams are formed in agile development, how technologies can be introduced and later improved to reflect a team's productivity, and how methodologies like Lean and Scrum tackle significant problems like low motivation, knowledge loss, and impediment documentation.

The last group is more abstract than the others – its goal is to categorize and systematize any knowledge related to the other three domains and use it strategically toward a successful closure. This includes Business patterns, Knowledge audits, and Information systems that support the organisation.

The model is represented in Figure 2, reviewing how different types of knowledge correspond to each other and formalise communications on higher and lower scales. It identifies knowledge domains and how they could be introduced according to the

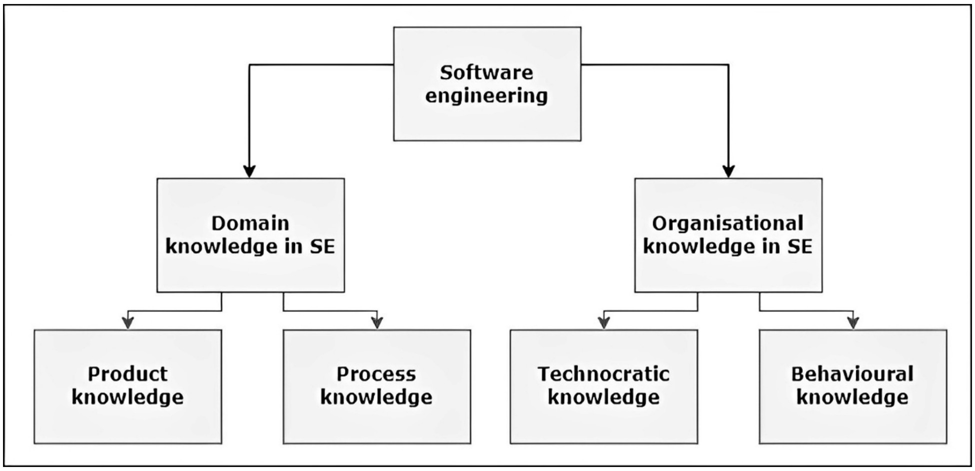

Figure 2. Knowledge types

perspectives described above, but also identifies the main aspects that drive the communication flow both horizontally and vertically through the organisation:

- **Technologies** represent all technological knowledge used for the solutions company provides to the market. Technologies depend on the set of problems developers want to tackle, the difficulty that should be reached to deliver a Minimum viable product (MVP), and the trends that influence the environment. The model compares the tech perspective to the organisation's mission and goals, as well as previous solutions developed;

- **Methodologies** represent all procedures the company applies regarding how business processes should be executed, how knowledge is transferred from one process to another, and how problems are handled to mitigate casualties. Methodologies depend on the overall dynamics, the organisational structure, what efficiency metrics have been established and how the product is being represented as components that need to be delivered. If the deployment plan requires bigger accuracy and quality is a primary driver of development, the sprints are more extended with fewer deficiencies found afterward. However, if the goal is value-oriented, shorter sprints with more frequent feedback are expected;

- **Culture** represents how the organisation introduces the working environment, and what values support its progress. This depends on the company's mission and vision and the socio-economic factors that influence the market. If it is an international company, the culture is more globally considered since it should apply to as many people as possible. But if the company is a local enterprise that works with people within a given country, the culture should be oriented according to the local mentality;

- **Human resources** represent all employees' experience and abilities as assets that should be monitored and used to fulfil company's goals. They depend on the technology stack defined, the culture the organisation establishes, and the environment considered internally and externally.

4. Discussion

Several trends can be analysed, further developed, and enriched. First, KM has a variety of applications on different levels, both from managerial and functional perspectives. These aspects are cross-examined by describing the four different types of knowledge, which shows the approaches and trends discussed in the previous sections and how they support the KM process (Figure 3).

Technocratic knowledge a given company relies on supports the process knowledge by setting up procedures everybody should know, and quality standards teams should strictly comply with. By introducing a knowledge modelling structure that correlates to the organisational one, different methodologies applied on a larger scale, like Agile, could confirm a well-described, balanced, and customised set of workflows,

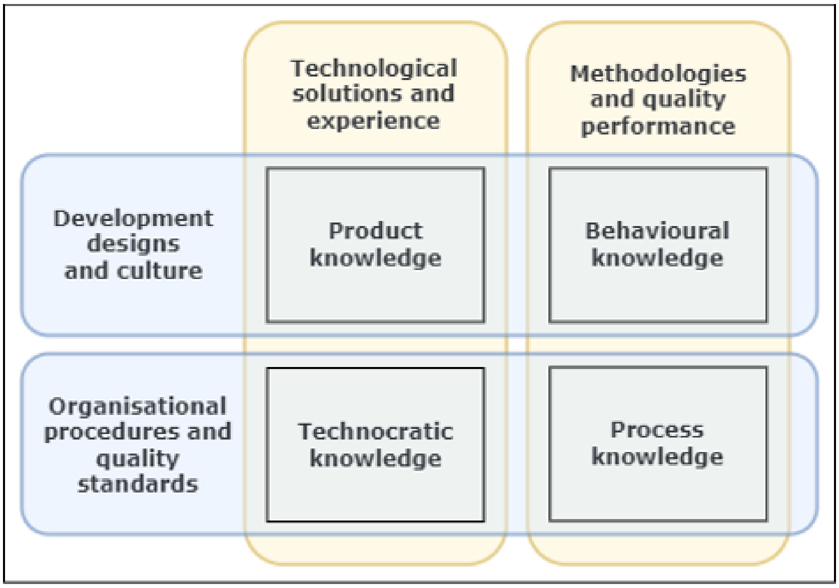
Technological solutions and experience
Methodologies and quality performance
Development designs and culture
Product knowledge
Behavioural knowledge
Organisational procedures and quality standards
Technocratic knowledge
Process knowledge

Figure 3. Knowledge types relations

metrics, and instructions managers can use when starting a new project. This sets the foundation that could guarantee a good start for every development process.

Technocratic knowledge could also support product knowledge by suggesting technological solutions based on previous experiences and saving new experiences, using them to improve the company's reputation according to the dynamic ICT environment. This is critical for industries whose progress counts on R&D and concurrency between different enterprises.

On the other hand, behavioural knowledge focuses on how these solutions mentioned above correspond to the organisational culture and the company's vision. In terms of product knowledge, this results in development designs and teams that split functionalities according to a given semantics and tackle technological problems by setting up a CI process that supports the client's expectations. In process knowledge, however, the essential part stands for quality assessment and how methodologies guarantee the proper execution of the development itself.

By further discussing how this categorisation could be expanded via local and global perspectives, problems are meant to be handled from top-down and bottom-up approaches. As Table 2 shows, knowledge is used locally for specific projects and on a global scale by adding new experiences or improving overall performance. This KM concept is relatively simple yet effective since there is a clear division between the management and the development that could be handled by a customised communication strategy, for which bigger experience is needed.

Table 2. Knowledge representation

		Technocratic knowledge		Behavioural knowledge	
		Global scale	Local scale	Global scale	Local scale
Product knowledge	Global scale	Experience and portfolio knowledge		Organisational culture	
	Local scale		Technological solutions		Development designs
Process knowledge	Global scale	Quality standards		Quality performance	
	Local scale		Organisational procedures		Methodologies

5. Conclusion and future work

In conclusion of this overview, Knowledge management has a meaningful role in data flows not only on an organisational level but also on a project level. It can be argued, especially in small and medium-sized companies, that the organisational structure aims to evolve towards the communication channels – a process described by Conway's law. This corresponds to the vital process that started in the 90s with the first methodologies, developed further in the 2000s, and formalized fully nowadays.

Requirements should be more descriptive and considered when making software structure decisions. When starting a project, most of the problems are related to how teams describe the domain, how the context is perceived, and how communications are established. If the scope is clear and well-defined, future problems could be predicted not only about the development process but also about human resources. This is a problem KM supports by introducing different types of solutions related to the technological perspective on the one hand – KMS, automation, and knowledge migration, and to the cultural perspective on the other – patterns, procedures, and learning by sharing methods.

It is critical to highlight the role of the business itself – more and more companies delegate resources to investigate their data flows and how knowledge has been spread among team members. This could be used for updating the organisation's culture, mission, and goals. In addition, big companies generate statistical data that could be used for academic research, resulting in theories and taxonomies described in the public literature.

In the future, these relations will be observed not only on the theoretical level but also in practice. Knowledge contributes critical value to a company's success and will be further observed as enabling factor of shared learning and growth.

Appendix

Id	Citation	Keywords
P1	Ragab and Arisha [43]	knowledge measurement, knowledge management
P2	Bjørnson and Dingsøyr [6]	software engineering, knowledge management, learning software organization, software process improvement, systematic review
P3	Centobelli et al. [9]	entrepreneurship, factors affecting KM, KMSs, knowledge management, performance, start-up firms, scalability
P4	Céspedes et al. [10]	systematic literature review, DevOps, product quality, ISO/IEC 25000
P5	M. Asrar-ul-Haq et al. [3]	knowledge management, knowledge sharing, antecedents, trends
P6	Iskandar et al. [27]	knowledge management system, KMS, current issues, systematic literature review, big data issue in KMS
P7	Kiv et al. [30]	agile manifesto, agile methods, agile methods adoption, partial agile adoption, systematic literature review
P8	Marinho et al. [32]	global software development, global teams, culture, systematic literature review
P9	Mora et al. [33]	agile paradigm tenets, agile ITSM methods, agile software engineering methods, FitSM, IT4IT, representative literature analysis
P10	Ouriques et al. [38]	knowledge management, agile software development, knowledge processes
P11	Saad and Zainudin [47]	computational thinking, project-based learning, PBL-CT, teaching and learning strategies
P12	Serenko and Bontis [50]	knowledge management, process management, intellectual capital
P13	Stray et al. [53]	agile coaching, skills, tasks, systematic literature review, agile transformation, software development practices
P14	Theobald et al. [54]	agile leadership, agile management, agile organization, motivation, systematic literature review
P15	Venkitachalam and Busch [58]	tacit knowledge, implicit knowledge, knowledge management, research
P16	Wang and Noe [59]	knowledge sharing, knowledge exchange, knowledge management

References

[1] F. Ahmed and L. F. Capretz, The software product line architecture: An empirical investigation of key process activities, Inf. Softw. Technol. 50(11) (2008) 1098–1113, https://doi.org/10.1016/j.infsof.2007.10.013.

[2] T. Aho, O. Sievi-Korte, T. Kilamo, S. Yaman and T. Mikkonen, Demystifying data science projects: A look on the people and process of data science today, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 153–167, https://doi.org/10.1007/978-3-030-64148-1_10.

[3] M. Asrar-ul-Haq, S. Anwar and T. Nisar, A systematic review of knowledge management and knowledge sharing: Trends, issues, and challenges, Cogent Business & Management 3(1) https://doi.org/10.1080/23311975.2015.1127744.

[4] V. C. Ayarza and S. Bayona-Oré, Cluster monitoring and integration in technology company, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Adv. Intell. Syst. Comput. 1071, Springer, Cham, 2020, 253–265, https://doi.org/10.1007/978-3-030-33547-2_19.

[5] A. Bagnato et al., Showcasing Modelio and pure:variants Integration in REVaMP^2 Project, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 590–595, https://doi.org/10.1007/978-3-030-35333-9_43,

[6] F. O. Bjørnson and T. Dingsøyr, Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used, Inf. Softw. Technol. 50(11) (2008) 1055–1068.

[7] J. Bogner, S. Schlinger, S. Wagner, and A. Zimmermann, A Modular Approach to Calculate Service-Based Maintainability Metrics from Runtime Data of Microservices, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 489–496, https://doi.org/10.1007/978-3-030-35333-9_34.

[8] J. Bowring and M. Paasivaara, Keeping the Momentum: Driving Continuous Improvement after the Large-Scale Agile Transformation, in: Product-Focused Software Process Improvement (PROFES 2021), ed. by L. Ardito et al., LNCS 13126, Springer, Cham, 2021, 66–82, https://doi.org/10.1007/978-3-030-91452-3_5.

[9] P. Centobelli, R. Cerchione and E. Esposito, Knowledge management in startups: Systematic literature review and future research agenda, Sustainability 9(3) (2017) 361, https://doi.org/10.3390/su9030361.

[10] D. Céspedes, P. Angeleri, K. Melendez and A. Dávila, Software product quality in DevOps contexts: A systematic literature review, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Advances in Intelligent Systems and Computing 1071, Springer, Cham, 2019, 51–64.

[11] M. El Hamlaoui, S. Bennani, S. Ebersold, M. Nassar, and B. Coulette, AHM: Handling heterogeneous models matching and consistency via MDE, in: Evaluation of Novel Approaches to Software Engineering (ENASE 2018), ed. by E. Damiani et al., Communications in Computer and Information Science 1023, Springer, Cham, 2018, 288–313, https://doi.org/10.1007/978-3-030-22559-9_13.

[12] F. Falcini and G. Lami, Embracing software process improvement in automotive through PISA model, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915 Springer, Cham, 2019, 73–88, https://doi.org/10.1007/978-3-030-35333-9_5.

[13] I. Figalist, C. Elsner, J. Bosch and H. Olsson, An end-to-end framework for productive use of machine learning in software analytics and business intelligence solutions, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 217–233, https://doi.org/10.1007/978-3-030-64148-1_14.

[14] J. Fischbach, J., Frattini, D. Mendez, M. Unterkalmsteiner, H. Femmer and A. Vogelsang, How do practitioners interpret conditionals in requirements?, in: Product-Focused Software Process Improvement (PROFES 2021), ed. by L. Ardito et al., LNCS 13126, Springer, Cham, 2021, 85–102, https://doi.org/10.1007/978-3-030-91452-3_6.

[15] D. Fontdevila, M. Genero, A. Oliveros and N. Paez, Evaluating the utility of the usability model for software development process and practice, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 741–757, https://doi.org/10.1007/978-3-030-35333-9_57.

[16] X. Franch, L. Lopez, S. Martínez-Fernández, M. Oriol, P. Rodríguez and A. Trendowicz, Quality-aware rapid software development project: The Q-rapids project,

in: Software Technology: Methods and Tools (TOOLS 2019), ed. by M. Mazzara et al., LNCS 11771, Springer, Cham, 2019, 378–392, https://doi.org/10.1007/978-3-030-29852-4_32.

[17] T. Fredriksson, D. I. Mattos, J. Bosch and H. H. Olsson, Data labeling: An empirical investigation into industrial challenges and mitigation strategies, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 202–216, https://doi.org/10.1007/978-3-030-64148-1_13.

[18] G. A. García-Mireles and M. E. Morales-Trujillo, Gamification in software engineering: A tertiary study, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Adv. Intell. Syst. Comput. 1071, Springer, Cham, 2020, 116–128, https://doi.org/10.1007/978-3-030-33547-2_10.

[19] E. Germain and P. N. Robillard, Towards software process patterns: An empirical analysis of the behaviour of student teams, Inf. Softw. Technol. 50(11) (2008) 1088–1097, https://doi.org/10.1016/j.infsof.2007.10.018.

[20] E. Gourova, A. Antonova and Y. Goleminova, Knowledge audit concepts, processes and practice, WSEAS Trans. Bus. Econ. 6 (2009) 605–619.

[21] E. Gourova and M. Dragomirova, Design of knowledge management info-structures, in: EuroPLoP'15: Proc. 20th Eur. Conf. on Pattern Languages of Programs, Art. No 15 (2015) 9 pp.

[22] E. Gourova and Y. Todorova, Knowledge audit data gathering and analysis, in: EuroPLoP'10, Art. No 14 (2010) 7 pp.

[23] E. Gourova and K. Toteva, Design of knowledge management systems. VikingPLoP 2014, Art. No 3 (2014) 15 pp.

[24] D. Guamán, J. Pérez, J. Garbajosa and G. Rodríguez, A systematic-oriented process for tool selection: The case of green and technical debt tools in architecture reconstruction, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 237–253, https://doi.org/10.1007/978-3-030-64148-1_15.

[25] K. Hayashi and M. Aoyama, A portfolio-driven development model and its management method of agile product line engineering applied to automotive software development, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 88–105, https://doi.org/10.1007/978-3-030-64148-1_6.

[26] M. Hell and M. Höst, Communicating Cybersecurity Vulnerability Information: A Producer-Acquirer Case Study, in: Product-Focused Software Process Improvement (PROFES 2021), ed. by L. Ardito et al., LNCS 13126, Springer, Cham, 2021, 215–230, https://doi.org/10.1007/978-3-030-91452-3_15.

[27] S. M. Huang, Y. T. Chu, S. H. Li and D. C. Yen, Enhancing conflict detecting mechanism for Web Services composition: A business process flow model transformation approach, Inf. Softw. Technol. 50(11) (2008) 1069–1087, https://doi.org/10.1016/j.infsof.2007.10.014.

[28] K. Iskandar, M. Jambak, R. Kosala and H. Prabowo, Current Issue on Knowledge Management System for future research: A systematic literature review, Procedia Comput. Sci. 116(C) (2017) 68–80, https://doi.org/10.1016/j.procs.2017.10.011.

[29] J. O. Johanssen, A. Kleebaum, B. Bruegge and B. Paech, Feature Crumbs: Adapting usage monitoring to continuous software engineering, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS

11271, Springer, Cham, 2018, 263–271, https://doi.org/10.1007/978-3-030-03673-7_19.

[30] P. Kettunen, M. Laanti, F. Fagerholm and T. Mikkonen, Agile in the era of digitalization: A finnish survey study, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al. LNCS 11915, Springer, Cham, 2019, 383–398, https://doi.org/10.1007/978-3-030-35333-9_28.

[31] S. Kiv, S. Heng, M. Kolp and Y. Wautelet, Agile manifesto and practices selection for tailoring software development: A systematic literature review, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 12–30, https://doi.org/10.1007/978-3-030-03673-7_2.

[32] L. Machuca-Villegas and G. P. Gasca-Hurtado, Towards a social and human factor classification related to productivity in software development teams, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Adv. Intell. Syst. Comput. 1071, Springer, Cham, 2020, 36–50, https://doi.org/10.1007/978-3-030-33547-2_4.

[33] M. Marinho, A. Luna and S. Beecham, Global software development: Practices for cultural differences, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 299–317, https://doi.org/10.1007/978-3-030-03673-7_22.

[34] M. Mora, F. Wang, J. M. Gómez and O. Díaz, A comparative review on the agile tenets in the IT service management and the software engineering domains, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Adv. Intell. Syst. Comput. 1071, Springer, Cham, 2019, 102–115, https://doi.org/10.1007/978-3-030-33547-2_9.

[35] J. Münch, S. Trieflinger and D. Lang, What's hot in product roadmapping? Key practices and success factors, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al. LNCS 11915, Springer, Cham, 2019, 401–416, https://doi.org/10.1007/978-3-030-35333-9_29.

[36] E. Nazaruka and J. Osis, The formal reference model for software requirements, in: Evaluation of Novel Approaches to Software Engineering (ENASE 2018), ed. by E. Damiani et al., Communications in Computer and Information Science 1023, Springer, Cham, 2019, 352–372, https://doi.org/10.1007/978-3-030-22559-9_16.

[37] P. Nikbakht, M. Höst, and M. Hell, HAVOSS: A maturity model for handling vulnerabilities in third party OSS components, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 81–97, https://doi.org/10.1007/978-3-030-03673-7_6.

[38] S. Osorio, A. P. P. Negrón, and A. E. Valdez, From a conceptual to a computational model of cognitive emotional process for engineering students, in: Trends and Applications in Software Engineering (CIMPS 2019), ed. by J. Mejia et al., Adv. Intell. Syst. Comput. 1071, Springer, Cham, 2020, 173–186, https://doi.org/10.1007/978-3-030-33547-2_14

[39] R. Ouriques, K. Wnuk, T. Gorschek and R. Berntsson, Knowledge management strategies and processes in agile software development: A systematic literature review, Int. J. Softw. Eng. Knowl. Eng. 29(03) (2019) 345–380, https://doi.org/10.1142/S0218194019500153.

[40] C. R. Prause and A. Hönle, Emperor's new clothes: Transparency through metrication in customer-supplier relationships, in: Product-Focused Software Process

Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 288–296, https://doi.org/10.1007/978-3-030-03673-7_21.

[41] A. Putta, M. Paasivaara and C. Lassenius, Benefits and challenges of adopting the Scaled Agile Framework (SAFe): Preliminary results from a multivocal literature review, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 334–351, https://doi.org/10.1007/978-3-030-03673-7_24.

[42] C. Quesada-López, A. Martínez, M. Jenkins, L. C. Salas and J. C. Gómez, Automated functional size measurement: A multiple case study in the industry, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 263–279, https://doi.org/10.1007/978-3-030-35333-9_19.

[43] M. A. F. Ragab and A. Arisha, Knowledge management and measurement: a critical review, J. Knowl. Manag. 17(6) (2013) 873–901, https://doi.org/10.1108/JKM-12-2012-0381.

[44] P. Ram et al., An empirical investigation into industrial use of software metrics programs, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 419–433, https://doi.org/10.1007/978-3-030-64148-1_26.

[45] P. Ram, P. Rodriguez and M. Oivo, Software process measurement and related challenges in agile software development: A multiple case study, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 272–287, https://doi.org/10.1007/978-3-030-03673-7_20.

[46] I. Rus, M. Lindvall and S. S. Sinha, Knowledge management in software engineering. A DACS state-of-the-art report, IEEE Software 19(3) (2002) 26–38, https://doi.org/10.1109/MS.2002.1003450.

[47] A. Saad and S. Zainudin, A review of Project-Based Learning (PBL) and Computational Thinking (CT) in teaching and learning, Learning and Motivation 78 (2022) 101802.

[48] A. Salameh and J. Bass, Influential factors of aligning Spotify squads in mission-critical and offshore projects – A longitudinal embedded case study, in: Product-Focused Software Process Improvement (PROFES 2018), ed. by M. Kuhrmann et al., LNCS 11271, Springer, Cham, 2018, 199–215, https://doi.org/10.1007/978-3-030-03673-7_15.

[49] A. Schmitt, S. Theobald and P. Diebold, Comparison of agile maturity models, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 661–671, https://doi.org/10.1007/978-3-030-35333-9_52.

[50] A. Serenko and N. Bontis, Meta-review of knowledge management and intellectual capital literature: Citation impact and research productivity rankings, Knowl. Process Manag. 11(3) (2004) 185–198.

[51] I. Signoretti et al., Success and failure factors for adopting a combined approach: A case study of two software development teams, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 125–141, https://doi.org/10.1007/978-3-030-64148-1_8.

[52] J. P. Steghöfer, E. Knauss, J. Horkoff and R. Wohlrab, Challenges of scaled agile for safety-critical systems, in: Product-Focused Software Process Improvement

(PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 350–366, https://doi.org/10.1007/978-3-030-35333-9_26.

[53] V. Stray, B. Memon and L. Paruch, A systematic literature review on agile coaching and the role of the agile coach, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 3–19, https://doi.org/10.1007/978-3-030-64148-1_1.

[54] S. Theobald, N. Prenner, A. Krieg and K. Schneider, Agile leadership and agile management on organisational level – A systematic literature review, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 20–36, https://doi.org/10.1007/978-3-030-64148-1_2.

[55] V. Torres, M. Gil and V. Pelechano, Software knowledge representation to understand software systems, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS, vol, 11915, Springer, Cham, 2019, 137–144 https://doi.org/10.1007/978-3-030-35333-9_10.

[56] S. Trieflinger, J. Münch, S. Wagner, D. Lang and B. Roling, A transformation model for excelling in product roadmapping in dynamic and uncertain market environments, in: Product-Focused Software Process Improvement (PROFES 2021), ed. by L. Ardito et al., LNCS 13126, Springer, Cham, 2021, 136–151, https://doi.org/10.1007/978-3-030-91452-3_9.

[57] M. Usman, M. Felderer, M. Unterkalmsteiner, E. Klotins, D. Mendez and E. Alégroth, Compliance requirements in large-scale software development: An industrial case study, in: Product-Focused Software Process Improvement (PROFES 2020), ed. by M. Morisio et al., LNCS 12562, Springer, Cham, 2020, 385–401, https://doi.org/10.1007/978-3-030-64148-1_24.

[58] K. Venkitachalam and P. Busch, Tacit knowledge: review and possible research directions, J. Knowl. Manag. 16(2) (2012) 356–371.

[59] Sh. Wang and R. A. Noe, Knowledge sharing: A review and directions for future research, Hum. Resour. Manag. Rev. 20 (2010) 115–131.

[60] M. Winterer, C. Salomon, G. Buchgeher, M. Zehethofer and A. Derntl, Establishing a user-centered design process for human-machine interfaces: Threats to success, in: Product-Focused Software Process Improvement (PROFES 2019), ed. by X. Franch et al., LNCS 11915, Springer, Cham, 2019, 89–102, https://doi.org/10.1007/978-3-030-35333-9_6.

Received on December 20, 2023
Accepted on March 8, 2024

Dilyan Georgiev

Faculty of Mathematics and Informatics
Sofia University “St. Kliment Ohridski”
5, James Bourchier Blvd.
1164 Sofia
BULGARIA

E-mail: diljang@fmi.uni-sofia.bg