Suggested Citation: Carroll, N. and Ajimati, M. (2024). Unleashing the Power of Hyperagility in Citizen Development. Citizen Development Lab: Technical Report, University of Galway.

Unleashing the Power of Hyperagility in Citizen Development

Noel Carroll [0000-0001-8344-1220] Matthew Ajimati [0000-0002-3511-0766]

University of Galway, Galway, Ireland
Noel.Carroll@UniversityOfGalway.ie
Matthew.Ajimati@UniversityOfGalway.ie

Abstract. In today's rapidly changing business landscape, the demand for fast and adaptive software solutions has become paramount. Citizen development and the rise of low-code/no-code platforms have revolutionized the way software is developed, allowing non-technical individuals to participate actively in the software creation process. This article delves into the intersection of citizen development and low-code/no-code platforms, highlighting how they introduce the need for hyperagility in modern organizations. Drawing upon the literature and case studies, we argue that the rapid pace of technological advancements and market changes necessitates a paradigm shift towards hyperagile software development. We propose the *Hyperagile Citizen Development Model* (HCDM) as a response to this need, exploring its key principles and benefits in fostering organizational hyperagility and represents a revolutionary strategy aimed at effectively tackling the immediate requirement for flexibility and adaptability.

Keywords: Hyperagility, Software Development, Citizen Development, Hyperagile Citizen Development Model, Low-Code/No-Code

1 Introduction

The digital era has brought about unprecedented opportunities and challenges, requiring organizations to continuously adapt and innovate to stay competitive [1]. Traditional software development methods often fall short in addressing the increasing demand for agility, speed, and flexibility in delivering software solutions. Citizen development, an approach which allows business users to participate in delivering digital solutions, and the rise of low-code/no-code platforms have emerged as potential game-changers in this regard [2]–[6]. Specifically, the nature of low-code/no-code capabilities now introduce a focus on hyperagility [6]–[9]. Hyperagility refers to the integration of intelligent automation into an organization's agile processes, transforming it into a highly adaptable and intelligent agile organization [10], [11]. By leveraging emerging technological trends (for example, artificial intelligence), at the forefront of the hype cycle, businesses can strategize and ensure they maintain a competitive edge [10]–[12].

Most major corporations operate within frameworks dating back to the 1800s. These structures typically follow a hierarchical approach, involving numerous approvals and

concentrating decision-making power exclusively within higher levels of management. These conventional organizational frameworks were designed with the intention of thriving in a relatively stable environment. Initially introduced by IBM in the 1980s, the Rapid Application Development Model (RAD) constitutes a form of incremental process model characterized by remarkably brief development cycles. This model comes into play when a solid comprehension of requirements is achieved, and a component-based construction approach is adopted. The RAD model encompasses distinct phases: (i) *Requirements Gathering*, (ii) *Analysis and Planning*, (iii) *Design*, (iv) *Build*, and ultimately, (v) *Deployment*. A salient aspect of this model involves its reliance on robust development tools and techniques [13], [14]. The viability of employing the RAD model is contingent upon a project's feasibility to be disintegrated into smaller, manageable modules, each of which can be assigned autonomously to separate teams. Following individual module development, these components can be harmoniously integrated to culminate in the final product. The development of each module adheres to fundamental steps akin to those in the waterfall model – encompassing analysis, design, coding, and testing procedures, as depicted in the illustration. A noteworthy attribute of this model is its short timeframe, typically ranging from 60 to 90 days, known as the time-box, ensuring prompt delivery [13], [15]. However, in the present day, the economic landscape is marked by turbulence, uncertainty, and complexity. These models were initially believed to be enduring, but in our fast-paced 21st century world characterized by swift change, the need for adaptability and faster time-to-market prompts a reevaluation of these traditional operational methods. Hyperagile projects are specifically developed to swiftly respond to shifts in business, economics, and technology. They hinge on trust and responsibility, empowering employees to undertake initiatives that drive innovation and rapid implementation [8], [11], [12]. Therefore, we need to explore the nature of hyperagile models and its transformation towards greater agility.

2 Hyperagility

Today's organizations must possess the agility to pivot swiftly and responsively in order to seize opportunities and uphold their standing in the market. This necessitates the ability to embrace change ahead of competitors and to adopt operational models that facilitate such adaptability. The optimal approach for an organization to acclimate to intricacy and volatility involves cultivating qualities of flexibility, innovation, responsiveness, and agility [10]–[12]. To differentiate organizational agility from the agility solely within project management, the term hyperagile is employed. Hyperagility corresponds to the overall organization in the same way that agility corresponds to project management. The attainment of genuine organizational agility frequently poses a significant challenge for large-scale enterprises. Consider two prevalent project execution methodologies: the waterfall approach and the agile approach. Waterfall emphasizes intensive planning and often compartmentalizes functional execution. In contrast, contemporary agile methodologies dissolve these barriers and foster active involvement among organizational stakeholders. However, this agile mindset typically remains confined to this level [12].

Table 1. Key Differences between RAD and Citizen Development.

Key Differences	Rapid Application Development	Citizen Development
Focus	RAD focuses on streamlining the entire software development lifecycle by emphasizing rapid prototyping and iterative development. The goal is to quickly develop a functional prototype and refine it through successive iterations to align with user requirements.	Citizen development focuses on enabling non-technical individuals (often business users) to create simple applications using low-code or no-code development platforms. The emphasis is on empowering domain experts to address specific business needs without deep technical expertise.
Role of Developers	In RAD, experienced developers play a crucial role in designing and building the initial prototype. They collaborate closely with end-users to ensure that the prototype accurately captures requirements and functionality.	In citizen development, non-developers, often referred to as "citizen developers," take the lead in creating applications. These individuals might have a basic understanding of technology but lack formal programming skills.
Iteration and Prototyping	RAD involves multiple iterations of prototyping, where each iteration refines and builds upon the previous version. User feedback is essential to shaping subsequent iterations and achieving a final product that meets user expectations.	While iterations are possible in citizen development, the emphasis is often on creating functional applications with minimal iterations. Rapid turnaround is a key aspect, and changes are typically made as needed based on user feedback.
Complexity	RAD is suitable for projects of varying complexity, including larger and more intricate applications that require extensive user involvement and multiple iterations.	Citizen development is best suited for relatively simple applications and tasks that can be automated without the need for deep technical skills.
Technical Skills	RAD assumes a certain level of technical expertise among the development team. Professional programmers and developers are typically responsible for creating the initial prototype and driving the development process.	Citizen development targets individuals with limited technical skills. Low-code/no-code platforms provide visual interfaces and pre-built components that simplify the application development process.
Governance	RAD projects are usually managed by experienced project managers and follow established software development methodologies. There is a structured approach to project planning, execution, and monitoring.	While citizen development empowers business users, organizations still need to provide oversight to ensure data security, compliance, and the overall quality of applications created by non-technical individuals.
Timeframe	RAD aims to deliver functional prototypes and subsequent versions quickly, but the overall timeline can vary based on project complexity and the number of iterations required (e.g., months).	Citizen development emphasizes rapid creation and deployment of applications to address immediate business needs, often within much shorter timeframes (e.g., hours) compared to traditional development.

Hyperagility seeks to extend the reach of an agile mindset beyond the confines of individual product teams or projects, integrating it into the very essence of the

organization and develop low-code/no-code solutions in hours or days rather than months (using RAD) (see Table 1) [7], [11], [16]. This is the essence of hyperagility. This article aims to elucidate the relationship between citizen development, low-code/no-code platforms, and the need for hyperagility in modern organizations, culminating in the proposal of the *Hyperagile Citizen Development Model* (HCDM) as a strategic response.

Both the RAD model and citizen development are approaches aimed at accelerating the software development process and involving non-developers in creating applications. However, they have distinct differences in their focus, processes, and roles. Table 1 outlines of the key differences between RAD and citizen development: In summary, while both RAD and citizen development share the goal of accelerating software creation, they target different user groups and have distinct processes, roles, and levels of technical involvement. RAD focuses on professional development teams creating prototypes with extensive user involvement, while citizen development empowers non-developers to create simple applications using low-code/no-code platforms.

2 The Rise of Citizen Development

Over the past twenty years, significant transformations have occurred in the field of information technology. There has also been a notable increase in digital business strategies [17], which are reliant on digital innovation across various industries [18]. Furthermore, the role of IT has evolved, with its widespread presence leading to a reversal in its fundamental nature [19], and there have been substantial shifts in how IT and business professionals collaborate [20]. These foundational shifts have coincided with the emergence of novel solutions, approaches, and tools, as well as significant changes in the methods employed for designing information systems.

While contemporary principles in software engineering like agile, DevOps, and lean methodologies have incorporated users more actively within the software development processes, the convergence of IT and business units into self-sufficient product teams and the growing significance of low-code and no-code platforms have given rise to novel ecosystems in software engineering [3], [8], [21], [22] and leading to a rise in citizen development.

Digital transformation has become a core priority for CEOs in recent years. But just as the pandemic accelerated the need for change through digital transformation, it laid bare the massive global shortage of skilled software developers needed to deliver and operationalize this transformation [9], [23], [24]. Against this backdrop, we are witnessing a new method of delivering low-code development to accelerate and expand digital transformation called citizen development. This method hides the sophistication and complexity of coding but empowers subject matter experts to design, develop, and deploy applications into production as though they were full-on, experienced coders.

Citizen development refers to the practice of non-IT professionals, often from business units, creating software applications using low-code/no-code platforms. Low-code platforms enable the visual assembly of applications using pre-built components, while no-code platforms allow the creation of applications without writing any code. These platforms empower individuals with domain expertise to actively participate in software development, reducing the burden on IT departments and accelerating solution delivery[5], [8], [9].

2.1 Empowerment and Democratization

Citizen development offers numerous benefits, including accelerated innovation, reduced backlog, and enhanced cross-functional collaboration [5], [6], [25]. The democratization of software development empowers employees who understand the business processes intimately to create applications tailored to their needs. This approach fosters a sense of ownership and drives innovation from the bottom up to align with the top management innovation trajectory towards delivering superior value for the organization in terms of operational excellence and business performance. Through citizen development, democratized process to software development does not only promote agile, flexible, and adaptable patterning of work [2], [3], [5], but also motivates employees to efficiently share knowledge and mentor one another without formal obligations [26]. This, in turn, helps employees to gain higher problem-solving capability that help facilitates smart and swift decision-making process solving problems and minimizing potential risks associated with software development, in addition to creating a more resilient teams and organizations [2], [5], [6].

2.2 Digital and Software Development Skills

The production of software requires a mix of technical skills and soft skills of employees. Technical skills relate to the operational use of technologies such as generative artificial intelligence (AI) and its functional attributes such as analyzing data, creating an intuitive application, working on cloud environment, and utilizing sensors and low-code interfaces for developing business applications [27]–[29]. Soft skills are non-technical by nature and they facilitate the design, development and deployment of technological solutions through creative potential of individuals. This includes the ability to plan and communicate the deployment of technology solutions, demonstrate critical thinking and logical reasoning in comprehending and presenting information and solutions to business problems, such as designing and developing usable software products [23], [27], [30]. Although the application of technical skills is often widely considered as a fundamental knowledge requirement in writing codes or engage in programming in traditional software development method. As technical skills are often concentrated in the hands of professional software developers, the shortage of professional software developers in IT firms created gap for organizations to rapidly develop and deliver product solutions for business success [5], [31]. Citizen development closes this gap by encouraging business professionals to develop digital and software related skills with minimal training, time, effort, and significantly reduced cost to efficiently and effectively leverage technical functionality embedded in low-code/no-code development

platforms for business solutions[6], [29]. Citizen development empowers business professionals to explore and derive benefits from both digital and non-digital related software skills to handle various development tasks [8].

2.3 Knowledge Integration and Development

Adopting a hyperagile mindset also promote simultaneous integration of business and technical knowledge for software development. The opportunity for business professionals (nontechnical employees) to access and combine the functional capabilities of low-code/no-code development platforms with their business knowledge in an impromptu nature towards continuous development and deployment of business solutions without relying on IT professionals bring to bear the significance of having hyperagile structural approach to software development in organizations [29], [32]. For example, the design of low-code/no-code development platforms' graphical interfaces and application layers are simple and user friendly, easier to customize solutions and integrate with other external applications, and are scalable and maintainable [2], [24], [29], [33]. This is critical for organizations in this fast-paced technological revolution and increasing demand for greater productivity in operational processes, as well as in rapidly scaling and deploying new products and services for customer satisfactions. Indeed, this type of knowledge recombination is what is needed to reduce the gap between business and IT departments, hence rapid alignment through simultaneous integration of business and technical knowledge help in facilitating and developing new ideas into prototypes, and in swiftly testing them for operational and commercial solutions [6], [32], [34].

2.4 Generative AI and Citizen Development

Emerging as widely adopted solutions, generative AI tools such as Bard, ChatGPT, and CoPilot have swiftly garnered immense popularity. These tools hold the promise of significantly enhancing productivity within the area of software engineering. Researchers are beginning to explore the use of AI to develop applications and utilization of generative AI within the software industry. Research focus is gaining momentum on exploring what is the potential and the impact of generative AI on software productivity? What are potential benefits and pitfalls and how should they be considered? [35].

With the advent of large language models (LLMs), machine learning (ML) and AI have become readily available to developers and to non-technical groups. Applications that seem to be developed in minutes, and even software projects that were once considered nearly impossible for large software organizations with massive research and development budgets, have suddenly become not only achievable. The momentum of AI-driven development began in 2021, saw rapid growth in 2022, and reached its highest point in early 2023. The pace of progress has accelerated thanks to the increasing number of LLM providers, such as Google, OpenAI, Cohere, and Anthropic, as well as the availability of developer tools such as ChromaDB and LangChain. Concurrently, the rise of natural language interfaces for code generation has made the process of

software development accessible to a wider audience than ever before which is now primed for citizen developmement.

Generative AI technology has existed for years, but hesitations due to lack of validation prevented its wide adoption. The sudden rush for innovation often ignores risks, even for well-intentioned tools. ChatGPT's 2022 release led to an unprecedented AI arms race, amassing 100 million users in just two months. For developers, seeking help from StackOverflow or Google is routine. Generative AI stands apart by creating answers rather than retrieving them. Trained on vast datasets and human feedback, it evolves via reinforcement learning to provide accurate and safe responses. It introduces "*prompt engineering*" to shape questions effectively, maintaining contextual relevance. Generative AI relies on large language models (LLMs) with a transformer architecture and an attention mechanism, a pivotal concept in computer science [35]. In the future, generative AI could reshape software engineering roles, with most companies adopting AI-augmented strategies within the next few years and will place greater demands for hyperagility. For example, a recent study has started to highlight the potential benefits of generative AI in no-code software development context [36]. It will drive content creation and software development, demanding new skills such as refining automatically generated code and understanding non-deterministic behaviors. While tools such as ChatGPT can produce code, developers still require a foundation to execute it. Organizations such as Replit support users to quickly generate a development environment (known as a Repl) in various programming languages or frameworks, cloud services, integrated editor, and AI collaborators that actively assists developers in debugging and deploying applications within secure and scalable cloud environments.

3 Methodology

To collect data on the emerging phenomenon of hyperagility, interviewees were selected through our industrial research projects on citizen development and low-code/no-code. We reviewed three case studies (using 75 interviews) whereby organizations adopted low-code/no-code over the past two years and we identified a shift towards hyperagility and the desire to experiment with emerging technologies such as AI. In doing so, we identified ten key factors which influenced how we present the HCDM. We extract these details from case studies across a variety of organizations (multinationals and SMEs) and extract key insights on the benefits and challenges associated with their citizen development initiatives. In doing so, we identified the shift from traditional agile approaches to a hyperagile approach which shaped the development of HCDM. To ensure anonymity and to focus on the broader concepts from the interview, interviewees and organizational details are omitted. Initially, in this research, interview scripts were developed using a set of warm-up and general digital transformation questions followed by deep-dive questions on the use of low-code/no-code, and some future-oriented questions on the changing nature of software development, along with various follow-up probing questions. The interview timing ranged between 65-80 minutes, and all the interviews were conducted and recorded online using MS Teams.

These recordings were then transcribed to obtain the data. Figure 1 presents an overview of the research model and key stages with our analysis of interview data across case studies.

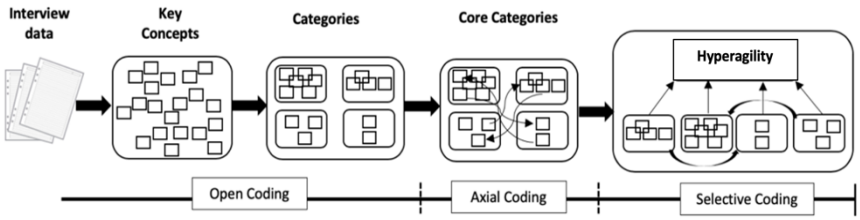

Figure 1. Overview of Research Method

Fig. 1. Overview of Research Method.

4 Hyperagile Software Development Model

Our research findings indicate the need for a HCDM to pace more emphasizes on rapid and flexible development practices than traditional agile methodologies. We position this model within a citizen development context with the aim of identifying key factors of hyperagility which are listed as follows:

1. **Minimalistic Processes**: advocates for even lighter processes and ceremonies compared to traditional agile methodologies. This requires an effort to minimize or eliminate the number meetings, artifacts, and roles that are present in existing agile frameworks.
2. **Continuous delivery and deployment**: The HCDM places a strong emphasis on continuous delivery and deployment, aiming to deliver software updates to users as frequently as possible, potentially even multiple times within a single day.
3. **Extreme Customer-Centricity**: The HCDM prioritizes developing a business case trough immediate and direct customer feedback, possibly involving customers in the development process on a real-time basis to ensure that the software meets their specific needs and delivers value.
4. **Micro-Increments and Iterations**: The HCDM encourages developers to work in very small iterations, focusing on delivering increments of functionality in a matter of hours or days.
5. **Fusion Team Structures**: Hyperagile teams should be more fluid in their composition, allowing team members to frequently switch roles, collaborate across different projects, or self-organize in novel ways to deliver a solution. Fusion teams are functionally focused teams driving to deliver digital transformation.
6. **Rapid Experimentation**: The HCDM promotes a culture of experimentation, where new ideas, features, or changes are quickly implemented and tested to gather data and immediate insights for decision-making.

7. **Real-Time Metrics and Analytics**: The HCDM involves real-time tracking of various metrics, such as user engagement, performance, and usage patterns. These metrics also guide development decisions on an ongoing basis.
8. **Automated Everything**: The HCDM relies on automation not only for testing and deployment but also for tasks such as requirement gathering, documentation, and code generation seeking ways to improve on quality and speed (e.g., using new AI capabilities).
9. **Adaptive Tools and Technologies**: The HCDM encourages teams to rapidly adopt and experiment with new tools, frameworks, and technologies that align with the projects needs and emerging contemporary technologies such as generative AI, etc.
10. **Continuous Learning and Improvement**: Similar to agile, the HCDM fosters a culture of continuous learning and improvement, where teams regularly reflect on their processes and practices to identify areas for enhancement. This also supports teams to improve the overall quality, team cohesion, and speed of hyeragile software development.

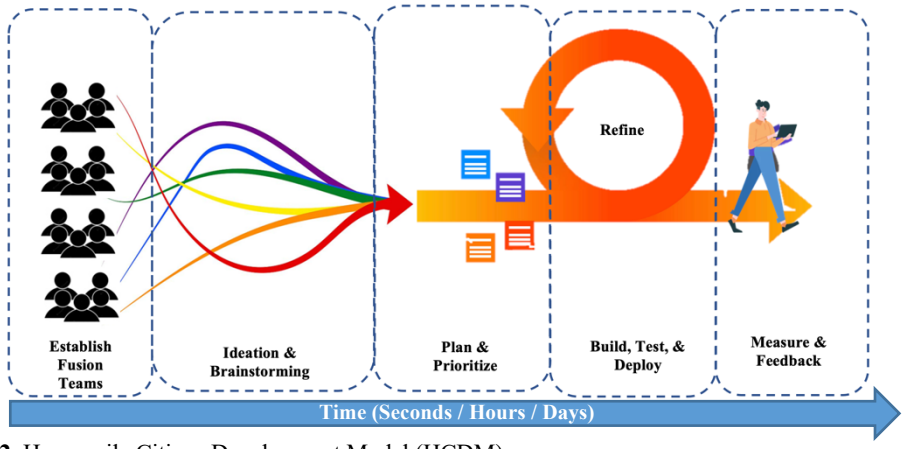

Fig. 2. Hyperagile Citizen Development Model (HCDM).

The HCDM presents five key stages which reflect the hyperagile nature of citizen development using low-code/no-code platform within an extremely short timeframe (i.e., hours or days, depending on the complexity of the project). We summarize these as follows:

1. **Establish fusion teams**: organizations need to establish multidisciplinary teams to fuse technology or analytics and business domain expertise and shares accountability for citizen development initiatives.
2. **Ideation & Brainstorming**: identify a guiding star for innovation and seek as many ideas as possible on how to solve the problem from different perspective. Ideation is closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. Ideation should be considered to be an individual pursuit (e.g., addressing a specific 'pain point'), while brainstorming is almost always a group activity (e.g., addressing a problem to drive business value for the organization).

3. **Plan & Prioritize**: Determine what the software solution will achieve and the reality of what a minimum viable product (MVP) should do. The software solution should align with a specific value statement or blueprint and develop a MVP that validates the concept of the software solution with the lowest form of risk.
4. **Build, Test, & Deploy**: Low-code/no-code platforms support organizations to design, create, test, and deploy software solutions multiple times faster and with fewer resources. This allows teams to streamline processes by providing automated testing capabilities and tools for managing deployments across different environments. The MVP provides a version of a product that possesses just enough features to provide value to users while also collecting feedback from customers for future iterations to refine the solution.
5. **Measure & Feedback**: Measuring performance and progress with well-defined metrics is a key step to achieving success with low-code/no-code solutions to ensure that a solution delivers value (e.g., usage, risk, return on investment, return on experience, etc.). Prior to investing significant resources in an organizational-wide project, low-code/no-code allows citizen developers to get feedback from customers by showcasing easy-to-build prototypes. This shifts the go/no-go decision earlier in the project schedule, minimizing risk and cost to a specific division within an organization.

5 Discussion

In the contemporary, rapidly evolving sphere of agile and digital business ecosystems, the need for expeditious and flexible software solutions has assumed paramount significance. The emergence of citizen development with the expansion of low-code/no-code platforms has instigated a paradigmatic transformation in the software development paradigm. We outline the implications for both research, implications for practice, and a future research agenda below.

5.1 Implications for Research

Our research and the adoption of the HCDM reflects an extremely fast-paced and adaptive approach to software development in contemporary business. Researchers studying this model would need to consider several implications:

1. **Rapidly Evolving Frameworks**: The HCDM, does by its nature, involve constantly changing tools, frameworks, and methodologies. One clear implication of hyperagility for researchers is the need to keep up with these changes to understand their impact on the software development process.
2. **Data Collection Challenges**: With the speed at which development occurs, collecting and analyzing data for research purposes becomes challenging. Researchers should consider developing real-time or near-real-time data collection mechanisms to examine the HDSM.
3. **Continuous Experimentation**: In the HDSM, software teams may experiment with different development practices continuously. Researchers may need to adapt their methodologies to accommodate this fluidity, potentially conducting more frequent, smaller-scale studies or ‘study sprints’.

4. **Human-Centric Focus**: The HDSM prioritizes rapid response to user feedback and frequent iterations. Researchers should focus on human-centered aspects such as user experience, usability, and customer satisfaction in their studies.
5. **Cross-Disciplinary Collaboration**: Given the pace and adaptability of the HDSM, researchers may need to collaborate with experts from various fields, including psychology, sociology, and human-computer interaction, to understand the holistic implications of this model.

5.2 Implications for Practice

The HCDM implies an even more accelerated and flexible approach than traditional agile methods. We list five implications for its practical application:

1. **Extreme Flexibility and Responsiveness**: The HCDM requires teams to be extremely responsive to changing requirements and user feedback. Practitioners need to be prepared to pivot quickly, even more so than in traditional agile approaches.
2. **Rapid Prototyping and Experimentation**: within the HCDM, teams may engage in rapid prototyping and experimentation to validate ideas and features quickly. Practitioners must be comfortable with a high degree of uncertainty and a willingness to iterate rapidly.
3. **Continuous Integration and Deployment**: Automation of testing, integration, and deployment processes becomes even more critical in the HCDM. Teams must invest heavily in continuous integration and continuous deployment pipelines to ensure that changes can be pushed to production swiftly and reliably.
4. **Enhanced Cross-Functional Collaboration**: The HCDM necessitates even closer collaboration among cross-functional teams, including developers, designers, testers, and product managers. Practitioners should foster a culture of open communication and collaboration to ensure success.
5. **Resource Allocation Challenges**: With the fast pace of development and frequent changes in priorities, resource allocation can be challenging. Practitioners must continuously assess resource needs and allocate them accordingly to support the most critical and dynamic areas of development.

5.3 Future Research Agenda

We propose five key areas for a future research agenda on the HCDM a number of disciplines including spanning software engineering, information systems, and management research:

1. **Adaptive Hyperagile Methodologies**: Additional research should investigate the development of the HCDM that can seamlessly integrate with varying project scopes, team sizes, and industry domains. Additional research could focus on creating frameworks that dynamically adjust hyperagile practices to ensure consistent delivery of value while accommodating changing requirements.
2. **Hyperautomation and AI in Hyperagile Development**: Additional research should explore the integration of hyperautomation and AI in the HCDM and processes. This could involve automating repetitive tasks, predicting bottlenecks, optimizing resource allocation, and leveraging AI to enhance decision-making in real-time, thereby improving development speed and quality.

3. **Metrics and Performance Evaluation**: Additional research should develop novel metrics and evaluation approaches specific to the HCDM. This could involve designing measures that capture the efficiency of rapid iterations, the effectiveness of continuous feedback loops, and the impact of quick deployments on overall project success. Additionally, research could investigate how these metrics align with traditional performance indicators.
4. **Hyperagile Team Dynamics and Collaboration**: Additional research should study the socio-technical aspects of the HCDM and hyperagile teams, including communication patterns, collaboration strategies, and the role of leadership. Research can delve into team composition, skill diversity, remote collaboration tools, and the challenges of maintaining cohesion and motivation within high-speed development environments.
5. **Risk Management and Governance in Hyperagile Environments**: Additional research should examine strategies for effective risk management and governance within the HCDM. Research could investigate how organizations can balance the need for rapid innovation with compliance, security, and ethical considerations. This area could explore tools, frameworks, and best practices to ensure responsible development practices.
6. **Sustaining Development Processes and Solutions**: Additional research should explore how the rapid design, development and deployment of new application tools influence our knowledge of hyperagile thinking and problem-solving capability to create and sustain new IT solutions and their processes within HCDM. This is because it is possible that different developers adopting hyperagile thinking may rapidly work on new or existing business problems to drive new IT solutions, such as developing new and expanding on the utility of software application tools and solutions. This because it may be difficult to track the pace of changes relating to the making of a tool, ways to operationalize and correctly utilize it by others for practice. This inability to document and track such information in rapid development and deployment of IT solutions can limit its usage into the future. Research should look into understanding the sustainability of the development process of IT tools and solutions in citizen development practices.

These research areas also present the multidisciplinary nature of the HCDM, spanning software engineering, information systems, and management. By advancing knowledge in these domains, researchers can contribute to the understanding, optimization, and responsible implementation of hyperagile approaches in the software business landscape. This is particularly important when organizations gain success and grow their efforts around hyperagility and citizen development and explore the potential for large-scale transformation [37] or using advancements in GenAI for innovation processes [38].

6 Conclusion

In the dynamic landscape of today's agile and digital business environment, the imperative for quick and adaptable software solutions has reached paramount importance. The emergence of citizen development alongside the growth of low-code/no-code platforms has brought about a transformative shift in the software creation process. These platforms empower non-technical individuals to actively engage in software development at a rapid pace. This article delves into the intersection of citizen development and the utilization of low-code/no-code platforms. It underscores how this juncture ushers in the necessity for hyperagility within contemporary organizations.

By drawing insights from existing literature and three real-world case studies, we contend that the rapid pace of technological evolution and market fluctuations compels a fundamental shift in the direction of hyperagile software development. To meet this demand, we present the HCDM. This model encapsulates a set of core principles and advantages that foster hyperagility within organizations. It represents a revolutionary approach designed to effectively address the immediate call for enhanced flexibility and adaptability. This article also considers the implications of the HCDM for research and practice and outlines a future research agenda for future exploration and new research frontiers on the HCDM.

References

[1] Z. Tekic and D. Koroteev, "From disruptively digital to proudly analog: A holistic typology of digital transformation strategies," Bus Horiz, vol. 62, no. 6, pp. 683–693, 2019, doi: 10.1016/j.bushor.2019.07.002.

[2] S. S. Bhattacharyya and S. Kumar, "Study of deployment of 'low code no code' applications toward improving digitization of supply chain management," Supply Chain Management, 2021, doi: 10.1108/JSTPM-06-2021-0084.

[3] A. C. Bock and U. Frank, "Low-Code Platform," Business and Information Systems Engineering, vol. 63, no. 6, pp. 733–740, 2021.

[4] M. Op 't Land, M. R. Krouwel, and S. Gort, "Testing the Concept of the RUN-Time Adaptive Enterprise: Combining Organization and IT Agnostic Enterprise Models with Organization Implementation Variables and Low Code Technology," Enterprise Engineering Working Conference (EEWC), vol. 411, pp. 228–242, 2021, doi: 10.1007/978-3-030-74196-9_13.

[5] D. Hoogsteen and H. Borgman, "Empower the Workforce, Empower the Company? Citizen Development Adoption," Proceedings of the 55th Hawaii International Conference on System Sciences, vol. 7, pp. 4717–4726, 2022, doi: 10.24251/hicss.2022.575.

[6] D. Krejci, S. Iho, and S. Missonier, "Innovating with employees: an exploratory study of idea development on low-code development platforms," in Twenty-Ninth European Conference on Information Systems (ECIS), 2021, pp. 1–16.

[7] A. Abidin, N. Senin, and A. A. A. Manaf, “A Preliminary Study of Low-Code/No-Code Ecosystem Practices: Translating Design Student Views on Crafting Interactive Design,” NVEO Natural Volatiles & Essential Oils, vol. 8, no. 4, pp. 10244–10258, 2021, [Online]. Available: http://www.nveo.org/index.php/journal/article/view/2126

[8] N. Carroll and M. Maher, “How Shell Fueled Digital Transformation by Establishing DIY Software Development,” MIS Quarterly Executive, vol. 22, no. 2, p. 3, 2023, doi: 10.17705/2msqe.00076.

[9] N. Carroll, L. Ó Móráin, D. Garrett, and A. Jamnadass, “The Importance of Citizen Development for Digital Transformation,” Cutter Business Technology Journal, vol. 34, no. 3, pp. 5–9, 2021.

[10] A. R. Dunie, D. Miers, J. Wong, M. Kerremans, K. Iijima, and P. Vincent, “Magic Quadrant for Intelligent Business Process Management Suites,” Gartner Inc, 2015. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-66AAPG8&ct=190131&st=sb

[11] S. Paluch et al., “Stage-gate and agile development in the digital age: Promises, perils, and boundary conditions,” J Bus Res, vol. 110, pp. 495–501, Mar. 2020, doi: 10.1016/j.jbusres.2019.01.063.

[12] T. Marion, D. Dunlap, and J. Friar, “Instilling the entrepreneurial spirit in your R&D team: What large firms can learn from successful start-ups,” IEEE Trans Eng Manag, vol. 59, no. 2, pp. 323–327, May 2012, doi: 10.1109/TEM.2011.2147792.

[13] P. Beynon-Davies, C. Came, H. Mackay, and D. Tudhope, “Rapid application development (Rad): An empirical review,” European Journal of Information Systems, vol. 8, no. 3, pp. 211–232, 1999, doi: 10.1057/palgrave.ejis.3000325.

[14] A. Mishra and D. Dubey, “A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios,” International Journal of Advance Research in Computer Science and Management Studies, vol. 1, no. 5, 2013, [Online]. Available: www.ijarcsms.com

[15] R. Agarwal, J. Prasad, M. Tanniru, and J. Lynch, “Risks of Rapid Application Development,” Commun ACM, 2000.

[16] M. A. Al Alamin, S. Malakar, G. Uddin, S. Afroz, T. Bin Haider, and A. Iqbal, “An empirical study of developer discussions on low-code software development challenges,” Proceedings - 2021 IEEE/ACM 18th International Conference on Mining Software Repositories, MSR 2021, pp. 46–57, 2021, doi: 10.1109/MSR52588.2021.00018.

[17] A. Bharadwaj, O. A. El Sawy, P. A. Pavlou, and N. Venkatraman, “Digital Business Strategy: Toward a Next Generation of Insights,” MIS Quarterly, vol. 37, no. 2, pp. 471–482, 2013.

[18] G. Vial, “Understanding digital transformation: A review and a research agenda,” Journal of Strategic Information Systems, vol. 28, no. 2, pp. 118–144, 2019, doi: 10.1016/j.jsis.2019.01.003.

[19] R. L. Baskerville, M. D. Myers, and Y. Yoo, “Digital first: The ontological reversal and new challenges for information systems research,” MIS Quarterly, vol. 44, no. 2, pp. 509–523, Jun. 2020, doi: 10.25300/MISQ/2020/14418.

[20] N. Urbach et al., “The Impact of Digitalization on the IT Department,” Business and Information Systems Engineering, vol. 61, no. 1. Gabler Verlag, pp. 123–131, Feb. 12, 2019. doi: 10.1007/s12599-018-0570-0.

[21] H. Karl, D. Kundisch, F. Meyer auf der Heide, and H. Wehrheim, “A Case for a New IT Ecosystem: On-The-Fly Computing,” Business and Information Systems Engineering, vol. 62, no. 6, pp. 467–481, Dec. 2020, doi: 10.1007/s12599-019-00627-x.

[22] L. M. Maruping and S. Matook, “The Multiplex Nature of the Customer Representative Role in Agile Information Systems Development,” MIS Quarterly, vol. 44, no. 3, pp. 1411–1437, 2020, doi: 10.25300/misq/2020/12284.

[23] J. Metrolho, R. Araújo, F. Ribeiro, and N. Castela, “An Approach Using a Low-Code Platform for Retraining Professionals To ICT,” in EDULEARN, 2019, pp. 7200–7207. doi: 10.21125/edulearn.2019.1719.

[24] Z. Yan, “The Impacts of Low/No-Code Development on Digital Transformation and Software Development,” arXiv preprint arXiv, 2021.

[25] J. Pacheco, S. Garbatov, and M. Goulao, “Improving Collaboration Efficiency Between UX/UI Designers and Developers in a Low-Code Platform,” in ACM/IEEE 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS-C, 2021, pp. 138–147. doi: 10.1109/MODELS-C53483.2021.00025.

[26] L. Corral, I. Fronza, and C. Pahl, “Block-based Programming Enabling Students to Gain and Transfer Knowledge with a No-code Approach,” 22nd Annual Conference on Information Technology Education, pp. 55–56, 2021, doi: 10.1145/3450329.3478314.

[27] V. Lara-Prieto and G. E. Flores-Garza, “iWeek experience_ the innovation challenges of digital transformation in industry,” International Journal on Interactive Design and Manufacturing (IJIDeM), 2021.

[28] A. Colantoni, L. Berardinelli, and M. Wimmer, “DevOpsML: Towards modeling DevOps processes and platforms,” Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings, pp. 480–489, 2020, doi: 10.1145/3417990.3420203.

[29] K. Talesra and G. S. Nagaraja, “Low-Code Platform for Application Development,” International Journal of Applied Engineering Research, vol. 16, no. 5, pp. 346–351, 2021, doi: 10.37622/ijaer/16.5.2021.346-351.

[30] A. Elbatanony and G. Succi, “Towards the no-code era: A vision and plan for the future of software development,” BCNC 2021 - Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code, co-located with SPLASH 2021, pp. 29–35, 2021, doi: 10.1145/3486949.3486965.

[31] C. Silva, J. Vieira, J. C. Campos, R. Couto, and A. N. Ribeiro, “Development and Validation of a Descriptive Cognitive Model for Predicting Usability Issues in a Low-Code Development Platform,” Hum Factors, vol. 63, no. 6, pp. 1012–1032, 2021, doi: 10.1177/0018720820920429.

[32] S. Iho, D. Krejci, and S. Missonier, “Supporting Knowledge Integration with Low-Code Development Platforms,” in Twenty-Ninth European Conference on Information Systems (ECIS), 2021, pp. 1–16.

[33] M. Tisi et al., “Lowcomote: Training the next generation of experts in scalable low-code engineering platforms,” in 2nd International Workshop on Model-Driven Engineering for Design-Runtime Interaction in Complex Systems, and 1st Research Project Showcase Workshop co-located with Software Technologies: Applications and Foundations, 2019.

[34] Y. Wang, Y. Feng, M. Zhang, and P. Sun, “The Necessity of Low-code Engineering for Industrial Software Development: A Case Study and Reflections,” IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW, pp. 415–420, 2021, doi: 10.1109/ISSREW53611.2021.00112.

[35] A. Beheshti et al., “ProcessGPT: Transforming Business Process Management with Generative Artificial Intelligence,” in IEEE International Conference on Web Services (ICWS), May 2023. [Online]. Available: http://arxiv.org/abs/2306.01771

[36] L. Sundberg and J. Holmström, “Democratizing artificial intelligence: How no-code AI can leverage machine learning operations,” Business Horizons, Apr. 2023, doi: 10.1016/j.bushor.2023.04.003.

[37] N. Carroll, K. Conboy, and X. Wang, "From transformation to normalisation: An exploratory study of a large-scale agile transformation." Journal of Information Technology (2023): 02683962231164428.

[38] J. Holmström and N. Carroll. “How organizations can innovate with generative AI.” Business Horizons, 2024 (in press).