From Modules to Marketplaces: A Vision for Composable Capability Sharing Across Organizations

Wei-Ji Wang
Graduate Institute of Networking and Multimedia,
National Taiwan University,
Taipei, Taiwan
sckhg1367@gmail.com

***Abstract*—Assembly-oriented software architecture is transforming how modern systems are developed—shifting focus from writing code to composing business-aligned capabilities. While modular components are increasingly common, most remain confined within organizational boundaries, limiting their reuse potential. This paper envisions a capability marketplace that enables cross-organizational sharing, discovery, and composition of semantically rich, contract-governed, and deployable software modules. Unlike traditional code repositories, this marketplace promotes scalable reuse of high-level capabilities supported by trust modeling, interface verification, and lifecycle governance. We outline the conceptual architecture and key design mechanisms, and identify open challenges in trust establishment and developer workflow integration. By rethinking software engineering as ecosystem-scale composition, this vision aims to advance a more composable, collaborative, and sustainable model of modular development.**

Keywords—Composable Capabilities, Capability Marketplace, Cross-Boundary Reuse, Assembly-Oriented Development, Governance and Discoverability

I. Introduction

In recent years, modular architectures have become a prevailing design paradigm in software engineering. Development teams increasingly decompose application logic into well-defined, independently deployable modules to enhance system maintainability, adaptability, and reuse [1]–[3]. These modules typically expose standardized interfaces and operate as self-contained units. Building on this foundation, some organizations and platforms have begun encapsulating such modules as composable business capabilities, treating them as foundational elements for system assembly [4]–[7]. However, development practices centered on capability composition remain in the early stages of experimentation and are not yet widely adopted in mainstream workflows.

A more fundamental limitation lies in the confinement of module design, maintenance, and reuse within organizational boundaries [8], [9]. Despite their high potential for reuse, most capabilities are not shared across teams or enterprises due to the lack of mechanisms for cross-boundary exchange. Existing platforms—such as npm [10] and Docker Hub [11]—primarily support low-level technical artifacts such as libraries and container images, but provide limited support for semantically meaningful, deployable modules that encapsulate business logic and can be composed at the application level. As software delivery cycles accelerate and collaboration across organizations becomes more common, enabling the sharing and integration of business-level capabilities has emerged as a critical challenge.

This paper presents a vision for an open capability marketplace that enables the publication, discovery, and composition of reusable capabilities across organizational contexts. The envisioned platform facilitates the exchange of deployable, contract-governed components—such as authentication services, payment processing modules, or customer engagement functionalities—allowing developers to assemble applications by integrating existing capabilities rather than building them from scratch. This vision promotes a shift in software development from code-centric authoring toward assembly-oriented engineering, advocating for a reuse-first mindset at the ecosystem level.

In the remainder of this paper, we outline the foundational elements of this marketplace, including its architectural structure, component specification model, governance mechanisms, and trust considerations. We also identify open challenges and research directions related to compatibility validation, module discoverability, and incentive design. We hope this vision will stimulate new discussions in the software engineering community around ecosystem-scale reuse and cross-organizational capability sharing, and inspire future work that advances the theory and practice of composable, assembly-driven development.

II. Background and Motivation

A. The Evolution and Limitations of Modular Design

Modular design has long served as a foundational strategy in software engineering, enabling developers to manage complexity through well-defined, interchangeable components. By enforcing clear interface boundaries and separation of concerns, modular systems support greater maintainability, extensibility, and internal reuse. However, most real-world modularization practices are confined to the scope of individual systems or organizations, focusing on local optimization rather than broader composability [12]–[15]. The design of such modules tends to prioritize technical encapsulation over semantic clarity or contextual interoperability [16], [17]. As a result, even when modules expose functional APIs, they often lack machine-readable descriptions of integration constraints, execution assumptions,

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00317

or intended usage contexts—factors critical for reliable reuse across organizational boundaries.

B. Emerging Practices and Sharing Challenges in Composable Capability Modules

With the rise of cloud computing [18-19], microservice architectures [20-24], and platform-oriented development [25-27], some enterprises have begun exploring the packaging of business logic into reconfigurable, composable capability modules [6]. This emerging design perspective—often referred to as composable capabilities [4]—extends the idea of a module beyond callable code by incorporating semantic annotations, interface contracts, and runtime assumptions. These self-contained capabilities can be selectively composed to build flexible and efficient applications, adapted to specific functionality and environment constraints [28]. Real-world examples include reusable services for authentication, resource orchestration, and automated customer support—each functioning as semantically meaningful, high-level building blocks.

Despite their potential, such capability modules are typically confined to internal ecosystems or closed platforms. The lack of standardized description models and exchange protocols has made it difficult to share or interoperate across organizational boundaries. Moreover, due to limited semantic transparency and insufficient contextual metadata, even modules with high reuse potential are often deemed untrustworthy or incompatible by external teams. As a result, valuable capabilities are frequently re-implemented, leading to redundant effort and systemic inefficiency.

C. The Promise and Challenges of Assembly-Oriented Development

In response to the growing limitations of module reuse and the barriers to capability sharing, assembly-oriented software engineering has emerged as a promising development paradigm. This approach advocates for shifting the focus of software development from authoring new code to selecting, configuring, and assembling modular capabilities. When designed with clear semantics, well-defined contracts, and contextual alignment, modules can be reliably reused across projects, teams, and even organizations—supporting scalable composition and faster system construction.

However, the realization of assembly-oriented development presents several technical and organizational challenges. First, module descriptions and contracts must follow consistent and machine-readable formats to enable automated discovery, compatibility validation, and integration planning. Second, module dependencies and integration constraints must be explicitly declared and supported by mechanisms for dependency resolution and conflict avoidance. Third, governance mechanisms must ensure the trustworthiness and traceability of module provenance and version history to maintain stability and security in reuse scenarios. Finally, sustainable module sharing requires effective lifecycle management and community governance, including incentives for contribution, mechanisms for quality assurance, and policies for long-term maintenance and evolution.

III. Vision and Platform Architecture Design

To address the cross-organizational barriers to capability reuse [29-30] identified in Section 2, we propose a preliminary vision for a capability marketplace—a semantically-aware, trust-oriented, and governance-enabled infrastructure that facilitates assembly-oriented software development. The marketplace aims to establish a collaborative environment in which composable capability modules can be safely published, discovered, verified, and integrated across organizational boundaries. By operationalizing this model, we envision a shift in development practice from code-centric authoring to capability-centric composition, unlocking new reuse potential across teams, projects, and enterprises.

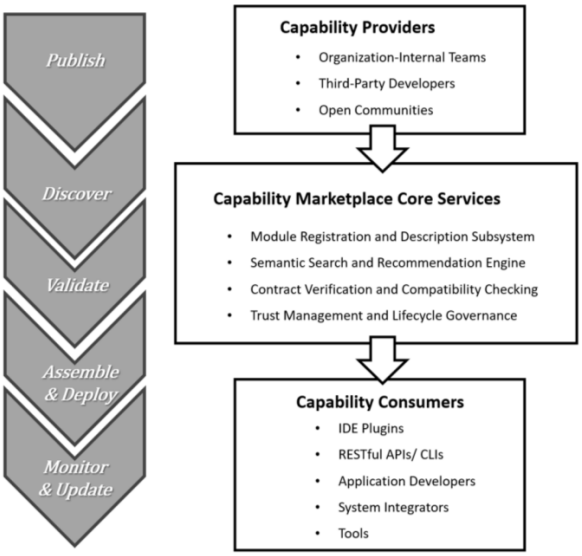

Fig. 1. Conceptual architecture of the capability marketplace for cross-organizational module publishing, discovery, and assembly.

A. Platform Positioning and Participant Roles

The capability marketplace is envisioned as a foundational layer that enables systematic module exchange and capability governance. It provides key services such as standardized metadata registration, semantic search, interface contract verification, trust assessment, and guided module assembly. The platform targets two primary roles: module providers and module consumers. Providers may include internal development teams, open-source contributors, or third-party vendors who publish deployable capability modules with structured metadata, semantic annotations, interface contracts, and version histories. Consumers, such as application developers, integration engineers, or system architects, can search for and assemble appropriate modules according to their functional and contextual requirements.

Beyond facilitating reuse, the platform also acts as a governance coordinator, maintaining lifecycle control, contribution workflows, and a reputation system. This governance layer ensures the consistency, traceability, and long-term sustainability of shared modules, while encouraging continuous maintenance and responsible participation by the developer community.

B. Core Functions and Architectural Components

The marketplace is composed of four key subsystems, each addressing a specific challenge in capability-based composition workflows:

1) Module Registration and Description Subsystem: This component enables providers to upload their modules along with structured, machine-readable descriptions using formats

such as YAML or JSON. Metadata includes functional descriptions, input/output schemas, dependencies, runtime constraints, version history, license terms, and optional service-level agreements. Standardized module descriptions provide the foundation for semantic indexing, compatibility evaluation, and automated recommendation.

2) Semantic Search and Recommendation Engine: The platform supports multi-dimensional search based on semantic tags, business domains, and functional scopes. Developers can query modules using combinations of keywords, data types, or usage contexts. The recommendation system can incorporate co-usage graphs and historical composition patterns to propose relevant module bundles or complementary components, thereby reducing search cost and improving adoption efficiency [37].

3) Contract Verification and Compatibility Checking: During module assembly, the platform automatically validates contract conformance between selected modules. This includes API schemas (e.g., REST, GraphQL, gRPC), parameter formats, and behavioral constraints. Compatibility verification prevents mismatched dependencies and ensures semantic alignment, thereby minimizing integration failures during runtime [38].

4) Trust Management and Lifecycle Governance: The platform incorporates mechanisms for assessing module trustworthiness based on provider reputation, community feedback, download activity, vulnerability scans (e.g., CVE), and SBOM compliance. Modules are annotated with maturity labels such as "experimental," "stable," or "deprecated," along with usage guidance and known risks. Upon updates or patches, the platform issues downstream notifications and compatibility reports to help consumers make informed upgrade decisions.

C. Usage Scenarios and Interaction Flow

The typical lifecycle of a capability module within the marketplace consists of five stages:

1) Publish: Providers register modules with their metadata for indexing and access.

2) Discover: Consumers search for candidate modules using semantic or structural criteria.

3) Validate: The platform checks contract compatibility and integration constraints.

4) Assemble & Deploy: Selected modules are composed and deployed to the target application environment.

5) Monitor & Update: The platform monitors module usage and propagates update alerts or compatibility warnings.

To support seamless developer workflows, the marketplace exposes a range of integration points—including RESTful APIs, command-line tools, and IDE plugins (e.g., VSCode extensions)—to enable testing, deployment, and module orchestration within familiar environments. Specifically, it provides:

- A VSCode extension for real-time semantic search, module preview, and contract compatibility suggestions.
- A command-line interface (CLI) to publish modules, inspect dependencies, validate contracts, and receive deprecation or vulnerability alerts.
- CI/CD pipeline hooks that automatically validate module compatibility, trust policies, and interface conformance during build and deployment.

These developer-facing tools reduce adoption friction by embedding composability and governance features into existing workflows. Rather than requiring paradigm shifts, the platform enhances familiar practices—making capability reuse a natural part of daily software development.

D. Feasibility and Incremental Adoption Strategy

Although the capability marketplace is currently a conceptual vision, much of its functionality can be realized using existing technologies. Module descriptors can extend standards like OpenAPI or AsyncAPI; trust mechanisms can leverage SBOM, Sigstore, or VEX; and composition validation can be supported by schema stitching, contract-based testing, or dependency graph analysis. In this way, the marketplace can act as a modular "assembly broker" that bridges existing module repositories and runtime frameworks.

We recommend an incremental adoption strategy. Initial efforts could focus on a narrow domain (e.g., identity management or customer service) to validate core features such as semantic tagging, contract verification, and lifecycle tracking. Mid-term development may include forming governance consortiums across organizations to co-develop module standards, contribution models, and joint maintenance practices. Ultimately, the marketplace aims to cultivate a sustainable, open ecosystem centered on capability reuse and trustworthy composition, thereby enhancing software development efficiency, quality, and long-term module viability.

To illustrate a concrete adoption path, we propose a domain-specific pilot focused on enterprise identity management. This prototype will include composable capability modules such as single sign-on handlers, OAuth 2.0 authorization providers, identity federation adapters, and user directory connectors. Each module will expose semantic annotations (e.g., authentication protocol types, tenant-specific settings) and explicit contract definitions (e.g., OpenID Connect flows, token formats). These modules can be semantically tagged, published, and composed into reusable authentication stacks across teams or vendors. The pilot will enable controlled validation of the marketplace's core services—including semantic discovery, contract verification, update propagation, and trust-based risk labeling—within a bounded and practical context. This implementation will serve as an empirical foundation for evaluating cross-vendor interoperability and lifecycle governance workflows, providing a low-risk entry point for gradual adoption and future expansion.

IV. Challenges and Future Directions

While the vision of a capability marketplace holds significant promise for enabling reusable and composable software across organizational boundaries, its realization entails a range of technical and organizational challenges—particularly in the areas of trust establishment [31–32] and seamless integration into existing development workflows. This section outlines two core challenge areas and identifies opportunities for future research and infrastructure development.

A. Module Trust and Governance

In cross-organizational contexts, establishing verifiable trust is a prerequisite for the adoption of external modules. Developers must assess whether a module's origin is credible, whether it is actively maintained, and whether it poses known security risks. While tools such as SBOM, Sigstore, and CVE scanning provide partial protections, there is currently no unified trust framework that integrates identity verification, community reputation, maintenance history, and risk classification [33–34]. Moreover, governance concerns—such as naming conflicts, versioning authority, and duplicate module publication—remain largely unresolved, impeding long-term module sustainability.

Future work could focus on designing standardized trust indicators that combine technical and social signals. For instance, modules could be labeled with maturity levels (e.g., experimental, stable, deprecated) and accompanied by automatically generated risk profiles derived from usage analytics and security scans [35–36]. On the governance side, further research is needed to develop cross-organizational collaboration protocols that define contribution rights, maintenance responsibilities, and quality review procedures. Such frameworks may include contributor reputation models, shared maintenance agreements, and governance consortia tasked with managing platform-wide standards and policies. Such collaborative governance frameworks not only mitigate fragmentation risks, but also enable a shared responsibility model essential for long-term ecosystem viability.

B. Integration into Development Workflows

Even with robust marketplace functionality, adoption will remain limited unless capability modules can be smoothly incorporated into developers' existing workflows. Most modern development environments—such as IDEs, CI/CD systems [39-40], and deployment frameworks—lack native support for composable module discovery, contract validation, and composition planning. Without semantic search, parameter guidance, or context-aware recommendations, the integration of reusable modules remains a high-friction task.

This highlights the need for composability-aware developer tools. For example, IDE extensions could offer real-time suggestions for compatible modules based on semantic tags or contract signatures, while lightweight domain-specific languages could be used to specify module assembly flows that are automatically converted into executable deployment pipelines. Additionally, incorporating semantic autocompletion and contextual filtering into the module discovery process could dramatically improve usability and lower the barrier to entry for modular composition. These tooling enhancements will be essential to making capability reuse not just possible, but natural in day-to-day development.

C. Toward Scalable and Sustainable Capability Reuse

While existing tools like Backstage.io [41] and Bit.dev [42] enable internal module cataloging and reuse, they primarily operate within organizational silos and lack support for semantic interoperability, trust assessment, or contract-based validation across organizations. In contrast, our proposed marketplace emphasizes ecosystem-wide reuse by incorporating machine-readable module semantics, automated contract compatibility checking, and governance-aware lifecycle management. Furthermore, our platform is explicitly designed for cross-enterprise scenarios—including third-party contributors, open module ecosystems, and shared governance models—which are not addressed by current internal developer platforms.

Addressing these limitations requires more than architectural design—it also calls for recognizing additional factors that influence scalability. One challenge is the difficulty of defining standards that accommodate heterogeneous domains. Because organizations differ in regulatory, semantic, and technical requirements, a single universal contract is unlikely to succeed. A pragmatic path forward is to establish a minimal core schema for interoperability while enabling domain-specific extensions coordinated by governance groups, thereby balancing consistency with flexibility.

Another consideration involves potential adoption barriers. Commercial companies, for instance, may be cautious about sharing modules that embed proprietary business value. This concern does not preclude participation but highlights the need for mechanisms that align openness with organizational interests. Options such as selective disclosure of APIs and hybrid participation models can help encourage broader engagement. Together with advances in trust modeling, governance design, and developer support, such measures can reinforce assembly-oriented development as a sustainable and collaborative practice, ultimately unlocking the potential of composable capabilities across teams, organizations, and ecosystems.

V. Conclusion

This paper presents a forward-looking vision for a capability marketplace—an open, semantically enriched platform designed to support the sharing, discovery, and composition of modular software capabilities across organizational boundaries. The proposed approach aims to shift software development from a code-centric model toward assembly-oriented engineering, where reusable modules are governed not only by technical interoperability, but also by semantic clarity, explicit interface contracts, and verifiable trust. By integrating structured metadata, contract verification mechanisms, and developer-centric tooling, the marketplace aspires to enable safe, scalable, and context-aware reuse of high-level capabilities.

We have outlined the conceptual architecture of the marketplace, including its core components such as module description, semantic search, contract validation, trust modeling, and lifecycle governance. In addition, we have identified two critical areas of open challenge. First, building robust models of trust and governance is essential for enabling sustainable capability reuse across organizational boundaries. Second, ensuring seamless integration of composable modules into everyday development workflows will be key to developer adoption. These areas point to rich opportunities for future research in formal contract design, socio-technical platform governance, composability-aware development environments, and ecosystem-level sustainability.

As systems become more distributed and ecosystems more collaborative, composing and sharing trusted, business-aligned capabilities will be key to scaling innovation, accelerating delivery, and sustaining maintainability. Though the capability marketplace is still conceptual, we see it as a timely step toward rethinking modularity, reuse, and cross-boundary collaboration. We encourage the community to refine and extend this vision—toward a more composable, trustworthy, and sustainable future for software development.

REFERENCES

[1] Dano, E. B. (2019, October). Importance of reuse and modularity in system architecture. In *2019 International Symposium on Systems Engineering (ISSE)* (pp. 1-8). IEEE.

[2] Mesa, J. A., Esparragoza, I., & Maury, H. (2020). Modular architecture principles–MAPs: a key factor in the development of sustainable open architecture products. *International Journal of Sustainable Engineering*, *13*(2), 108-122.

[3] Liew, H., Grubb, D., Wright, J., Schmidt, C., Krzysztofowicz, N., Izraelevitz, A., ... & Nikolić, B. (2022, July). Hammer: a modular and reusable physical design flow tool. In *Proceedings of the 59th ACM/IEEE Design Automation Conference* (pp. 1335-1338).

[4] Natis, Y., Gaughan, D., O'Neill, M., Lheureux, B., & Pezzini, M. (2019). Innovation Insight for Packaged Business Capabilities and Their Role in the Future Composable Enterprise. Gartner Research. Available at: https://www.gartner.com/en/documents/3976170

[5] Gartner. (2021). Future of Applications: Delivering the Composable Enterprise ID: G00465932 (pp. 3–4).

[6] Wang, W. J. (2025). Achieving Business Scalability With Composable Enterprise Architecture. *IEEE Access*.

[7] Ćorić, I., & Mabić, C. M. (2023). Is composable enterprise the key to digital transformation. In *6th international scientific conference on digital economy DIEC, Tuzla, BiH*.

[8] Beohar, A. (2023). *How Can Composable Commerce Benefit Small Businesses?* Composable.Com. https://composable.com/insights/composable-commerce-what-is-it-and-how-can-it-benefit-small-businesses

[9] Wang, W. J. (2025, June). Platform-Centric Agile Transformation: Bridging Business Modularity with Federated Data Governance. In 2025 IEEE International Conference on Computation, Big-Data and Engineering (ICCBE). IEEE.

[10] npm, Inc. (n.d.). *npm*. Retrieved May 12, 2025, from https://www.npmjs.com

[11] Docker Inc. (n.d.). *Docker Hub*. Retrieved May 12, 2025, from https://hub.docker.com

[12] Vescovi, R., Ginsburg, T., Hippe, K., Ozgulbas, D., Stone, C., Stroka, A., ... & Foster, I. (2023). Towards a modular architecture for science factories. *Digital Discovery*, *2*(6), 1980-1998.

[13] Fioraldi, A., Maier, D. C., Zhang, D., & Balzarotti, D. (2022, November). Libafl: A framework to build modular and reusable fuzzers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1051-1065).

[14] Fursin, G. (2021). Collective knowledge: organizing research projects as a database of reusable components and portable workflows with common interfaces. *Philosophical Transactions of the Royal Society A*, *379*(2197), 20200211.

[15] Chamari, L., Petrova, E., & Pauwels, P. (2023). An end-to-end implementation of a service-oriented architecture for data-driven smart buildings. *Ieee Access*, *11*, 117261-117281.

[16] Monetti, F. M., & Maffei, A. (2024). Towards the definition of assembly-oriented modular product architectures: a systematic review. *Research in Engineering Design*, *35*(2), 137-169.

[17] Monetti, F. M. (2025). Evaluating an assembly-and disassembly-oriented expansion of Modular Function Deployment through a workshop-based assessment. *arXiv preprint arXiv:2505.01762*.

[18] Khan, H. U., Ali, F., & Nazir, S. (2024). Systematic analysis of software development in cloud computing perceptions. *Journal of Software: Evolution and Process*, *36*(2), e2485.

[19] Guo, R., Tafti, A., & Subramanyam, R. (2025). Internal IT modularity, firm size, and adoption of cloud computing. *Electronic Commerce Research*, *25*(1), 319-348.

[20] Hasan, M. H., Osman, M. H., Novia, I. A., & Muhammad, M. S. (2023). From monolith to microservice: measuring architecture maintainability. *International Journal of Advanced Computer Science and Applications*, *14*(5).

[21] Guo, D., Wang, W., Zeng, G., & Wei, Z. (2016, March). Microservices architecture based cloudware deployment platform for service computing. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)* (pp. 358-363). IEEE.

[22] Calderón-Gómez, H., Mendoza-Pitti, L., Vargas-Lombardo, M., Gómez-Pulido, J. M., Rodríguez-Puyol, D., Sencion, G., & Polo-Luque, M. L. (2021). Evaluating service-oriented and microservice architecture patterns to deploy ehealth applications in cloud computing environment. *Applied Sciences*, *11*(10), 4350.

[23] Di Francesco, P., Malavolta, I., & Lago, P. (2017, April). Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International conference on software architecture (ICSA)* (pp. 21-30). IEEE.

[24] Gouigoux, J. P., & Tamzalit, D. (2017, April). From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. In *2017 IEEE international conference on software architecture workshops (ICSAW)* (pp. 62-65). IEEE.

[25] Bayer, F. (2024). How Metamodeling Concepts Improve Internal Developer Platforms and Cloud Platforms to Foster Business Agility. In *Metamodeling: Applications and Trajectories to the Future: Essays in Honor of Dimitris Karagiannis* (pp. 1-18). Cham: Springer Nature Switzerland.

[26] Aune, A. A. W. (2024). *Towards Enhanced Developer Experience: An Empirical Study on Successful Adoption of Internal Developer Platforms* (Master's thesis, NTNU).

[27] Riehle, D., Capraro, M., Kips, D., & Horn, L. (2016). Inner source in platform-based product engineering. *IEEE Transactions on Software Engineering*, *42*(12), 1162-1177.

[28] Wang, W. J., Lain, W. D., & Wang, Y. S. (2024, August). Enhancing Virtual Assistant Service in Data Centers with Packaged Business Capability and Post-Quantum Security Measure. In 2024 7th International Conference on Knowledge Innovation and Invention (ICKII).

[29] Reinhartz-Berger, I. (2024). Challenges in software model reuse: cross application domain vs. cross modeling paradigm. *Empirical Software Engineering*, *29*(1), 16.

[30] Balci, O., Ball, G. L., Morse, K. L., Page, E., Petty, M. D., Tolk, A., & Veautour, S. N. (2017). Model reuse, composition, and adaptation. In *Research Challenges in Modeling and Simulation for Engineering Complex Systems* (pp. 87-115). Cham: Springer International Publishing.

[31] Bengtsson, A., Nielsen, P., & Li, M. (2021, November). Component trustworthiness in an enterprise software platform ecosystem. In *Norsk IKT-konferanse for forskning og utdanning* (No. 2).

[32] Giedrimas, V. (2020, October). The role of blockchain for increase trust on software components and services. In *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)* (pp. 1-4). IEEE.

[33] Ma, Y., Gao, X., & Zhou, W. (2022, August). The trustworthiness measurement model of component-based software based on combination weight. In *International Conference on AI Logic and Applications* (pp. 270-285). Singapore: Springer Nature Singapore.

[34] Lu, Z., Delaney, D. T., & Lillis, D. (2023). A survey on microservices trust models for open systems. *IEEE access*, *11*, 28840-28855.

[35] Gkortzis, A., Feitosa, D., & Spinellis, D. (2019, June). A double-edged sword? Software reuse and potential security vulnerabilities. In *International Conference on Software and Systems Reuse* (pp. 187-203). Cham: Springer International Publishing.

[36] Yarygina, T., & Bagge, A. H. (2018, March). Overcoming security challenges in microservice architectures. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)* (pp. 11-20). IEEE.

[37] Kikas, R., Gousios, G., Dumas, M., & Pfahl, D. (2017, May). Structure and evolution of package dependency networks. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (pp. 102-112). IEEE.

[38] Zimmermann, T., & Nagappan, N. (2008, May). Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th international conference on Software engineering* (pp. 531-540).

[39] Rostami Mazrae, P., Mens, T., Golzadeh, M., & Decan, A. (2023). On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering*, *28*(2), 52.

[40] Badampudi, D., Usman, M., & Chen, X. (2025). Large scale reuse of microservices using CI/CD and InnerSource practices-a case study. *Empirical Software Engineering*, *30*(2), 41.

[41] Spotify. (n.d.). *Backstage*. Retrieved June 5, 2025, from https://backstage.io

[42] Bit. (n.d.). *Bit.dev*. Retrieved June 5, 2025, from https://bit.dev