Developing a Go-to-Market Navigator for IC Product Strategy

Maximilian Tyrchan
Stuttgart Media University
Stuttgart, Germany

Abstract – *Emerging markets driven by the mobility transition, including electric vehicles, autonomous driving, and robotics, are opening opportunities for market-oriented product strategies. This study conducted in Cooperation with Robert Bosch GmbH investigates factors hindering market-oriented product strategies in the semiconductor industry. Neither the problem nor its causes could be precisely articulated at the outset and were only progressively structured through an iterative, design-oriented approach. The study identified four hindering factors and developed a Decision Support System to improve traceability between market requirements, architectural solutions, and key performance indicators. Evaluation results confirm that the system improves the efficiency of product development kickoffs, increases transparency, and facilitates component reuse, though process-related changes remain a limitation. Beyond that, the work transformed an initially unstructured problem space into a structured problem for all involved stakeholders.*

Keywords - *Decision Support System (DSS), Design Science Research (DSR), Design Thinking, IP-Reuse, Application Specific Standard Product (ASSP), Application Specific Integrated Circuit (ASIC)*

I. INTRODUCTION

The semiconductor industry has historically been influenced by market cyclicality, structural shifts in application domains and advancements in technology. While established market segments have been clearly defined for decades, emerging trends driven by the mobility transition, such as electric vehicles, autonomous driving, and robotics, are opening up new markets where market positioning is not yet finalized. This development presents semiconductor companies with the opportunity to complement traditional ASIC development, with market-oriented product strategies based on ASSP [1]. While an ASIC approach passes on development costs entirely to a single client, ASSPs allow the investment to be recouped across multiple customers and market segments [2]. This reduces dependence on individual clients and makes it easier to offset declines in demand in specific segments. Furthermore, IP-reuse makes this strategy economically viable by amortizing development costs across several customers and segments [2]. Nevertheless, this strategic shift has far-reaching consequences for a semiconductor company. Traditional ASIC development relies on a complete specification of requirements at project start, defined in close coordination with a single client [1]. A market-oriented approach, by contrast, requires the incorporation of volatile market requirements from diverse customers and segments [1]. In addition, both the required time-to-market and the ability to assess which products should and can be offered, presuppose that interdisciplinary decisions are made efficiently, transparently, and on a solid foundation. Consequently, the focus shifts further toward the early phase of product development, during which strategic product decisions must be made under conditions of uncertainty. Yet approximately 85% of all semiconductor projects miss their original schedule [3] and only around 30% achieve first silicon success [4], indicating structural deficits in precisely this early decision phase. The problem thus lies not in the fundamental ability to develop ASSPs, but rather in the way in which they currently emerge under prevailing conditions.

In collaboration with Robert Bosch GmbH, this study investigates factors in the early stages of the product development that hinder a more efficient execution and how this situation can be improved. At the outset of this research, however, neither the problem itself nor its root causes were known, rendering the problem space a wicked problem in the sense of Rittel and Webber [5], characterized by low agreement among stakeholders and low certainty about cause-and-effect relationships. Only through an iterative, design-oriented approach could the problem be progressively structured and its causes uncovered.

Methodologically, this study follows the Design Science Research (DSR) Methodology according to Johannesson and Perjons [6], with Design Thinking applied as a user-centered design and problem-solving practice within the DSRM activities. Based on this approach, the study addresses the following research questions:

- **RQ1:** What challenges exist in the early stages of IC product development that hinder a more efficient development of ASSPs?
- **RQ2:** What requirements must a Decision Support System (DSS) fulfill to address these challenges?
- **RQ3:** How can a software system be designed to operationalize the identified requirements as a DSS?
- **RQ4:** To what extent does the developed DSS meet the defined requirements, and how do the involved stakeholders assess its value?

The contribution of this work is twofold:

- **First**, a DSS was developed as a Minimum Viable Product (MVP) that consolidates the relationships between market requirements, IC architecture, and key performance indicators (KPIs) in a graph-based data structure, enabling data-driven product strategy decisions.
- **Second**, beyond the artefact itself, this work transformed an initially unstructured and diffuse problem space into a shared, structured, and actionable problem for all involved stakeholders. While the addressed problem remains a wicked problem by nature [5].

II. METHOD

This study is situated within the design-oriented research paradigm, which focuses on the construction and evaluation of artefacts to solve practical problems [7]. Within this paradigm, the study follows the DSR-Methodology according to Johannesson and Perjons [6]. The framework comprises five core activities consisting of *Explicate Problem*, *Define Requirements*, *Design and Develop Artefact*, *Demonstrate Artefact*, and *Evaluate Artefact*. Although the activities are presented sequentially, they are logically connected through input-output relationships and are carried out iteratively [6].

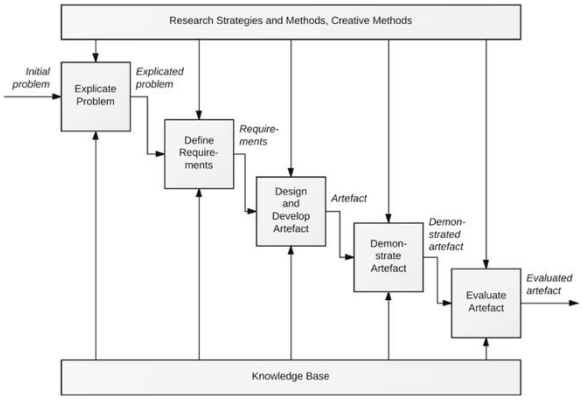

Figure 1: DSR Methodology Framework [6]

Design Thinking was applied as problem-solving practice with a focus on a user-centered design within the DSRM activities. While it is not positioned as a research methodology with claims to reliability and validity [8], it rather describes how the research activities were practically carried out. Notably, this explorative, user-centered approach did not only produce the artefact itself but also progressively structured the problem space, transforming it from a diffuse, barely articulable state into a shared and manageable problem for all involved stakeholders.

For problem identification, requirements gathering, ideation, and formative evaluation, unstructured explorative expert interviews were conducted with relevant stakeholders, including a system architect, a product area responsible, domain experts, and project managers. The open interview format was chosen to minimize the risk of overlooking unrecognized aspects due to an overly narrow methodological framework [9]. Detailed interview notes served as the primary data source, following the rapid techniques approach validated by Halcomb and Davidson [10], which offers a viable alternative to full transcription when resources are limited [11], [12]. The collected data was analyzed using *Qualitative Content Analysis* according to Mayring [13]. An approach that emphasizes the systematic interpretation of textual data to identify and extract underlying themes or patterns [13]. This method facilitates a systematic analysis of text-based material while maintaining the methodological rigor required by DSR [7]. The analysis followed a combined *deductive-inductive* coding approach. While the deductive categories were derived from the DSR activities and the defined requirements, the inductive categories emerged from the material itself. The summarizing content analysis technique was applied to reduce, paraphrase, and generalize the central statements from the interviews [13].

For the summative evaluation, the study adopted a qualitative multi-methods approach combining

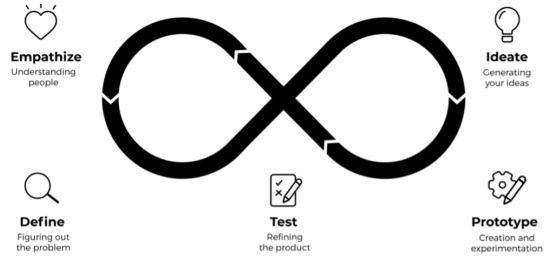

Figure 2: Design Thinking [9] [modified]

observation and semi-structured interviews. Four stakeholders participated in individual evaluation sessions. Each session began with an observation phase (approximately 45 minutes) in which the stakeholder used the artefact independently within their work context, followed by a semi-structured interview. The observation addressed the verification of the functional requirements, while the interviews captured subjective assessments of the artefact's contribution to problem resolution, its perceived utility, and its potential side effects. The data was analyzed using the same *qualitative content analysis* method by Mayring [13].

III. PROBLEM & REQUIREMENTS

A. Explicate Problem

The first DSR activity, *Explicate Problem,* aims to precisely define the problem and to specify its underlying sources [6]. For this case, the problem a strategic shift described in the introduction. While the ASIC development process relies on a strictly sequential development process in which specifications must be defined early on [14]. Market-oriented approaches, on the other hand, are based on the volatile market requirements of a wide variety of customers and segments [15]. These uncertainties not only come with a higher level of risk during development but also require faster decision-making in order to gain a time-to-market advantage over competitors.

These increased demands on the go-to-market strategy require that interdisciplinary decisions be made efficiently, transparently, and on a solid foundation. However, when examining the established conditions, it was found that it not yet realizing its full potential. This led to the following problem definition:

The development of market-oriented product variants (ASSPs) cannot currently be implemented efficiently enough.

The problem has a clear focus on the unattained outcome rather than on a specific cause. Furthermore, it does not imply that ASSPs cannot be developed at all under the current conditions. Rather, it addresses the manner in which they currently emerge.

It is important to highlight that at the outset of this research, neither a shared understanding of the problem nor articulated requirements for a potential solution existed among the involved stakeholders. The problem space was characterized by low agreement among stakeholders and low certainty about cause-and-effect relationships. In addition, the problem contains all characteristics of a *wicked problem*, as no definitive problem formulation is possible, no clear stopping criteria exist, and solutions cannot be evaluated in binary terms but require normative assessment [5].

To identify the underlying causes, a root cause analysis was conducted following Johannesson and Perjons [6], with the results structured along an Ishikawa diagram. The categories were derived inductively from the qualitative content analysis of the collected data. Four categories of root causes were identified shown in figure 3.

The root cause analysis reveals that the research problem is multicausal, spanning over product, process, organizational, and software factors. The central finding, however, is the missing knowledge about the effects of market requirements on architecture and KPIs, such as cost or area size. When project manager tries to derive a product variant from an existing product, they have to know which

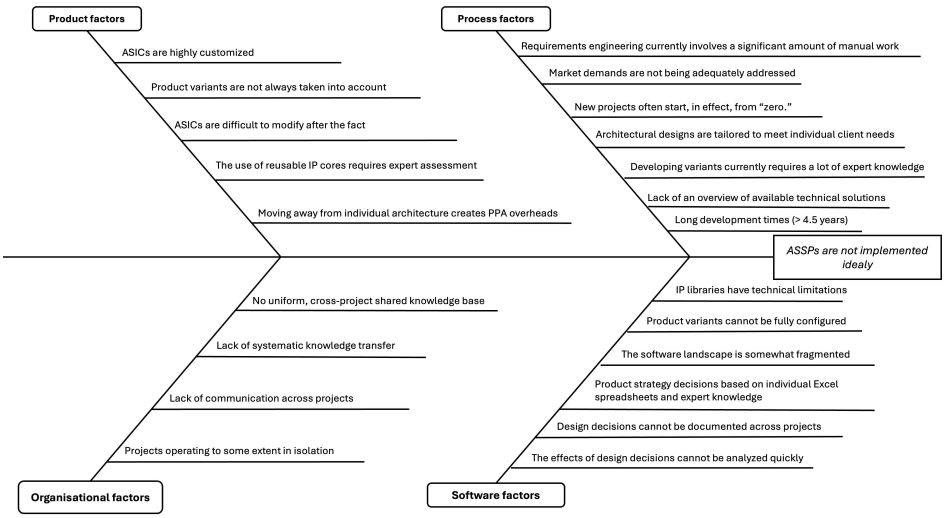

Figure 3: Ishikawa diagram illustrating the root causes

architectural changes are required and how these changes affect dependent components and relevant KPIs. Currently, this process requires requests across multiple stakeholders, based on individual calculations and implicit expert knowledge, without a traceable decision basis.

B. Define Requirements

The problems identified in the previous activity cannot be directly translated into requirements since the design process consists of partial and incremental solutions [16]. Therefore, it is necessary to define additional requirements for the artefact.

The study aims addresses the root causes identified in the problem analysis to varying degrees through a software-based artefact. Therefore, its primary contribution is to address the identified problem by explicitly connecting market requirements, architecture, and KPIs in a single system. By uncovering these multicausal relationships, the artefact enables decision-makers to evaluate technical and economic trade-offs and to make transparent, efficient, and well-founded product-strategic decisions independently. Additionally, four objectives were defined to operationalize the intended effects of the artefact. The first objective targets the increase of efficiency in early conceptual project phases by lowering the barrier to IP-reuse and enabling product conception based on existing architectures. The second objective aims to increase the transparency between technical functionality and customer value, making it visible which technical components contribute which market value. The third objective focuses on reducing the information dependencies by establishing a single source of truth. Stakeholders should be enabled to retrieve information independently and in real time, instead of relying on inefficient communication loops. The last objective addresses the acceleration of feasibility and cost estimation through simulation capabilities. By highlighting the impact of decisions on architecture and KPIs the artefact should enable faster decision-making.

In consequence twelve functional requirements were collected iteratively with the involved stakeholders and derived from the identified root causes. These requirements can be grouped into seven clusters:

- The first cluster contains the necessary essential functionalities to ensure the general usability of the software. This contains the login process and the overall management of the data.
- The second grouping covers all functionalities around the product-decision-making through a use-case dashboard, what-if analyses, and real-time KPI calculation that reveal the impact of design decisions on architecture and KPIs.
- The third set focuses on product configuration and variant creation. This includes the ability to initiate new IC's, duplicate existing products, and manage use case assignments.
- The fourth cluster concerns IP-reuse through a central, cross-project IP library with search, filtering, and integration capabilities.
- The firth set of requirements includes visual architecture modeling with a hierarchical block diagram, size-dependent component visualization, and architecture views for external customers.
- The sixth cluster covers knowledge management through documentation of design decisions, versioning and comparison to previous states.
- The last set of requirements addresses the automated requirements engineering support which was later set aside for the time being.

The non-functional requirements were defined based on the quality characteristics specified in ISO/IEC 25010:2023 [17], as these could not be directly elicited from the stakeholders. They include *usability, performance, reliability, security, scalability, maintainability, modularity*, and *compatibility*.

IV. IMPLEMENTATION

The third DSR activity *Design and Develop Artefact* is dedicated to the implementation of the software. This phase followed a systematic progression from prototype to MVP. First, a low-fidelity Excel prototype was developed to visually validate initial requirements with involved stakeholders in short, approximately two-week iterative cycles, following *Design Thinking*. The insights gained from this prototype served as the

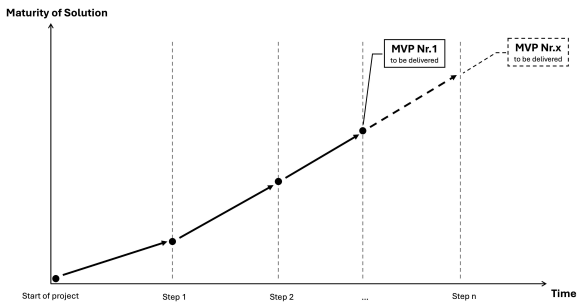

Figure 4: Development process over time

foundation for the following MVP development. Core functionalities with high contribution to problem resolution were prioritized for the prototype, while supplementary functionalities were systematically added in the MVP phase. Figure 4 highlights this development process.

A. *Architecture*

The artefact can be classified as a data-driven Decision Support System (DSS) [18]. The architecture follows a layered web application design with three logically separated layers. The frontend serves as the user interface (UI), the backend acts as the processing layer, and a graph database provides the data persistence layer. The communication between the frontend and backend is realized through REST-API, which decouples the UI from all data processing and ensures that data access occurs exclusively in the backend, as shown in figure 6.

B. *Frontend*

The frontend is implemented with Next.js in combination with React and TypeScript [19], [20], [21]. The UI is structured into five pages. A landing page lets users create and select products, as shown in figure 5. In addition, products can also be duplicated, enabling users to create new product variants based on existing products. The dashboard page provides the decision-making functionality by giving an aggregated overview of use cases and dynamically calculated KPIs. The decision support mechanism is a what-if analysis that allows stakeholders to activate or deactivate individual use cases and observe impact on KPIs such as price or area size. A separate use case page allows detailed editing of individual use cases. The workspace integrates product configuration, visual architecture modeling, and an IP library in a single view. The architecture is visualized as a block diagram in which components are displayed proportionally to their actual size. Users can add, edit, and remove systems, components, and elements, with the chip area dynamically recalculated during

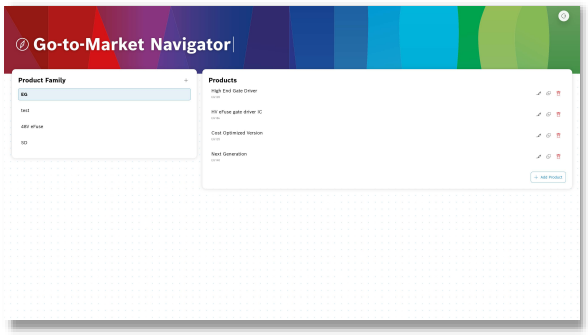

Figure 5: Landing page of the Go-to-Market Navigator

configuration. Components can be connected to represent power flows, and missing connections are flagged. The block diagram can be exported as a PDF, and an alias-based customer view can be generated for external communication. An IP library synchronization service extracts IP elements from a knowledge management platform via a REST API and persists them in the database. Users can search and filter the IP library, drag and drop IP elements into the architecture or replace existing elements with IP cores. The Documentation page supports knowledge management through commenting and versioning. This ensures that design decisions remain traceable and that knowledge is not lost during the product development process. Lastly a settings page enables users to see their profile and log out of the software.

C. *Backend*

The backend is built on Node.js with the Express.js framework and TypeScript following a multidimensional structuring approach that combines horizontal layering by technical responsibility with vertical segmentation by domain [22], [23]. An API gateway serves as the entry point, handling server initialization, route provisioning, request forwarding and middleware integration for authentication via Microsoft Entra ID using JWT Bearer Tokens according to the OAuth 2.0 standard [24], [25], [26]. Each domain-specific module provides standardized CRUD endpoints, complemented by specialized

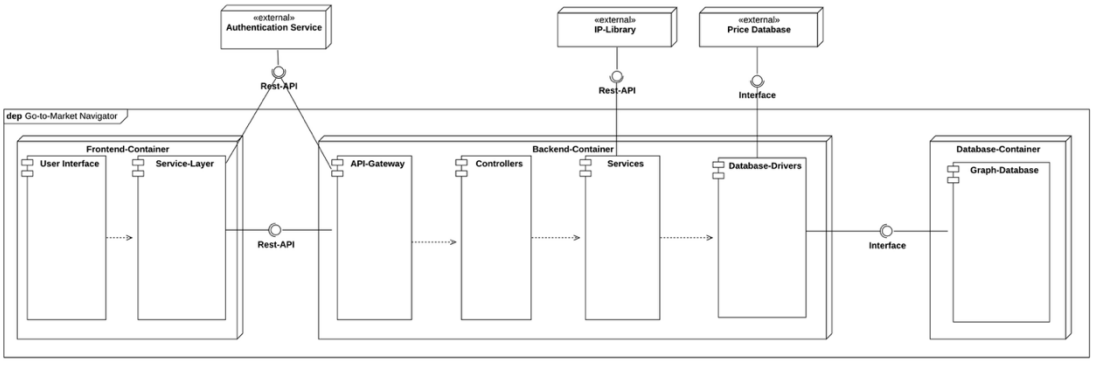

Figure 6: UML architecture diagram

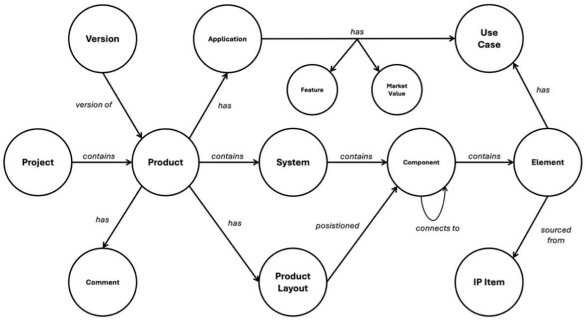

Figure 7: Knowledge graph

endpoints. The backend additionally integrates external data sources, including price calculation data via an SQL database and an IP library via a REST API interface to Confluence.

D. Database

To address the interconnected relationships between requirements, architecture, and KPIs in a scalable manner, a graph database was identified as a suitable solution. As figure 7 highlights, the data model is implemented as a property graph that explicitly encodes the previously missing relationships.

Neo4j [27] has been selected as the graph database for the artefact to be developed. This decision is based on several factors. First, Neo4j offers its own declarative query language, which enables SQL-like database queries. Furthermore, as the market leader in graph databases, Neo4j offers a certain degree of future-proofing, which is particularly relevant in the practical context of this application [28]. The graph structure consolidates the identified dimensions during this study, consisting of use cases, features, architectural components, and KPIs, into a unified data structure. This enables stakeholders to trace the impact of design decisions on various KPIs from a single use case down to an individual element of the IC.

V. Evaluation

A. Results

The summative evaluation was structured along four evaluation goals following Johannesson and Perjons [6]. The assessment examined the extent to which the artefact fulfills the defined requirements, whether and how it contributes to solving the identified problem, whether unintended side effects arose, and which improvements are needed for future development.

Overall, the research objective of developing a decision-support system to assist with product strategy decision-making in the early phase of IC development was achieved. First, it should be noted that, out of a total of 38 functional requirements, 33 were fully met, four were partially met, and one was not met, indicating that the majority of requirements were successfully implemented. The unfulfilled requirement regarding the internet accessibility of the artefact, could not be realized due to the data classification constraints.

With regard to the non-functional requirements, it must be noted that *security* and *scalability* could not been assessed during the summative evaluation. The fact that the artefact was initially developed as an MVP within a limited scope meant that these two criteria could not be validated. Consequently, reliable validation of these aspects will only be possible once the artefact is in production use. In contrast, the non-functional requirements *maintainability*, *modularity*, and *compatibility* were assessed as "*fulfilled by design*" through an argumentative-deductive approach. Although the observations revealed that the software demonstrated adequate *performance* and *usability*, partially incomplete implementations and a fundamental skepticism toward the use of new software prevented complete trust in the artefact. This reflects a well-known phenomenon associated with new software systems. An isolated malfunction can significantly undermine trust [29]. This so called *algorithm aversion* describes how errors in an algorithm lead to far greater losses of trust than those caused by human error. Trust in the artefact, however, is considered a necessary prerequisite for users to rely on the software for product strategy decision-making. This leads to the conclusion that trust in the software is more relevant to user acceptance than the mere scope of functionality.

Regarding the achievements of the objectives, the evaluation confirmed that that the artefact can overcome the inefficient "start-from-zero" approach in the design phase by lowering the barrier to IP reuse and enabling designs based on existing products. In addition, the evaluation participants confirmed an increase in transparency between technical functionality and market value. The implementation of the artefact as a single-source-of-truth to reduce information dependencies was also confirmed by the participants during the evaluation. However, the evaluation of the efficiency increase for economic feasibility and cost estimations showed a mixed result. The artefact enabled stakeholders to conduct simulations and to *"go directly into a meeting and conduct what-if scenarios with customers or managers"*. The presentation of information was described as "*very clearly prepared, making cost estimation easily readable"*. However, the reliability of the calculation results was not yet sufficient at the

time of evaluation. A stakeholder summarized this by noting that *"how good it [the software] is at that will show over the course of time*".

Another component of the evaluation was to examine whether the use of the artefacts in practice resulted in unintended or potentially harmful side effects. No side effects were identified, neither during the observations or in the subsequent interviews with stakeholders.

Regarding the possible improvements, the main criticism was directed at redundant information in the UI. In addition to these issues, limitations were also identified on the workspace page. For example, it was noted that newly created components are sorted alphabetically in an unpredictable manner, making them no longer immediately visible to users. The usability of the visualized powerflows was also rated as insufficient, as the connections do not scale proportionally with the components when their size changes. Another problem arose from the naming of terms within the artefact. Several stakeholders emphasized that the levels of abstraction between application, use case, market value, and features were ambiguous and thus inconsistent.

For expanding the artefact's functional scope, the analysis of the interviews resulted in suggestions for optimizing the dashboard, the workspace, and the initially discarded aspect of requirements engineering. Regarding the dashboard page, it was noted that additional information, such as how old processing statuses are, would be beneficial to users. Furthermore, it would be useful if comments could also be created for individual use cases to enable more granular documentation. To further simplify the initial effort involved in variant configuration, one participant suggested that not only the products but also the use cases should be duplicatable. In addition, it would also be useful if a fixed set of basic use cases were automatically created as part of this process. For the workspace, an automated compatibility check between components could proactively prevent architectural errors. Then participants expressed a desire for a search function to make components easier to find, as well as the ability to further customize their colors in the block diagram. To improve documentation, it was also suggested that changes made during product configuration be annotated. For the visual representation of components, one user also requested more inputs and outputs for power flows. To integrate additional downstream process steps, a logical next step could be the ability to convert existing use cases directly into requirements, which was rated as potentially very useful. Finally, general suggestions for improvement were identified that affect the entire artefact. In this regard, additional views could be offered, such as an edit and read-only mode.

B. Answers to the research questions

The evaluation results allow a consolidated answering of the four research questions. The root cause analysis gives answers to **RQ1** and reveals multicausal challenges spanning from product, process, organizational, over software factors. **RQ2** is being answered by the iterative elicitation of 38 functional individual requirements across five segments and eight non-functional requirements. The artefact demonstrates who a graph-based data model, combined with a layered web architecture and dedicated UI, constitutes a viable operationalization for **RQ3**. Lastly, the evaluation confirms a high fulfillment rate of functional requirements (33 of 38 fulfilled) and positive stakeholder assessments regarding transparency, IP reuse, and the establishment of a single-source-of-truth as an answer to **RQ4**.

C. Limitations

These results must be interpreted in light of several limitations. The first limitation relates to the early stage of the artefact's development. Development as an MVP is primarily aimed at demonstrating proof of concept. Consequently, the system in its current form is not yet ready for full-scale production use. Furthermore, the artefact lacks a larger dataset, without effects such as cross-project component usage cannot be conclusively investigated and the artefact cannot realize its full potential. It is therefore important to emphasize that this work can only provide a limited insight into the artefact's benefits. Furthermore, the inherent limitations of a software solution must also be taken into account. Accordingly, while the developed artefact provides an initial technical foundation for addressing the identified problems, it does not resolve the associated and overarching factors, such as a company's strategic direction or necessary procedural changes, that would additionally be required. The results of the evaluation must also be further relativized in terms of their validity. A key issue is the small sample size in the summative evaluation. Although the four participants represented four different roles and thus contributed diverse perspectives, the number is too small to allow for reliable and generalizable conclusions. Due to the small sample size, the subjective influence of individual opinions in the statements cannot be offset,

meaning that the occurrence of biases cannot be ruled out. Furthermore, the small sample size limits the theoretical depth and breadth of perspectives in the qualitative study. Therefore, a larger number of participants would have increased the validity of the findings. Additionally, there is a lack of insights into the long-term use of the artefact. This is particularly true given that the development process of an IC typically spans several years. Due to organizational constraints, the summative evaluation conducted as part of this master's thesis could only provide a snapshot. It is therefore not possible at this time to demonstrate how strategic decisions made based on the artefact will impact subsequent phases and time-to-market in the long term.

VI. CONCLUSION & FUTURE WORK

At the outset of this study the development of ASSPs in the early stage of product development were not carried out efficiently enough. When shifting towards a market-oriented product strategy the established conditions currently do not yet realize its full potential. The qualitative studies conducted as part of the DSR activities identified various causes related to four different factors (product, process, organizational, and software), with the lack of an explicit connection between market requirements, architecture, and KPIs emerging as the main cause. Based on these findings, a Decision Support System was iteratively developed as a MVP following the user-centered Design Thinking practice embedded within the DSR methodology.

The evaluation confirmed a positive contribution to problem solving and indicated a clearly recognizable added value for the involved stakeholders. The results suggest that the artefact breaks the inefficient *start-from-zero* approach by enabling product conception based on existing architectures and by lowering the barrier to IP reuse through the integration of an IP library. With enterprise-wide adoption, the artefact can establish a single source of truth that increases knowledge transfer and transparency within the organization. The central innovation of the artefact, however, lies in the integration of market requirements, architecture, and KPIs, thereby making the impact of design decisions transparent. This reduces the previously necessary inefficient coordination among stakeholders and enables decision-makers to make product-strategic decisions on a sound and transparent basis.

Beyond the artefact itself, this work achieved a second contribution that lies in the structuring of the addressed problem space. Gregor and Hevner [30] distinguish different types of knowledge contributions in DSR, among which reflexive insights from the design process count as an independent contribution. At the beginning of the study, no shared understanding of the identified problem or its underlying cause-and-effect relationships existed among the involved stakeholders. By applying the DSR in conjunction with the user-centered approach of Design Thinking, the problem space was progressively structured, and potential solutions were identified. Although the problem remains a wicked problem by its nature, the shared understanding among the participating stakeholders has fundamentally changed [5]. Starting from a state of low consensus regarding cause and effect, the problem can now be classified as more clearly defined and better understood when, for (communication purposes only) positioned in a Stacey Matrix (figure 8). The answers to the four research questions document this transformation, from an initially diffuse and barely articulable state to named and categorized causes (RQ1), agreed-upon requirements (RQ2), a functional artefact (RQ3), and validated results (RQ4).

The present findings and limitations open several possible directions for future research. This includes long-term studies to examine the artefact in a production setting in order to investigate factors such as scalability, security, and potential long-term effects. There are also further areas of interest, such as the integration of AI into decision support systems. To address the already existing skepticism towards new software, parallel research into the explainability of AI-driven recommendations will be essential. In addition, the evaluation revealed that the previously deprioritized aspect of requirements engineering retains practical relevance and should be reconsidered for integration. Finally, the identified problem cannot be resolved through software alone. Further research should investigate the processual and organizational changes necessary to complement the artefacts potential as a navigator for IC product strategy.

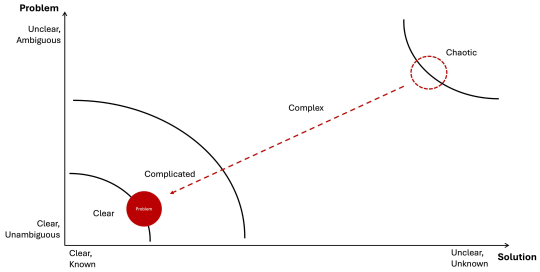

Figure 8: Stacey Matrix according to Stacey [32]

VII. REFERENCES

[1] M. J. S. Smith, *Application-specific integrated circuits*, 11. printing. Boston: Addison-Wesley, 2003.

[2] B. Lojek, *History of semiconductor engineering*. New-York: Springer, 2006.

[3] "Effort behind reusing IP blocks is underestimated." Accessed: Apr. 02, 2026. [Online]. Available: https://www.design-reuse.com/news/202514333-effort-behind-reusing-ip-blocks-is-underestimated/

[4] H. Foster, "2020 Wilson Research Group study: FPGA tend report," Siemens Digital Industries Software. Accessed: Jun. 25, 2026. [Online]. Available: https://resources.sw.siemens.com/en-US/white-paper-2020-wilson-research-group-functional-verification-study/

[5] H. W. J. Rittel and M. M. Webber, "Dilemmas in a General Theory of Planning," *Policy Sci.*, vol. 4, no. 2, pp. 155–169, 1973.

[6] P. Johannesson and E. Perjons, *An Introduction to Design Science*. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-10632-8.

[7] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," 2004.

[8] U. Johansson-Sköldberg, J. Woodilla, and M. Çetinkaya, "Design Thinking: Past, Present and Possible Futures," *Creat. Innov. Manag.*, vol. 22, no. 2, pp. 121–146, Jun. 2013, doi: 10.1111/caim.12023.

[9] Y. Zhang and B. M. Wildemuth, "Unstructured Interviews," *Appl. Soc. Res. Methods Quest. Inf. Libr. Sci.*, pp. 222–231, 2009.

[10] E. J. Halcomb and P. M. Davidson, "Is verbatim transcription of interview data always necessary?," *Appl. Nurs. Res.*, vol. 19, no. 1, pp. 38–42, Feb. 2006, doi: 10.1016/j.apnr.2005.06.001.

[11] C. Vindrola-Padros and G. A. Johnson, "Rapid Techniques in Qualitative Research: A Critical Review of the Literature," *Qual. Health Res.*, vol. 30, no. 10, pp. 1596–1604, Aug. 2020, doi: 10.1177/1049732320921835.

[12] K. Eaton, W. Stritzke, and J. Ohan, "Using Scribes in Qualitative Research as an Alternative to Transcription," *Qual. Rep.*, pp. 586–605, Mar. 2019, doi: 10.46743/2160-3715/2019.3473.

[13] P. Mayring, *Qualitative Inhaltsanalyse: Grundlagen und Techniken*, 13., Überarbeitete Auflage. Weinheim Basel: Beltz, 2022.

[14] A. Shetty, "ASIC Design Flow And Methodology – An Overview," *Int. J. Electr. Electron. Eng.*, vol. 6, no. 7, pp. 1–5, Jul. 2019, doi: 10.14445/23488379/IJEEE-V6I7P101.

[15] EETimes, "ASIC/ASSP platform speeds development," EE Times. Accessed: Feb. 16, 2026. [Online]. Available: https://www.eetimes.com/asic-assp-platform-speeds-development/

[16] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[17] "ISO/IEC 25010:2023(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model." Accessed: Jun. 12, 2026. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25010:ed-2:v1:en

[18] D. J. Power, "Types of Decision Support Systems (DSS)." Accessed: Apr. 08, 2026. [Online]. Available: https://www.gdrc.org/decision/dss-types.html

[19] "Next.js Docs | Next.js." Accessed: May 12, 2026. [Online]. Available: https://nextjs.org/docs

[20] "React Reference Overview – React." Accessed: May 12, 2026. [Online]. Available: https://react.dev/reference/react

[21] Typescriptlang, "The starting point for learning TypeScript," Typescriptlang. Accessed: May 12, 2026. [Online]. Available: https://www.typescriptlang.org/docs/

[22] "Index | Node.js v26.1.0 Documentation." Accessed: May 13, 2026. [Online]. Available: https://nodejs.org/docs/latest/api/

[23] "Express - Node.js web application framework." Accessed: May 13, 2026. [Online]. Available: https://expressjs.com/

[24] "OAuth 2.0 — OAuth." Accessed: May 13, 2026. [Online]. Available: https://oauth.net/2/

[25] "JWT Access Tokens for OAuth 2.0." Accessed: May 13, 2026. [Online]. Available: https://oauth.net/2/jwt-access-tokens/

[26] "Microsoft Entra ID (früher Azure AD) | Microsoft Security." Accessed: May 13, 2026. [Online]. Available: https://www.microsoft.com/de-de/security/business/identity-access/microsoft-entra-id

[27] "Neo4j Graph Intelligence Platform," Graph Database & Analytics. Accessed: May 14, 2026. [Online]. Available: https://neo4j.com/

[28] "DB-Engines Ranking," DB-Engines. Accessed: May 14, 2026. [Online]. Available: https://db-engines.com/en/ranking/graph+dbms

[29] B. J. Dietvorst, J. P. Simmons, and C. Massey, "Algorithm aversion: People erroneously avoid algorithms after seeing them err.," *J. Exp. Psychol. Gen.*, vol. 144, no. 1, pp. 114–126, 2015, doi: 10.1037/xge0000033.

[30] S. Gregor and A. R. Hevner, "Positioning and Presenting Design Science Research for Maximum Impact," *Manag. Inf. Syst. Q.*, vol. 37, no. 2, pp. 337–355, Jun. 2013, doi: 10.25300/MISQ/2013/37.2.01.