Multivocal Literature Review of Micro Front-end Architectures

Gamma A. Octafarras
VU Amsterdam, The Netherlands
gammaandhika@gmail.com

1 INTRODUCTION

As microservices become more prominent in the scaleable software development industry, extending the concept of the architecture to the frontend has become the approach that companies such as Allegro, HelloFresh, OpenTable, Skyscanner, DAZN and others [2, 8, 9, 11, 16, 17, 19] utilize as either their main frontend architecture or is used in some of their products.

The term micro frontend(MFE) gained traction around at 2016 [1, 11] in the industry as developers who were working on large scale projects had difficulties maintaining the multiple front end features as a single codebase and were looking for an approach that would ease the management of separate features. As they were already adopting the micro service architecture in the back end, extending the concept to the front end was seen as a possible solution to avoid a large monolithic front end. An early definition of micro frontend [1] was: '*A web application broken up by its pages and features, with each feature being owned end-to-end by a single team. Multiple techniques exist to bring the application features—some old and some new—together as a cohesive user experience, but the goal remains to allow each feature to be developed, tested and deployed independently from others.*' The main ideas of the Micro Frontend Architecture can be formulated as:

- Technology Agnostic
- Code Isolation
- Native Browser Features for Communication
- Resilient Application

The **goal** of this study is to search and analyze existing literature on MFEs, which will then be used to create a general model of how MFEs are formulated. In addition, MFEs that are currently being used in practice will also be used as a reference in conjunction to the existing literature to get a view of how the technology is being utilized by the industry. Furthermore, this study will also explore the data management aspects, and discuss the methods, benefits and drawbacks of the approaches used currently in the industry.

The **audience** of this study are researchers who are interested in recent developments of the front-end architecture field and the intricate details of how data-management and communication is handled in MFEs. And additionally software engineers that are interested in the adoption of MFE as the staple for their front end architecture.

2 RELATED WORK

Although a few studies encompassing MFEs exist in literature, there are only a few of them that focuses on the intricate aspects of MFEs itself [12, 13, 15, 21]. These studies consist of either i) A study of a specific system which uses MFEs[7, 10, 20] or ii) the migration of traditional monolithic front-ends to MFEs[6, 14]. However, since a few papers that studies on the concept of MFEs still can be found, these papers will be used in conjunction with experience reports of notable tech companies that utilize MFEs.

This study will focus on using the previously mentioned resources to create a systematic model of how MFEs are generally formulated, but will also mention the discrete approaches.

3 STUDY DESIGN

3.1 Research Approach

The goal of this study is to provide readers with the different approaches that can be done to create a MFE architecture, which includes the different ways the architecture itself can be designed and on the more intricate details such the communication and data management of MFEs. With this is mind, we have drafted the research questions as follows:

RQ1 How are Micro Frontend Architectures designed?
This research question aims to identify the approaches to designing a micro frontend architecture, we will then formulate a general model based on the approaches which will then be used as the context for the next research question.

RQ2 What are the methods used for the communication and data management between the broken-down frontends?
This research question aims to analyze the different methods of inter-communication and data management that are used in the industry. Additionally aspects such as security will also be included in the analysis.

3.2 Research Methodology

The Multivocal Literature Review (MLR)[3] process will be used in this study as Micro Frontends is a relatively recent concept and has only recently gained traction as web applications continue to grow in size. Hence there is a lack of academic research in the field, however many companies in the industry have experimented and adopted MFEs. With several publishing experience reports and their findings as *grey* literature, which makes the systematic MLR the ideal process for this specific topic. In this MLR, we will include both published papers and *grey* literature, which in this literature review will include tech blogs, experience reports and online talks(podcasts/videos) with the reasoning that the reports of these companies can be a basis for future research on the subject.

3.3 Search and Selection

We use the tool *Elsevier Scopus*[1] as the search engine to find the related literature that will be used in this study.

The initial search with the query (ref query) resulted in 11 papers. Of which only 5 papers focused on MFE as the main topic, the other 6 consisted of either case studies or migration reports. However,

VU Amsterdam, 2022, The Netherlands
2022.

[1] https://www.scopus.com/

these papers will still be used as parts still give valuable data on MFEs.

The second search was done using *Google Scholar*[2]. Relevant papers which have already been found during the initial search will be filtered out, which resulted in another 4 papers.

A third search was also conducted to find the pertaining *grey* literature, in which we aimed to find out how MFEs are being used in practice. Since most of the companies in the industry have not released any publications. Experience reports and tech blogs was the next best source of information, which is one of the main reason that the MLR process was used.

3.3.1 Exclusion/Inclusion Criteria. As MLR utilizes *grey literature* as a literature source, a robust exclusion and inclusion criteria is needed to ensure the validity of the literature review.

We used only reports that were in written in the companies official tech blogs and filtered out personal blogs and posts in developer forums as our exclusion/inclusion criteria. This search resulted in 4 experience reports and 4 tech blog posts.

3.3.2 Snowballing. Backwards snowballing was applied to both literature and grey literature to identify more relevant papers from the initial list of selected papers. The snowballing for the grey literature was done by following outgoing links, and further assessing the credibility of the link.

3.4 Data Extraction

For the literature, data was extracted by firstly reading through the abstract and then only extract relevant information from the rest of the paper via open and selective coding [18], which will then be stored in a sheet. From this combined data, we look for patterns and differences of the designs of MFEs which will then be combined by information extracted from the grey literature.

3.5 Data Synthesis

Previously extracted data from literature was iteratively analyzed following the descriptive synthesis design [3] to first create a base model of the design of MFEs. Furthermore, the initial model was compared and updated with information of real-world industry data from the grey literature to create a generalized model of MFE designs and the patterns which are used for data communication and management.

4 RESULTS

4.1 Micro Frontend Designs

As MFE is a relatively recent concept, there is no set standard practice on how it is designed. However, through the compilation of sources, we can formulate a general model of the design and the different approaches of the composition methods which will be discussed in this section.

[2]https://scholar.google.com/

4.2 General Design approaches

A general design of MFE is to have one main container, usually composed by an *orchestrator*, which houses the multiple MFE applications. The main container may also contain base shared components such as the navigation menu and titles. As one of the main goals in adopting MFE is for each app to be independent, each app is isolated from each other. As depicted in Figure 1, each application is technologically agnostic where each app can run different frameworks. This general design is what makes MFEs scale well, as each application is independent they can be developed individually without blocking other applications.

With different the separate applications built, there needs to be a system to combine the different applications into the main container. This is usually done by a composition system[21] which will be discussed in the next section.

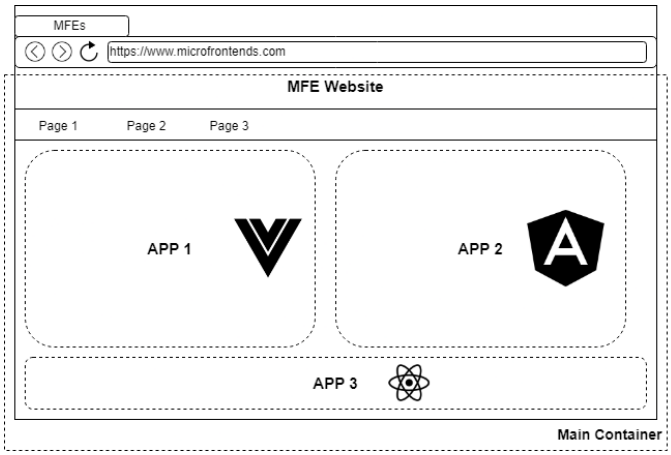

Figure 1: Basic Design of MFE

4.3 Composing methods

4.3.1 Client side. In client-side composition (Figure 2),the application shell loads Micro-Frontends inside itself in the browser. MFEs should have as an entry point a JavaScript or HTML file so that the application shell can dynamically append the DOM nodes in the case of an HTML file or initialize the JavaScript application when the entry point is a JavaScript file. [21]

Another approach that is less popular is to utilize a combination of *iframe* HTML element, with each element hosting an MFE, a method utilized by Spotify [8] in both their desktop app and website.

4.3.2 Server side. In server-side composition, composition could happen either at runtime or at compile time (Figure 3). In this case, the server is composing the view by retrieving all the different MFEs and assembling the final page in the server-side before sending it to the client. This solution is a decent option if all users see the same page. However if the pages are personalized per user, it will require serious considerations regarding the scalability of the eventual solution as there will be multiple requests to compose different pages in the server.

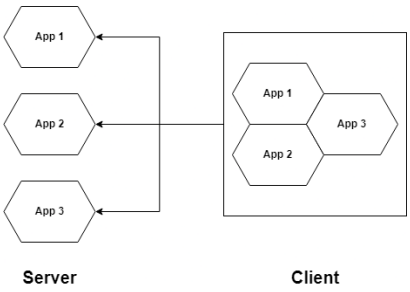

Figure 2: Client side composition

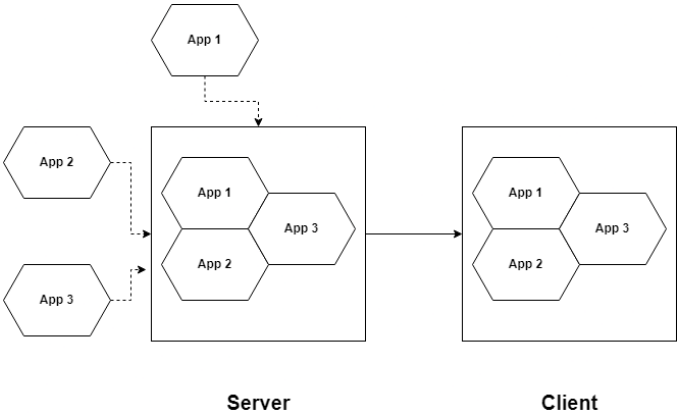

Figure 3: Server side composition

4.4 Communication and Data Management

As MFEs generally want to keep their state separated[21], unlike in traditional monolith frontend solutions, communication between components is a crucial aspect. In this section, the different approaches that can be used for the communication and data management between the components will be discussed. Although server-side solutions are possible, We will focus mainly on client-side strategies.

4.4.1 Shared Event Bus. The shared event bus (Figure 4) exposes a global API that every component can access, which includes listening and broadcasting of the events. With this approach, components need to know both the event and the actual component name itself to identify where the emitted event is being sent to. This approach can be done by using the native *EventTarget* interface in browsers but can also be done by creating a separate API layer.

An example of this approach in the industry is the one used by DAZN where they expose a client-side orchestrator with an API layer to share data between the micro-frontend components[9].

4.4.2 Direct Component-to-Component. With direct component-to-component (Figure 5), the requirements to send data is similar to the shared event bus where both component and event name is required. However, instead to sending to a dedicated event bus, the components directly send the data to each other. This approach is done by using the native browser *Window.postMessage* method.

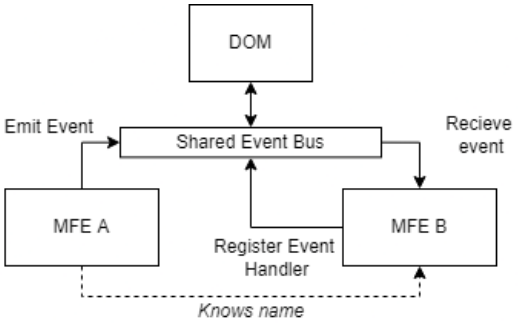

Figure 4: Shared Event Bus

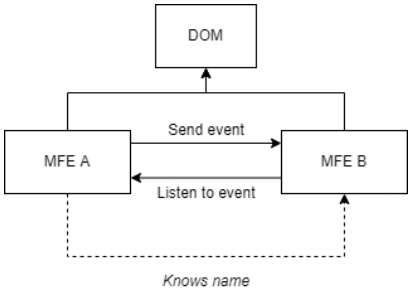

Figure 5: Direct Component-to-Component

4.4.3 Publish/Subscribe. With the publish/subscribe approach (Figure 6), separate channels are used as the bus for components to communicate with each other. The separation of channels are often based on function that the components are trying to achieve, an example would be in an e-commerce platform the *checkout* will be a channel that multiple components would subscribe to to achieve the checkout function. With this approach, components can effectively communicate with each other without having to identify each other, only the channel context is needed. This approach scales well with more complex web applications where there are multiple MFEs in a single page and having a clear distinction between channels further reduces unwanted coupling[4].

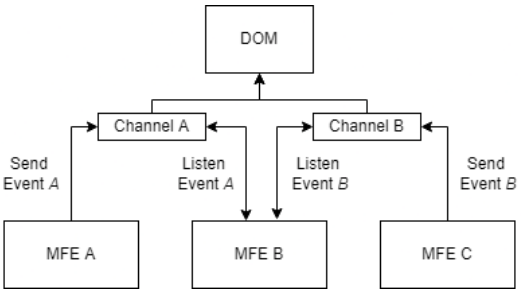

Figure 6: Publish Subscribe

4.5 Frameworks

Although most companies prefer to create their MFE design themselves to have more control, frameworks that handles most of the functionalities from basic design to data management exists.

4.5.1 OpenComponents. OpenComponents [3] is an open-source MFE framework that handles the components, composition and has its own registry to store the separate apps. Companies such as OpenTables and SkyScanner utilizes this framework.

4.5.2 Project Mosaic. Project Mosaic[4] is a combination of tools and libraries with a specification that defines how components are made and interact with each other. Project Mosaic breaks down components into *fragments* which are composed at runtime. This framework is used by Zalando as their main MFE design.

5 DISCUSSION

This multivocal literature review aimed to combine academic literature and industry knowledge in the form of grey literature to formulate a basic design on the MFE architecture, how they are built and its various building blocks to output a full-fledged website. This provides both researchers and software developers with the insights of how the technologies in the academic literature are used in industry, and if there is correlation between them. During this study, we found multiple sources on the subject but discusses the organizational aspects of adopting MFEs which we did not use as a source as it was deemed out of scope.

6 THREATS TO VALIDITY

This paper might suffer from threats related to the inaccuracy of the data extraction, a possible incomplete set of results due to limitation of the search terms, bibliographic sources and grey literature search engine, and possible subjectivity related to the definition and the application of the exclusion/inclusion criteria.

In this section, we discuss these threats and the strategies we adopted to mitigate them.

6.1 Internal validity

Possible issues in the selection process are related to the selection of search terms that could have lead to a non complete set of results. To mitigate this risk, we applied a broad search string. This was possible because of the novelty of the topic.

To overcome the limitation of the search engines, we queried the academic literature from multiple bibliographic sources, and the grey literature was queried only from Google search, however a snowballing process was applied to include more possible sources.

The possible subjectivity was not mitigated in this paper, as the inclusion/exclusion criteria was decided by the sole author and might be subject to the authors opinion and experience.

6.2 External validity

External validity is related to the generalizability of the results of this multivocal literature review. [5]

In our study we utilize both the academic and the grey literature on MFEs. However, we cannot claim to have screened all the possible literature, since some documents might have not been properly indexed and data from the industry not freely available.

6.3 Construct validity

The construct validity is concerned with the relation between theory and observation. This threat was limited by searching through multiple sources with the same general search string, which was then further filtered by the exclusion/inclusion criteria.

6.4 Conclusion validity

The conclusion validity is concerned with the degree in which our conclusions are reasonable based on our extracted data. This was mitigated by following well known multivocal and systemic literature review methods and guidelines [3, 5].

7 CONCLUSION

This paper gives an overview on approaches of how micro-frontends are designed, how the components are composed into a page and the data communication and management between each component. Furthermore, by using the Multivocal literature review method, this paper also gives an overview of how the industry incorporates MFEs into their solution.

During this study, we found that there is 2 general ways to compose components: i) client-side and ii) server-side. With the main methods of communications between the components in the client-side being: i) Shared Event Bus, ii) Direct Component-to-Component and iii) Publish/Subscribe. Major frameworks also exists, with the ones used in the industry being: i) OpenComponents and ii) Project Mosaic.

As micro-frontend is a relatively recent concept, we can conclude that although MFEs are gaining traction in the industry, there are still problems with its adoption in less complex web applications, where its heavier payload and eventual complexity can outweigh the benefits.

REFERENCES

[1] 2016. Micro frontends: Technology radar. https://www.thoughtworks.com/radar/techniques/micro-frontends

[2] Bartosz.walacik Bartosz.galek. 2016. Managing frontend in the microservices architecture. https://blog.allegro.tech/2016/03/Managing-Frontend-in-the-microservices-architecture.html

[3] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2017. Guidelines for including the grey literature and conducting multivocal literature reviews in software engineering. *CoRR* abs/1707.02553 (2017). arXiv:1707.02553 http://arxiv.org/abs/1707.02553

[4] Michael Geers. 2020. *Micro frontends in action.* Manning.

[5] Barbara Kitchenham. 2004. Procedures for Performing Systematic Reviews. *Keele, UK, Keele Univ.* 33 (08 2004).

[6] Manuel Kroiß. 2021. *From Backend to Frontend-Case study on adopting Micro Frontends from a Single Page ERP Application monolith.* Ph.D. Dissertation. Wien.

[7] Manel Mena, Antonio Corral, Luis Iribarne, and Javier Criado. 2019. A Progressive Web Application Based on Microservices Combining Geospatial Data and the Internet of Things. *IEEE Access* PP (07 2019), 1–1. https://doi.org/10.1109/ACCESS.2019.2932196

[8] Author luca mezzalira. 2019. Adopting a micro-frontends architecture. https://lucamezzalira.com/2019/04/08/adopting-a-micro-frontends-architecture/

[9] Luca Mezzalira. 2019. Adopting a micro-frontends architecture. https://medium.com/dazn-tech/adopting-a-micro-frontends-architecture-e283e6a3c4f3

[10] Nattaporn Noppadol and Yachai Limpiyakorn. 2021. *Application of Micro-frontends to Legal Search Engine Web Development.* 165–173. https://doi.org/10.1007/978-981-16-4118-3_15

[3] https://opencomponents.github.io/

[4] https://www.mosaic9.org/

[11] OpenTable. 2016. OpenComponents - microservices in the front-end world. https://tech.opentable.co.uk/blog/2016/04/27/opencomponents-microservices-in-the-front-end-world/?utm_medium=referral&utm_campaign=ZEEF&utm_source=https%3A%2F%2Fmicro-frontends.zeef.com%2Felisabeth.engel
[12] Andrei Pavlenko, Nursultan Askarbekuly, Swati Megha, and Manuel Mazzara. 2020. Micro-frontends: application of microservices to web front-ends. (06 2020).
[13] Severi Peltonen, Luca Mezzalira, and Davide Taibi. 2020. Motivations, Benefits, and Issues for Adopting Micro-Frontends: A Multivocal Literature Review. *CoRR* abs/2007.00293 (2020). arXiv:2007.00293 https://arxiv.org/abs/2007.00293
[14] Istvan Poloskei and Udo Bub. 2021. Enterprise-Level Migration to Micro Frontends in a Multi-Vendor Environment. *Acta Polytechnica Hungarica* 18 (01 2021), 7–25. https://doi.org/10.12700/APH.18.8.2021.8.1
[15] Y. R. Prajwal, Jainil Viren Parekh, and Dr. Rajashree Shettar. 2021. A Brief Review of Micro-frontends.
[16] Pepijn Senders. 2017. Front-end microservices at Hellofresh. https://engineering.hellofresh.com/front-end-microservices-at-hellofresh-23978a611b87
[17] Jan Stenberg. 2018. Experiences using micro frontends at IKEA. https://www.infoq.com/news/2018/08/experiences-micro-frontends/
[18] Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research.* Sage publications.
[19] Simon Tabor. 2021. How DAZN manages micro-frontend infrastructure. https://medium.com/dazn-tech/how-dazn-manages-micro-frontend-infrastructure-f045d7c634c2
[20] Daojiang Wang, DongMing Yang, Huan Zhou, Ye Wang, Daocheng Hong, Qiwen Dong, and Shubing Song. 2020. A Novel Application of Educational Management Information System based on Micro Frontends. *Procedia Computer Science* 176 (01 2020), 1567–1576. https://doi.org/10.1016/j.procs.2020.09.168
[21] Caifang Yang, Chuanchang Liu, and Zhiyuan Su. 2019. Research and Application of Micro Frontends. *IOP Conference Series: Materials Science and Engineering* 490 (04 2019), 062082. https://doi.org/10.1088/1757-899X/490/6/062082