Building Mobile Software Ecosystems - A Practical Approach

Steffen Hess[1(✉)], Susanne Braun[1], Johannes Feldhaus[2], Marco Hack[2], Felix Kiefer[2], Dominik Magin[1], Matthias Naab[1], Dominik Richter[1], Torsten Lenhart[1], and Marcus Trapp[1]

[1] Fraunhofer IESE, Kaiserslautern, Germany
{steffen.hess, susanne.braun, dominik.magin, matthias.naab, dominik.richter, torsten.lenhart, marcus.trapp}@iese.fraunhofer.de

[2] John Deere ETIC and ISG, Kaiserslautern, Germany
{FeldhausJohannes, HackMarco, KieferFelix}@JohnDeere.com

Abstract. Mobile apps are gaining great importance in the world of business software. Developers' intentions are to build apps that support a specific piece of functionality with great user experience, business often needs to cover a large spectrum of functionality. The results are Mobile software ecosystems (MSE), which usually consist of a large number of apps supporting a certain type of business and combine the strengths of multiple service providers. At a first glance, developing mobile software might look simple. Doing it for business and at an ecosystem scale makes it extremely challenging in practice. Initiating an MSE means to come up with an attractive set of apps that provide adequate openness so that other companies can contribute to them and increase the value of the ecosystem for customers. This paper describes an approach to build MSEs in their initial version. This approach is based on software engineering state-of-the-art practices from requirements engineering, user experience (UX) engineering, and software architecture. The paper elaborates the specifics of MSEs and describes how they can be addressed in the approach. The approach has been applied in a large-scale industrial case study in the agricultural domain in a joint project of John Deere and Fraunhofer IESE. Within that case study, lessons learned with regard to user experience and software architecture are derived and described in detail. Practitioners setting up an MSE can avoid these pitfalls by taking our lessons learned into account.

Keywords: Mobile · App ecosystem · Practical experiences · Human-computer-interaction · Software architecture · User experience

1 Introduction

Mobile apps are gaining great importance in the world of business software. Developers' intentions are to build apps that support a specific (usually small) piece of functionality with great user experience, whereas business often needs to cover a large spectrum of functionality. Thus, there is a trend to provide multiple apps, each of them providing a decent amount of functionality. Therefore, companies have to develop and maintain a large number of business apps which are interconnected and still easy to use, and which share companies' look & feel. Furthermore, companies are collaborating more and more to provide added-value products and services for their users, which neither of the involved companies could provide on their own. This is directly reflected in their business software which requires the integration and alignment of their software offerings and thus their mobile business apps. The results are mobile software ecosystems (MSE), which usually consist of a large number of apps supporting a certain type of business and combine the strengths of multiple service providers. "A software ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions." [1].

At a first glance, developing mobile software might look simple. Doing it for business and at an ecosystem scale makes it extremely challenging in practice. Initiating an MSE means to come up with an attractive set of apps that provide adequate openness so that other companies can contribute to them and increase the value of the ecosystem for customers. Providing a great and consistent user experience across apps, platforms, devices, and form factors is really challenging. MSEs do not contain only mobile apps. For business software, a tight integration with existing IT infrastructure and potentially new backend software is necessary. Many challenges in MSEs are related to the handling of data: Due to their many sensors and multiple connections (e.g., WLAN, Bluetooth, NFC), mobile devices provide a lot of data sources and communication possibilities. Providing always the best possible user experience requires an excellent understanding of the domain and creative solutions. Sharing large amounts of data can be challenging, as well as sharing data among apps on the same device (e.g. on iOS), but sharing data between apps of different providers can be especially challenging as there might be the need to restrict access on a very fine-grained level.

This is only a small excerpt of aspects that in the end determine the user experience and the success of a mobile software ecosystem. In this paper, we want to describe our approach (Sect. 2) and report our lessons learned (Sect. 3) from developing an MSE that is used as an extended enterprise resource planning system running on iOS as well as on Android as a native solution. This approach has evolved from many software engineering best-practices and was refined with the learnings of multiple projects with industrial customers.

2 Approach and Application in a Case Study

In this section, we describe our development approach for setting up an MSE. Practitioners, which plan to initiate and develop an MSE, should work on the following four questions:

- **What is my company's maturity regarding mobile app development?**
- **What is the mobility potential of my application domain?**
- **What should our customers and users experience?**
- **What is the best technical realization?**

Before starting to develop an MSE we recommend to assess your company's maturity regarding mobile app development. Therefore, in Sect. 2.1 we describe typical maturity stages we experienced in a large number of projects with industrial partners in various application domains. Our experience shows that it is hard to decide which business processes or tasks will be supported by mobile apps and even harder which ones will not be supported. To underline this assessment of the mobility potential of an application domain we describe our mPOTENTIAL method in Sect. 2.2. As mentioned before, a great user experience is crucial for mobile business apps. Due to the larger number of apps within an MSE, each single app will not get the same attention as in a single app development scenario. To assure the expected user experience we introduce our mConcAppt method in Sect. 2.3. Each application domain and systems class has its own architectural drivers that support architectural decisions. Section 2.4 therefore introduces typical architectural drivers for MSEs.

We have applied and refined our approach in numerous projects with industrial customers in several application domains. In this paper, we specifically focus on one MSE which we developed in a large-scale case study of John Deere and Fraunhofer IESE. We illustrate aspects and specifics of the approach with examples from this case study.

Key stakeholders in the agricultural domain are operators of machines and owners or managers of farms. Additionally, there are contracting companies that offer services like harvesting to farmers and own a large set of machines and employ further operators.

Key elements in an agricultural ecosystem are operators that use machines to conduct certain field or transport operations. Machines are often equipped with so-called implements that are specific to certain operations, like a harvester implement. Managers are planning and supervising the agricultural operations and are adequately assigning machines and operators to certain operations on a specific field. Large farms own hundreds of fields, machines and employ numerous operators.

While modern agricultural machines are already equipped with proprietary displays and software, there is a trend towards using standard mobile devices for further supporting applications. Similarly, existing desktop-based planning software is also accompanied by further mobile solutions for managers. Thus, mobile devices like pads and phones are gaining more and more importance.

Additionally, smart devices add further value: iBeacons can be used to track the position of equipment, and wearables like watches and glasses allow further new interactions for operators.

All mobile apps and devices need connection and integration in order to optimally support the work on a farm. Thus, at least one backend system is necessary in order to allow data transfer and orchestration. Further other sources of information can complement an agricultural MSE: existing agricultural software systems might be included and software services like one providing weather data or map data are beneficial.

2.1 Maturity Stages Towards Mobile Software Ecosystems

The typical development stages of an MSE are illustrated in Fig. 1. In **stage 1 (first app)**, companies usually start with a first app that is individually designed and delivered in a good quality. The challenges are limited to the construction of the app user interface and local data storage – usually a company look & feel is applied in a hands-on way. Often, existing backend interfaces are simply reused for the app without mobile specific adjustments to demonstrate the feasibility. With respect to stage 1 the resulting quality is mainly based on applying a sound user centered design technique that ensures that the individual requirements are met and a good user experience is provided. Regarding the backend services, the most quality-critical aspect is whether they can deliver the right data in the right time, otherwise UX is compromised.

In **stage 2 (many single apps),** companies deal with the existence of many single apps that are not necessarily connected and usually delivered by different development groups, business units, suppliers etc. This usually leads to potential inconsistencies with regard to all software engineering disciplines. Technically, the treatment of backend services is often uncoordinated and becomes harder with an increasing number of single apps.

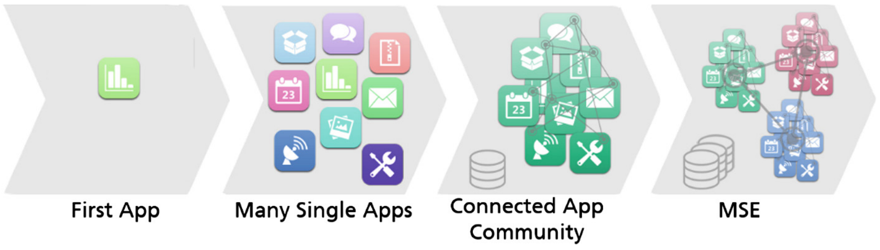

Fig. 1. Mobile software ecosystem development stages

In **stage 3 (connected app community),** the rising ecosystem gains a lot of maturity by aligning the apps with a uniform UX concept and introducing consequent connections among apps, e.g. for navigation and data exchange. Technically, this step often comes with a consolidation of backend services and their consequent alignment with the needs of the mobile apps. Mostly, this means that the backend services for mobile apps are different from those serving for other purposes, like data exchange with other systems or web interfaces.

In **stage 4 (app ecosystem)**, the state of a mobile software ecosystem is reached. It goes beyond the app community by involving different organizations contributing to the app ecosystem for added overall value. It is open towards the extension with further apps and still has a focus on adequate and consistent UX, which might also mean an intended differentiation along company boundaries of the contributors. Technically, this stage requires a well-aligned communication across company boundaries and clear concepts for data exchange, which strongly increases security demands.

In the described large-scale case study, we developed an MSE that belongs to stage 4 of the ecosystem scale.

2.2 Mobility Potential Analysis

When developing an MSE the application domain provides very individual and unique constraints. It is critical to clearly understand these specifics of the application domain and the companies involved to evaluate the mobility potential.

The described case study takes place in the agricultural domain where we are facing a scenario with a focus on online/offline capability, large amounts of precise data due to precision farming techniques and very individual user requirements. Therefore, we applied a mobility potential analysis (mPOTENTIAL [2]) that consists of the following stages:

1. The **definition of goals and constraints** defines the global goal that the MSE should fulfill based on the overall company strategy. Another major outcome of this stage is that the scope and possible cooperation partners are already drafted.
2. **Identification of relevant business areas** deals with the systematic analysis and the definition of business areas that are promising from a business perspective. The result is a prioritization that supports the evolution strategy of the MSE.
3. After that, the most promising business areas are analyzed with respect to the **involved roles and business processes**. The goal is to identify possible mobile touchpoints that could be supported by an app and provide both, business benefit and user benefit.
4. The mobility potential analysis is concluded with an **ideation phase** that evolves the intial app ideas a bit further so that app ideas are basically ready to decide if the app should be part of the MSE and to determine what kind of app we are going to build.

Those four steps described above may sound simple but as they deliver valuable results and we expect MSEs to grow and get more and more complex, we assume that it is necessary to perform them continuously in order to set a basic strategy and refine this strategy throughout the development of single apps. We recommend to document the results of each step at least roughly.

Especially in the agricultural domain, the integration of different legacy farm management system backends, different kinds of machinery and mobile devices sets the context. On the one hand, this means a lot of potential for innovation, on the other hand it puts a lot of constraints on the MSE development.

In the agricultural domain, there are some specific aspects, for example, that modern machines track data such as sensor information about soil, plant, and machine conditions with a GPS positioning system and store the data every few seconds. This huge amount of data can be distributed across several farm management systems and needs to be accessed on mobile devices in the field.

The MSE has to integrate already existing mobile apps of different categories which were developed before the MSE. While our MSE focusses mainly on enterprise resource planning, other apps are closer to the agricultural machines as sources of data.

Machine-related apps in agricultural business can be distinguished between *documentation* apps like the SeedStar Mobile app by John Deere [John Deere, 2014] and *semi-machine-controlling* apps. These app types are closely related to the machine and its data. Hence a CAN to Wi-Fi access point is needed to transmit the machine information to the app. The communication between the electrical control units (ECU) and a user interface is based on CAN bus technology. The communication flow could be realized either directly between user interface devices and CAN bus or via the backend, which of course has many architectural implications, in particular also on the achievable UX.

Table 1. MSE product philosophy

Attribute A	1	2	3	4	5	Attribute B
Innovative	▨					*Classic*
Company Experience		▨				*Native Experience*
Playful					▨	*Useful*
Context Aware		▨				*Static*
Stand-Alone				▨		*Integrated*
Smart	▨					*Sluggish*
Easy of Learning			▨			*Ease of Use*
Explaining					▨	*Intelligent*

2.3 UX-Centered Mobile App Conception Method

When developing the overall UX strategy of an MSE, it is important to follow a holistic strategy to ensure a high level of consistency across all the different platforms and devices. Therefore, the starting point of UX foundation in our case study is the definition of a so called product philosophy (see Table 1). This product philosophy supports the achievement of a consistent experience as it is used continuously throughout the app development process with different team and by different stakeholders. The product philosophy of the case study shows that there is a high emphasis on innovation and providing a very unique user experience. The product philosophy is mainly used as a communication tool to substantiate decision making during the development of an MSE.

For the UX conception of single apps within the MSE we followed the mConcAppt Approach [3]. This user centered approach (see Fig. 2) allows a lightweight conception of the app performing exactly the steps that are necessary to develop apps of the MSE and to gain information to coordinate with the architectural specification. In addition to performing the app conception, decision points and activities are taken into account, which is shown in Table 2.

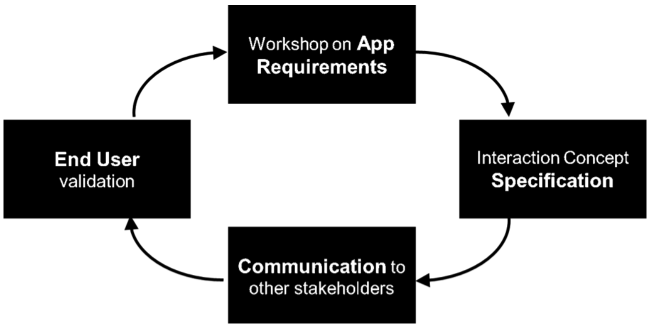
Workshop on App Requirements
Interaction Concept Specification
Communication to other stakeholders
End User validation

Fig. 2. mConcAppt overview

Table 2. Decision points that influence the UX of an MSE

Decision Points:
Target devices;
Linkage to the overall ecosystem product philosophy
Native vs. web
Elaboration of usage of artificial intelligence to support a positive UX
Service based app scoping
Possible reuse of user interface elements *Distribution of functionality between backend and client*
Innovation and creativity
Variable vs. static user requirements
User requirements with regard to data usage
Adoption of already existing apps
Usage of ecosystem crowd functionalities
Innovation degree
Usage contexts within the MSE
Consistency

2.4 Architectural Drivers

When designing the architecture of an MSE, we often identify similar architectural drivers which can be addressed in various ways by taking different architectural decisions. The best suitable decision very often depends on the characteristics and context of the respective system. At Fraunhofer IESE, we use the Architecture-centric Engineering Solutions (ACES) [5] approach to identify the architectural drivers from all relevant stakeholders and to derive appropriate decisions and views from them. This approach is in general independent from the type of system under design. However, the specific knowledge about MSEs is in the comprehension of the architecture-specifics of this system category. Thus, we outline typical drivers and decisions we regularly observe in MSEs below.

Offline Capability: One common and important decision we often encounter is how to deal with limited or interrupted network connections. While it means definitely a much better user experience to offer the complete functionality of an app in offline mode, it has to be considered that adding this feature significantly increases complexity of synchronizing data between the different components of the app ecosystem. In the end this often leads to the tradeoff of reduced maintainability and extendibility. A compromise can be to implement offline support only for specific apps or particular features of the ecosystem, but finally it depends on the importance of the offline capability driver which approach should be taken.

Distribution of Functionality: Closely related to this is also the question where to locate specific functions respectively their implementation in the system. Here again we have a tradeoff between the limited resources on a mobile device that have to be handled carefully and the non-disruptive user experience in offline mode. For example, computing-intensive processes that need significant data from different resources should clearly reside on the backend, tailored expert systems can be integrated directly into the apps.

Selection of Synchronization Technologies: A crucial part of an MSE is also a reliable, performant and easy-to-use synchronization mechanism that takes care of the distribution of data between backend, apps and other involved components. Software architects and developers typically have to decide if they want to use a third party database replication technology or if they want to implement their own solution for this. While the latter gives the opportunity to perfectly tailor the mechanism to specific needs, such a task is very complex and costly in terms. On the other hand, even if using existing solutions results in lower effort at first, it might turn out later that they cannot fulfill the specific requirements in terms of performance and scalability, which in turn means significant additional work for adapting the solution or replacing it by another one.

Design of Data Models: Another important decision to take is how to align the data models on the different components of the system (backend, iOS apps, Android apps…). On the first glance it may seem obvious that all parts of the system should work on the same data model. However, since they are based on different technology stacks this is often very hard to achieve and also has negative impact on the tools and libraries that can be used as well as the maintainability of a specific app. On the other hand having separate platform-optimized data models on each component also has negative impact on the overall maintainability of the system and might lead to issues because of incompatibilities between the different data models.

Categorization of Data: One crucial question that typically comes up is if all data should be treated in the same way or if it should be distinguished between several types of data, especially regarding their change frequency. There is a clear trend that data gets classified in some way, e.g. master data, that changes seldom, is handled differently than data related to a concrete transaction. However, there are several approaches that differ in the number of data classes and the handling of them.

Design of APIs: An additional question that needs consideration is how to organize the APIs between the different components of the system. Backend and Apps can communicate via web services based on REST or SOAP, whereby REST is clearly state-of-the-art nowadays. However, independent of the technology, it is important to build the API in a way that it is open and extendable so that new Apps or connections to other MSEs or third party components can be realized with minimal effort.

There are many more recurring aspects that come up when designing a MSE, but because of limited space, we cannot describe them in detail here. Among these are multiple version support, internationalization, push notification concepts and data validation.

The conceptual architecture of our agricultural MSE case study consists of various elements. Farmers and their operators are the potential users of the apps. They use native apps on tablet and phone devices (iOS and Android). The system requires the availability of a central backend for data management and exchange. The backend works in a multi-tenancy way to allow efficient operation for all customers with their apps on the same backend machines. The backend is connected to external data sources, like for example a weather data provider. More detailed architectural information, with a particular focus on data, on this concrete case study can be found in [Naab et al., 2015].

3 Experiences from Building Mobile Software Ecosystems

3.1 Lessons Learned on User Experience Foundation

The following lessons learned originate from our user interface conception work during the prototyping of the described MSE. The by far most challenging task was reaching consistency with regard to UX across the different platforms and devices.

Using Product Philosophy as a Communication Tool. Using the product philosophy approach was time-consuming and unusual at the beginning, as every involved stakeholder needs to adopt this method. In the end, it was very beneficial to keep track of such a baseline that supports decision making on the interface between architecture and user experience. Many existing conflicts with regard to feature prioritization realization could be solved based on the specified philosophy. Additionally, every business stakeholder could identify himself with the philosophy as this communicates the general MSE vision on a consumable format.

Template-IOriented User Interface Conception. The usage of initial app templates for user interface conception provided a high consistency across apps of the different platforms. Reuse of design elements between smartphone and tablet was also supported. In the start-up period of the MSE, a set of user interface and interaction templates have been created and continuously maintained throughout the project. This enabled also a high amount of reusable assets during development. Nevertheless being able to provide a unique experience by having those – it was very challenging to balance between following the templates and individual solutions that provide a better UX within the single app. The following decision points accompanied us throughout the project: deciding if a template needs to be adopted; deciding in an optimal solution needs to be adopted based on an existing template; decide if an individual solution is appropriate; adopt the individual solution to the consistency requirement.

Artificial Intelligence Provides a High Potential for UX Improvement. By classifying historical user data and the application of learning algorithms in the backend a MSE can provide positive UX to the end user. In a prototype we managed to realize intelligent auto completion mechanisms based on intensive data analysis. This resulted in faster and more efficient task planning of the farmer and provided a great satisfaction in the end.

Close Alignment Between UX and Architecture Provides Unexpected Benefits. Performing joint workshops including UX experts and architects to create ideas on how UX can be improved by the reasonable usage of the data that is provided by the system provides essential benefits. This joint activity produced valuable features that would not have been identified by only applying a user-centered design approach. In the case study, we could foresee many scenarios in which we precautionary saved and analyzed data and leveraged functionalities that would not have been possible with the initial specification.

Maintenance and Integration of UX Assets. Especially with larger MSEs it gets important to think early about maintenance, integration and communication of UX assets. Changes with respect to the user interface are often expensive if they need to be done in various app instances. Therefore we recommend to use asset libraries, widgets and templates from the very beginning and ensure a sound traceability between them. In our case study we benefit a lot from having mobile developers involved into the conception phases to overcome those challenges. In addition we used if available the UI-builder within the IDE (in our case XCode) for early prototyping to show interactivity and reuse created prototypes during implementation. In general, for MSE, UX design and development need to have a tighter integration than in single app development.

Integration of Smart Devices. A sound integration of wearables (glasses, watches) and iBeacons for reasonable use cases provided a huge impact on the UX of the MSE. However, the challenge is to design the right functionality on the right device, and the design space becomes even larger when different types of smart devices are accompanying the usual mobile devices. In this case, iBeacons could be used to ensure that the right operator is driving the intended machinery.

Balance Business Goals and User Goals. Comparing to mobile development – MSE have a rather long development timespan although single apps within the MSE are realized quickly. We made good experiences with a continuous involvement of various business stakeholders and continuous user feedback. This has been very promising during the strategy phase where the scope of mobile support was set. Several innovative use cases have been provided by users and internal stakeholders (e.g., marketing, product management). Negotiating the given ideas and communicating them back and forth is a key success factor of the MSE.

3.2 Lessons Learned on Technical Foundation

The following lessons learned originate from our design and implementation work in the prototyping of the described app ecosystem. The by far most challenging task was the design and implementation of data synchronization in order to support offline mode for the apps of our ecosystem. Thus, our lessons learned mainly origin from this area. Further lessons learned with a broader focus can be found in [Naab et al., 2015].

There is No One-Size-Fits-All Solution for Data Synchronization. Even though mobile networks are still massively expanded, there may never be absolute connectivity for mobile systems. Concurrent modifications of data during offline mode inevitably lead to anomalies like e.g. lost updates. However, the criticality and the handling of those anomalies highly differ between different application domains and even between different use cases, as each use case can have completely different consistency requirements.

Although there's a lot of database replication technology around, it is a very hard task to pick the right one upfront during design phase, as they will always provide you with a compromise between availability, scalability and consistency. You will have to choose the one that serves your specific application domain and workloads best.

Designing Data Synchronization Requires Close Collaboration of Software Architects, UX Engineers and Requirements Engineers. We experienced that finding a technical solution alone is not enough. The technical solution has to support the application domain and its most important use cases in an optimal way in order to provide outstanding user experience. For example, a technical solution might be to use revision numbers in order to detect conflicting updates performed during offline mode. Clients would be notified about the conflict and would have to resolve the conflict somehow later. Nevertheless, as long as there are no good UI concepts for the presentation and resolving of conflicts this solution might be technically feasible but would result in bad usability.

Data Synchronization is Costly to Develop. From our experience designing and implementing sound synchronization is one of the most challenging tasks that software architects and developers can face. Even experienced engineers regularly fail to think ahead of all the pitfalls and imponderables. Therefore, software architects, UX engineers and requirements engineers should carefully outweigh the costs for it with the gain in user experience that brings the support for offline operations.

Keep Clients Simple with Respect to Data Synchronization. Although we do have a trend towards fat native apps, the latter should be kept simple with respect to data synchronization. If large parts of the synchronization logic like handling of concurrency anomalies, detection and resolution of conflicts is to be implemented by the clients itself, it will add an enormous amount of complexity to the app code. If multiple platforms are to be supported one will have to implement and test all or at least parts of it multiple times. Still in our experience server-side software is easier to implement, debug, quality-assure, maintain and to roll-out.

A Great Deal of Data Validation and Security Checks must be Implemented on the Client. On the other hand, clients must implement a great deal of data validation and security rule checking. If the app permits the user to write invalid data or update data he has no permission for, the app will later on not be able to synchronize back the changes made during offline mode. At the worst, the app becomes unusable due to one faulty data item that is in the change set. Thus, client-side data validation and security defects can have an enormously negative impact on the usability of the app. There are a lot of open questions on how to make the overall system more robust and fault tolerant with respect to this issue: Is it possible to discard only parts of the faulty change set? Moreover, if so, how to determine this part of the change set? If data has to be discarded, how to notify the user about this? This is not trivial, in particular as the user might have done the update some time ago during offline mode and is already in a completely different usage context then.

4 Conclusion

The given paper shows our initial approach to the development of an MSE with an emphasis on the provided UX and architectural decisions. As setting up an MSE is a very complex activity with many pitfalls, we provide our lessons learned from the application of our approach in a large scale case study within the agricultural domain.

Lessons learned from UX show that many additional aspects need to be taken into account besides the user centered conception of a single user interface. In our case study especially the usage of a product philosophy throughout the development of various apps for communication purpose derived as a best practice as well as the usage of an approach for object-oriented user interface conception. This conceptual approach has reduced the effort for conception and maintenance of apps within the MSE in a significant way.

Lessons learned on the technical foundation show that data synchronization has been a key success factor in our case study. There are many approaches available on how to tackle this challenge within MSEs and our lessons learned show that a collaboration among the stakeholders of the user interface and the software architecture overcomes many obstacles. This may sound obvious, but it has also impacts on the other lessons learned described above such as ‘there is no one fits all solution’ and ‘data synchronization is costly to develop’ as we recommend to develop a sound synchronization concept from the very beginning.

Acknowledgement. This work has been partially supported by the German Ministry of Education and Research, grant number 01IS12053.

References

1. Bosch, J.: From software product lines to software ecosystems. In: 13th International Software Product Line Conference, SPLC 2009 (2009)

2. Dörr, J., Trapp, M., Hess, S.: Mobile Prozesse: eine Chance für die Wirtschaft. Computerwoche (2014). http://www.computerwoche.de/a/mobile-prozesse-eine-chance-fuer-die-wirtschaft,2555126
3. Hess, S., Kiefer, F., Carbon, R., Maier, A.: mConcAppt – a method for the conception of mobile business applications. In: Uhler, D., Mehta, K., Wong, J.L. (eds.) MobiCASE 2012. LNICST, vol. 110, pp. 1–20. Springer, Heidelberg (2013)
4. Deere, J.: Press Release (2014)
5. Keuler, T., Knodel, J., Naab, M.: Fraunhofer ACES: Architecture-Centric Engineering Solutions. Fraunhofer IESE, IESE-Report, 079.11/E (2011)
6. Naab, M., Braun, S., Lenhart, T., Hess, S., Eitel, A., Magin, D., Carbon, R., Kiefer, F.: Why data needs more attention in architecture design - experiences from prototyping a large-scale mobile app ecosystem. In: International Working Conference on Software Architecture, WICSA 2015 (2015)