Electronic Markets (2019) 29:407–421
https://doi.org/10.1007/s12525-019-00334-1

POSITION PAPER

# Effectual application development on digital platforms

Alan Hevner[1] · Onkar Malgonde[2]

Received: 20 February 2018 / Accepted: 8 January 2019 / Published online: 7 February 2019


## Abstract

The development of novel software applications on digital platforms differs radically from traditional software development. In this position paper, we posit that software development managers and teams face unique challenges in platform environments and require new development approaches to be successful. While traditional software development approaches have focused on achieving application-market match, platform-based applications must also achieve application-platform match, application-market match, value propositions exceeding platform's core value propositions, and novelty. We argue that these desired properties support a new vision of the software development team as entrepreneurs. To support this positioning insight, we discuss the limitations of existing software development approaches and introduce an innovative approach for application development on digital platforms that is grounded in the *theory of effectuation* from the field of entrepreneurship. We investigate an existing application development environment (Apache Cordova) on digital platforms to see if the concepts of effectuation are present. The preliminary findings provide support for the promise of effectual development methods. We conclude with a call for innovative effectual methods of software development on digital platforms and an accompanying research agenda.

**Keywords** Digital platforms · Software application development · Effectuation · Novelty

**JEL classification** M15

## Digital platforms

Digital innovations are new combinations of digital and physical components characterized by reprogrammability, homogenization of data, and use of digital technology (Yoo et al. 2010). The *digital platform,*[1] as shown in Fig. 1, is becoming a pervasive technology that is rapidly transforming the ways in which products and services are produced and consumed in our market economy (Parker et al. 2016). Digital platforms represent complex electronic markets that facilitate value-creating interactions between disparate producers and consumers spanning different industries and geographies (see: Alt and Zimmermann 2014, 2015). These electronic markets provide new opportunities and challenges to participating organizations due to the ease of interactions between multiple stakeholders.

The *platform ecosystem* consists of the platform and the applications that are available via the platform or connect to the platform via the interfaces offered by the platform, in a contextual *environment* of regulations and competitors (Tiwana 2013; Tiwana et al. 2010). Platforms enable value-creating interactions among organizations with disparate resources and specializations. This transfers the locus of innovation, which traditionally has been within the organization, to a diverse set of external organizations that collaborate to develop applications available via the platform. A platform *owner* is the organization or group of organizations that determine the architecture, governance, and curation mechanisms for the platform. *Producers* are the organizations that develop

[1] We adopt the definition of digital platform given by Tiwana et al. (2010) and extended by Ghazawneh and Henfridsson (2015, p. 199) as "software-based external platforms consisting of the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate." For reviews of digital platforms and their components, the reader is referred to de Reuver et al. (2017), Tiwana (2013), and Parker et al. (2016).

This article is part of the Topical Collection on Design Science Research in the Networked Economy

Responsible Editor: Alexander Mädche

✉ Alan Hevner
ahevner@usf.edu

[1] Department of Information Systems and Decision Sciences, University of South Florida, Tampa, Florida, USA

[2] Department of Operations Management and Information Systems, College of Business, Northern Illinois University, Illinois, USA

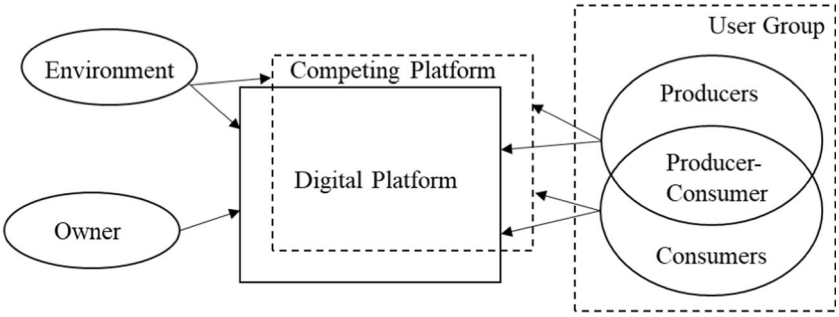


**Fig. 1** The digital platform ecosystem

applications (extensions to the core functionality offered by the platform) that are available via the platform. *Consumers* are the organizations that use applications offered via the platform. Further, consumers can mix-and-match applications available via the platform to satisfy their needs. Examples of software platforms include Apple's iTunes, Google's Play, Salesforce's appexchange, SAP's HANA, Valve's Steam, Microsoft's Azure, and Instructure's Canvas, among others.

The development of novel software applications on a digital platform differs radically from traditional software development. Motivations of this position paper are to understand these key differences and to discuss the limited applicability of existing software development approaches for digital platform environments. The paper draws on entrepreneurship theories to propose a novel effectual approach for software development on digital platforms. The following digital platform characteristics illustrate the challenges faced:

- A platform offers a compelling set of *core value propositions* to its consumers (Parker et al. 2016). Applications on the platform play off the core values and add novel extensions to the platform's capabilities (Koch and Bierbamer 2016).
- Over time, the platform core values evolve based on consumer demands and goals (Haile and Altmann 2016) and, as a result, platform applications are added, updated, and dropped.
- As the number of similar applications on a software platform increases, investment incentives for individual producers are crowded out (Boudreau 2012). Similarity of applications available via a platform limits the platform's value proposition and incentivizes the platform to assimilate those features into the core value proposition of the platform. Consequently, applications whose value proposition is assimilated into the core offering of the platform are discontinued.
- All applications must adhere to connection specifications and development procedures determined by the platform (Tiwana 2013). Platforms provide standard connection interfaces[2] in the form of application programming interfaces (API's) that are used by applications to access common features within the platform. Thus, platform owners and user groups often require that application producers follow certain best practices such as 'look and feel' interactions. In many cases, the platform owners evaluate and approve (i.e. curate) new applications before they are offered to consumers via the platform.
- Application developers may request changes in platform interfaces and protocols based on environmental changes or new customer demands. For example, popular digital platforms such as Android, Azure, HANA, among others encourage community input to request new features.
- Platforms exhibit different levels of maturity over time. Changes to platform architecture, governance, and curation mechanisms requires application producers to adapt their applications and routines to comply with updated platform regulations. For example, Microsoft Azure[3] platform releases multiple new and updated features every week. As features are introduced and updated, platform's maturity improves which impacts application development teams' decisions (e.g. upgrade, obsolete) about their application.

To manage these unique challenges and provide value-added applications, producers must achieve four key goals (a) application-platform match, (b) application-market match, (c) value propositions exceeding platform's core value propositions, and (d) novelty. An application is valuable to platform consumers if it provides features and extensions that can enable consumers to perform activities that the platform does not provide. Further, an application is novel if it provides features and extensions that the platform and other applications do not provide.[4]

[2] Updates to a platform's connection interfaces are typically focused on the data available via the interface. For example, an update to a platform's core may allow the sharing of additional data. The connection interface may use the same technology and standard (e.g. XML or JSON).

[3] https://azure.microsoft.com/en-us/updates/ (accessed 07/11/2018)

[4] These follow the accepted definitions of value and novelty in software development context as used by Austin and Devin (2009) who builds on extant literature related to new product development in the fields of information systems, business, and psychology.

Prior research in software application development largely focuses on the desired properties of application-market match (Harris et al. 2009) and project performance (Weiner et al. 2016). However, the success criteria for software applications on digital platforms[5] significantly exceeds these traditional properties since the environment provided by digital platforms is more integrated and complex (McKelvey et al. 2015). Similarly, Bygstad (2016) highlights the emergence of knowledge regimes which focus on connectedness, innovation, and experimentation in the emerging world of consumerization and Internet of Things. These desired properties of application-platform match, application-market match, value propositions exceeding platform's core value propositions, and novelty for an application on digital platform support a new vision of the software development team as *entrepreneurs*.

In this position paper, we propose an effectual approach for development of novel applications on digital platforms. Our arguments are structured concisely as follows. First, we review the theory of effectuation and develop a research model for representing the key constructs of effectual software development. A qualitative data analysis is performed to garner support for the presence of effectual concepts in the development of systems in the Apache Cordova environment—an open source software development environment that supports multiple digital platforms. We conclude with a discussion of implications of this study for application development on digital platforms along with research agenda for future research.

## Effectual software development

The consumer demand for new and interesting applications on digital platforms has energized the software development world to greater requirements for delivery speed and higher quality user experiences. Existing software development approaches often fail to provide the knowledge, constructs, and processes to address fully the challenges of digital platform application development. Thus, we propose a new, theory-based approach of *effectual software development*.

### Effectuation

Sarasvathy (2001) conceptualizes *effectuation* as the opposite of *causation*.[6] Unlike causation, effectuation does not focus on finding causes that explain or achieve a given (intended) effect, but considers available actions through given means and their spectrum of possible effects. Effectuation therefore is about designing and evaluating alternatives with differing effects (and choosing one of them) instead of choosing among given alternatives which all lead to the same effect. Thus, effectuation logic constitutes a logic of controls; specifically controlling the future by actively shaping one's environment within one's possibilities (e.g. digital platforms).

In effectuation, the choice of action depends on the three given means of (a) the actors (effectuators) themselves and their traits ("who I am"), (b) their knowledge ("what I know"), and (c) their social connections ("whom I know"). It also depends on what the effectuators can imagine to be possible effects and what they perceive the corresponding risks or potential losses of those effects to be. These risks and losses are matched with effectuator's set of aspirations, leading to the eventual choice of action. Neither the means nor the aspirations are treated as invariant, leading to a concept that embraces flexibility and dynamism, allowing the exploitation of emerging contingencies (Sarasvathy 2001). Figure 2 illustrates the basic concepts of effectuation.

Two decision heuristics are employed when the entrepreneur pursues possible actions: *acceptable risk/affordable loss* and *logic of control*. Acceptable risk/affordable loss favors those actions which carry a degree of risk that is acceptable to the entrepreneur. It avoids actions that carry undue existential risk to the enterprise. This contrasts with causation where decision making is based on expected returns of the alternative actions. Logic of control involves decision making based on factors that the entrepreneur can control as opposed to the prediction of future events. As the iterative process of effectuation evolves, the entrepreneur accumulates new means and goals, and converges to a set of effects resulting in an artifact that embodies the desired aspirations. Before we apply effectuation to software development context, we present the framework of control and prediction to understand the need for effectuation and its positioning in the spectrum of software development approaches. This framework of control and prediction allows us to understand the underlying theoretical underpinnings of the different software development approaches. Aligning these approaches with the characteristics of digital platforms will allow us to evaluate the applicability of different software development approaches.

### Prediction vs. control in software development

The proposed effectual software development process can be contrasted to more traditional approaches for developing software such as plan-driven, controlled-flexible, and ad-hoc (Harris et al. 2009) by considering (as seen in Fig. 3) the dimensions of control (x-axis) and prediction (y-axis) (Wiltbank et al. 2006). Increasing prediction posits that a

[5] Extant work (e.g. Weiner et al. (2016) and Harris et al. (2009)) has considered software development for technological platforms (e.g. JAVA or .NET environment which define specific framework requirements for the development team). However, the platform considered in prior work does not include the uncertainty and risk characterized by third-party ownership of the platform, competing firms on the platform, and focus on novelty of the application.

[6] "Effectuation processes take a set of means as given and focus on selecting between possible effects that can be created with that set of means. Causation processes take a particular effect as given and focus on selecting between means to create that effect." (Sarasvathy 2001, p. 245)

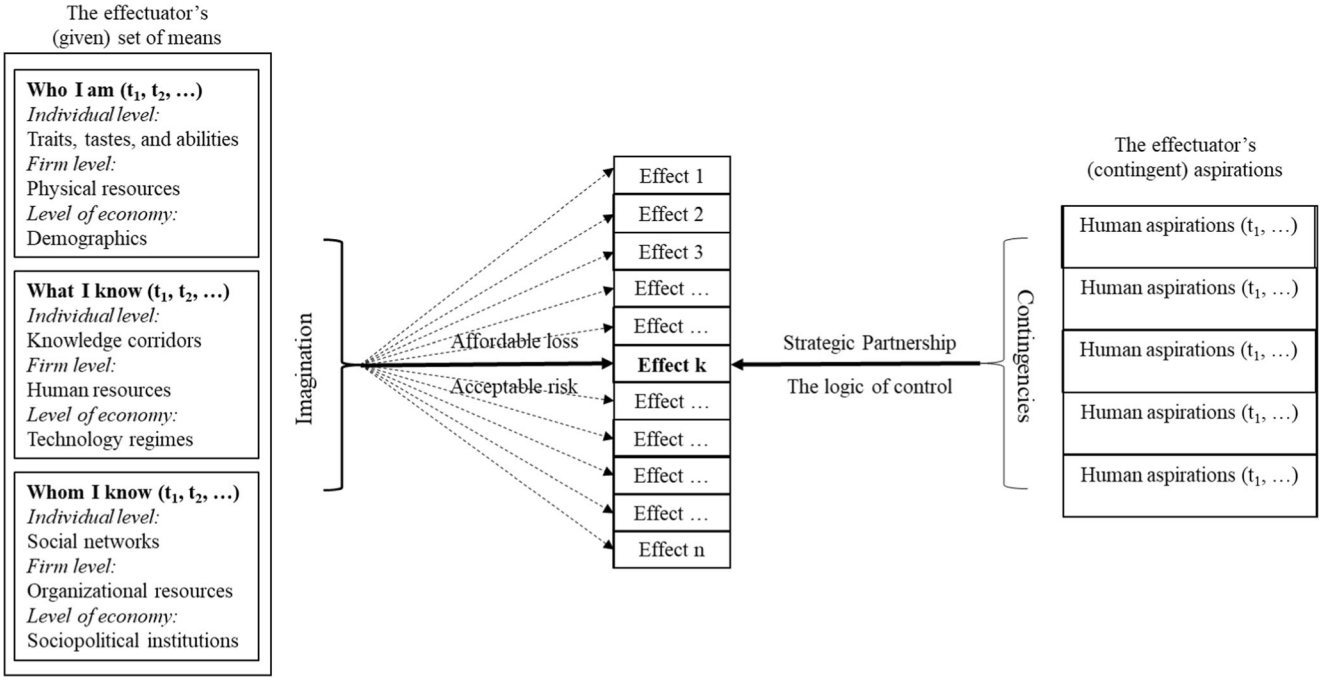


**Fig. 2** Theory of effectuation (Sarasvathy 2001)

development organization can predict the exogenous environment and position itself to be relevant in the future via planning. Increasing control focuses on the ability of an organization to control and shape its own endogenous environment to be relevant in the future via adaptation.

The two dimensions characterize two different entrepreneurial strategies: planning and adaptation. They differ on how to handle uncertainty in the exogenous environment. First, the *planning* approach advocates predicting the exogenous environment and positioning accordingly. Applying the framework to software development, this approach defines the underlying logic of a *plan-driven* development approach. Thus, software project plans are devised, and resources are identified and acquired, at the start, with an understanding that the software product will be relevant (and profitable) upon its completion.

A predominately *adaptive* approach suggests positioning for future relevance via bounded goals and on-going feedback from the environment. With few controls in place (i.e. trial and error), this approach leads to an ad-hoc approach to software development, where the development team is constantly calibrating its course by reacting to changes in the environment. Spanning the planning and adaptive approach is the *controlled-flexible*[7] approach in software development, where planned control mechanisms are inherently prediction-based while emergent control mechanisms introduce flexibility to adapt to uncertain environments (Harris et al. 2009). Discussion on the differences and similarities between plan-driven, controlled-flexible, ad-hoc, and effectual approaches is presented in next subsection and summarized in Appendix 2.

With increasing emphasis on control, a development strategy involves actively shaping the environment by making it endogenous rather than navigating and positioning in an exogenous environment. In the *visionary* strategy, the firm emphasizes construction by considering possible alternatives and proactively working to realize their potential. Consideration

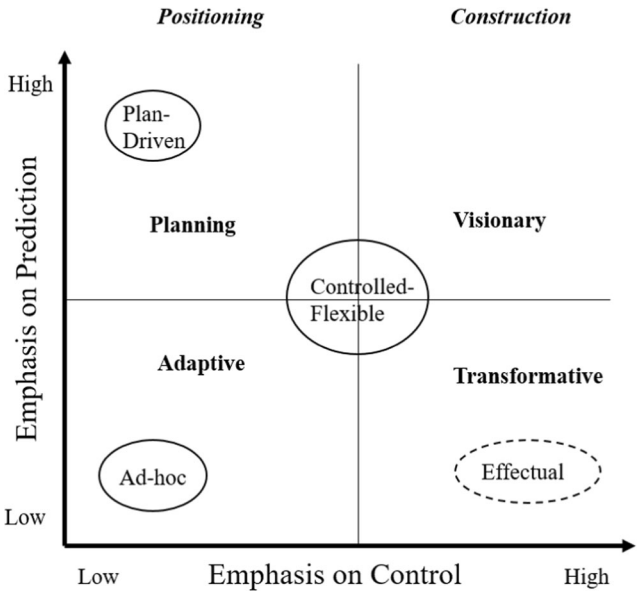


**Fig. 3** Framework of prediction and control (adapted from Wiltbank et al. (2006))

[7] We use the term controlled-flexible in preference to the term agile. The term 'controlled-flexible' is identified formally by Harris et al. (2009) with a Control Theory foundation. The term 'agile software development' is used informally in common parlance and can identify a broad range of flexible software development practices (Baskerville et al. 2011).

and analysis of possible alternatives includes predicting the future possibilities and the alternatives' potential to be relevant. However, proactively working to realize the alternatives includes treating the environment as endogenous and achieving goals by gathering required resources.

A *transformative* strategy focuses on controlling its environment with minimal prediction. Unlike a visionary approach, a transformative approach does not emphasize consideration and analysis of alternatives. Instead, a transformative strategy focuses on existing means to derive possible alternatives and selects alternatives which allow the firm to embrace future contingencies. Focusing on existing means, transformative approaches can enhance their understanding of goals and means with intermediate artefacts. Wiltbank et al. (2006) argue that effectuation is applicable in environments which can be classified in the transformative quadrant.

We note that similar control-based directions are underway in other areas of research where prediction-based approaches have limited applicability (Hevner 2018). For example, Stanley and Lehman (2015) discuss artificial intelligence algorithms and their design approaches to solve problems like finding a way out of a room. They find that an algorithm which focuses on exploring its solution space is better positioned to find solutions than another algorithm that is focused on the task and attempts to predict possible solution avenues. Also, in strategic management literature, Hayes (1985) and Kim and Mauborgne (1997) argue the limited applicability of objective and prediction driven approaches to projects that are increasingly in uncertain and risky environments. Here, we propose that novel software development on digital platforms demands more adaptable approaches that emphasize controls over predictions. We now discuss the applicability of effectuation theory to the practice of software development on digital platforms.

## Applying effectuation to software development practice

The development of novel software applications requires human creativity and innovative design activities. For example, Drechsler and Hevner (2015) provide guidance for incorporating the concepts of effectuation into the design science research (DSR) paradigm. They argue that effectuation-oriented DSR may provide a superior lens to examine problem spaces that are characterized with uncertainty and dynamic evolution. In our research, we make a practical application of this conceptualization to propose an effectual process to develop novel applications on software-based platforms.

### Why effectuation?

Effectuation aligns with software development on digital platforms due to the limitations of causation-based approaches in the literature (Harris et al. 2009) and the challenges offered by digital platforms. Causation-based approaches to software development identify a particular goal and realize it through a linear and/or iterative development process. These are prediction based approaches, where the application's ultimate fit and utility in the platform context is identified a priori. Such a priori identification of application's utility is possible in environments that are characterized by greater degrees of certainty and stability (Gill and Hevner 2013).

However, software development on digital platforms must navigate uncertain, resource constrained, and high-risk environments. Such settings render software development approaches from the traditional realm of prediction largely infeasible. We contend that current software development processes and methods do not effectively address the challenges of digital platforms for the following reasons:

- They focus on product-market match. The emphasis is on iterative development toward the goal of a single product. Each cycle/iteration allows the team to evaluate the market match of a predicted product and realign, if needed. While the effectual approach also emphasizes iterative development, it focuses on building and assimilating the new knowledge to expand its resources and attenuate its aspirations. The team's aspirations (generalized end goals) form the key evaluation criteria before the application is deployed.
- They perform actions based on expected returns. In other words, action alternatives are favored if they provide the greatest return to achieve the success criteria. The effectual approach evaluates alternatives based on their risk and controllable aspects. Action alternatives are chosen if the risk associated with those actions are acceptable to the team and if all the aspects of the action alternative are controllable to the team.
- They focus on the development of an end product with little attention to the intermediate design products along the way. In effectual approach, rapid effectual cycles identify design artifacts (documentation, code, prototypes, proof of concepts, among others) as intermediate effects to provide feedback to subsequent iterations. The effectual approach treats the uncertain environment as endogenous and shapes it via these intermediate effects.

Digital platforms represent socio-technical, dynamic, and challenging contexts for software development teams. Using the effectual perspective allows software development teams to identify multiple possible effects based on their available means. Through market and stakeholder feedback, the

development team can iteratively attenuate their aspirations and identify appropriate effects that embody their aspirations, fit the application context, and provide the greatest amount of utility to its users. This approach is in contrast to the causal approach since the team does not identify a particular goal; rather they iteratively attenuate their aspirations to arrive at the desired effects which will include the final product but also multiple intermediate artefacts throughout the development process.

## A model of effectual software development

The development of a practical representation of effectual software development begins with a fuller understanding of components of the process and their relationships as presented in Fig. 4. The theory of effectuation provides the theoretical lens to identify constructs and relationships which are adapted and extended for the software development context. To begin, we note that the nature of application influences the overall process enacted by the software development team. Nature of application may be diverse across multiple dimensions—functionality, scale, interoperability, data, stakeholders, dependencies, technologies, among others. These factors provide context and guidance for the definition of the effectual software development process.

As resources to the software development project, the effectual model considers three key, independent components:

- *Means* for the project manager and development team are the existing resources that are available to them. Means consist of technology and skills (programming language, API's, tools), market knowledge (customer orientation, seasonal trends, patterns from archival data), platform knowledge (connection interface, tools and technology, best practices, available API's on the platform), control mechanisms (scope boundaries, stakeholder feedback), the social capital that the development team can draw upon, and team's culture which may provide routines and communication channels.
- The *software platform* provides a set of resources and constraints. For example, the connection interfaces to the platform, development guidelines, tutorials, and development standards that provide resources for the project to draw upon while constraining them to those specific alternatives. The platform also shows different levels of complexity and maturity for the development team to consider.
- Four *aspirations* for the project team are identified – application-platform match, application-market match, value proposition of the application should exceed the core value proposition of the platform, and novelty of the application. These aspirations for software development on platforms are in addition to the more traditional entrepreneurial aspirations of human, social, and economic goals.

For the development team, means, platform, and aspirations exist a priori to the development process. Drawing on these resources, the software development team evaluates and selects action alternatives that are feasible and will move the project forward. An action may encompass identification of new application feature, fixing an existing issue, or revising a design feature. Identification of actions draws on a subset of means and aspirations. Thus, an identified action may draw on technological means to satisfy application-platform match

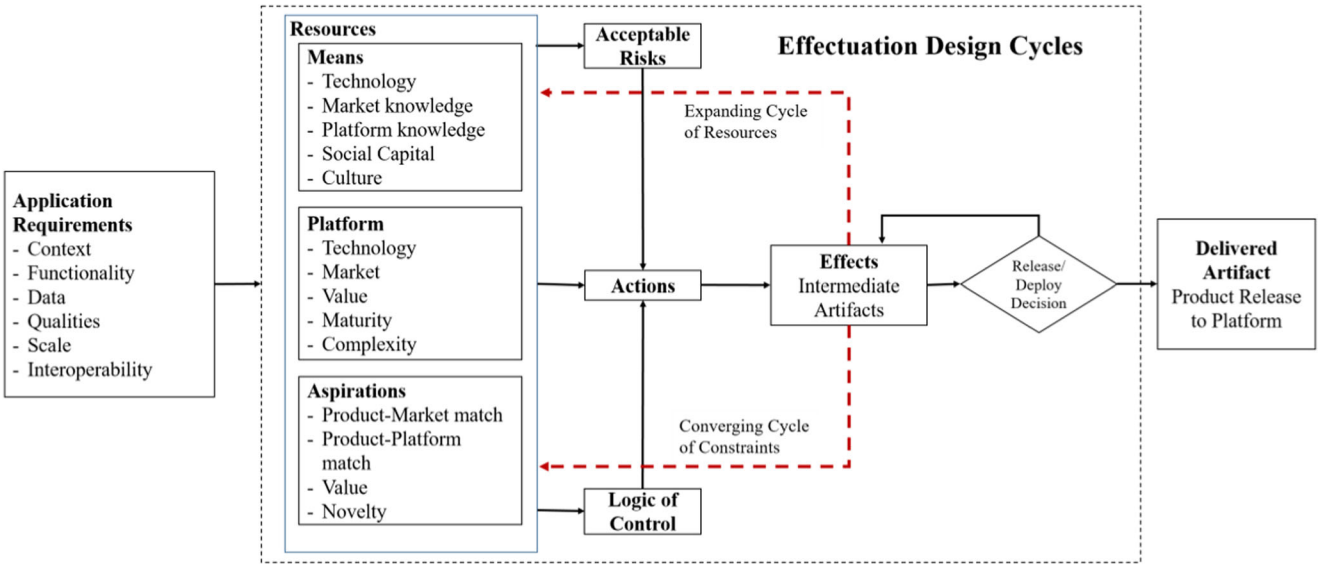


**Fig. 4** Research model of effectual software development

while another action may draw on market and platform knowledge to accomplish novelty. Pervasive in the identification of actions is the platform's capabilities and constraints that the team must consider.

The mechanism to select appropriate actions from identified action alternatives is provided by two heuristics: *acceptable risk* and *logic of control*. According to classical decision theory, risk associated with an alternative is the variation in its possible outcomes (March and Shapira 1987). The larger the variation in possible outcomes, the larger is the risk associated with the alternative. Thus, evaluation of potential actions is based on the trade-off between its expected return and associated risk. An action is said to possess acceptable risk if the development team can perform corrective actions in case the alternative does not satisfy the team's aspirations.

Further, the managerial perspective notes that risk is *controllable* and *modifiable* through skills and information (MacCrimmon and Wehrung 1986). The logic of control emphasizes controllable aspects of future events i.e., a focus on aspirations that can be controlled by the project team (Sarasvathy 2001), favouring actions that can be controlled by the team given its current resources. For example, implementation of a feature is not favoured if it requires API's from the platform which are not yet available. The project team conducts a risk analysis (Benaroch et al. 2006; Flyvbjerg and Budzier 2011; Lyytinen et al. 1998) on the set of possible actions. Actions that have an acceptable risk are identified. Platform state, existing portfolio of controls (Harris et al. 2009; Kirsch 1997), and aspirations of the project team identify the controllable aspects of the possible actions.

Together, actions selected via the mechanism discussed above give rise to an effect. An *effect* is the operationalization of abstract aspirations (Sarasvathy 2001). Specifically, effects encompass the growing set of intermediate artifacts that include the features and operational specifications of the application being developed by the team. In software engineering terminology, effects may include software specifications, nightly or weekly builds, documentation, demos, design documents, user interface mock-ups, proof of concepts, backlog items, and so on. The current set of project effects allows the team to identify new avenues (features, improvements, bugs) and attenuate aspirations. The effects are 'intermediate' because they represent artifacts which are in the state of development. Effects embody the current understanding of the software application development team. Effects also serve as a point of reference to validate ideas and decisions. We term this iterative approach to identify and grow the set of intermediate artifacts as the *effectual design cycle*.

This effectual design cycle allows the software development team to grow system artifacts which represent the knowledge base of the team. The team evaluates its aspirations with this knowledge base and identifies new resources. This mechanism gives rise to new means and constraints for the development team – *expanding cycle of resources*. New means stem from an improved understanding of the problem space. Similarly, new constraints are identified that help retain appropriate and promising aspects of the aspirations – *converging cycle of constraints*. Finally, the final system *Artifact* (application product or service) is the realization of team's aspirations and is developed, implemented, and deployed on the platform by the team.

Applications developed on digital platforms have numerous 'releases' to the platform. These releases represent on-going outcomes of iterative effectual design processes. Each release may include minor and/or major improvements, bug fixes, and enhancements. A common understanding in the team should be that the next release of the software application is not the last or concluding release for the application. We see three primary reasons for this continual need for releases. First, the application development team prioritizes issues based on contextual needs. Second, market-driven needs require frequent releases to the platform as competitors update their applications and users' requirements evolve. Thus, the application needs to be updated in accordance with market forces. Finally, changes to the digital platform require frequent releases to ensure that the application is compatible with the platform. These changes are often in the form of changes to APIs—add, update, or deprecate. Consequently, the development team has to ensure that every release uses the platform's updated APIs.

We propose the model of effectual software development as a radically new approach for planning and executing application development on digital platforms. The following section provides some initial evidence of effectuation found in current platform projects.

## Evidence of effectuation in platform development projects

To investigate the new ideas of effectual software development on digital platforms, we perform a qualitative study of open-source application development projects. This analysis is not designed to provide a rigorous test of the presented research model. Instead, it provides evidence for the effectuation ideas as found in existing software development projects on digital platforms.

### Research method

Three key selection criteria are established to identify appropriate samples for data collection and analysis: (a) the application should not be developed by an individual only, (b) the digital platform should be owned by a different organization, and (c)

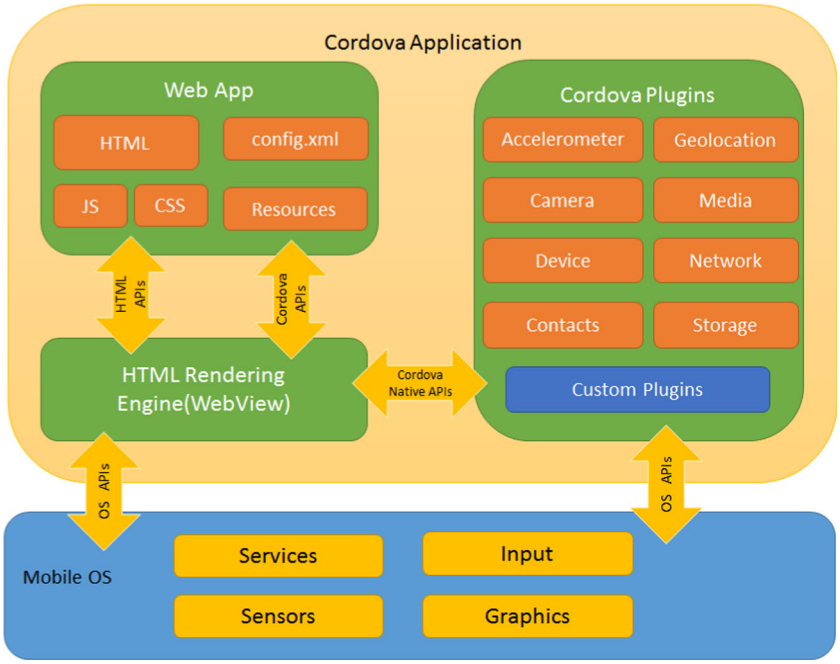


**Fig. 5** Apache cordova architecture (From https://cordova.apache.org/docs/en/latest/guide/overview/)

application users should have alternative options other than the application under investigation. We identified *Apache Cordova* as an open-source mobile application development framework (Fig. 5). The application is developed by a distributed team of contributors. The digital platform on which the application is built is controlled by disparate organizations. Finally, rival applications for Apache Cordova are available to its users.

Following the mantra of Apache Software Foundation (ASF), the Cordova application framework is used by numerous application developers to develop platform-based applications and provides tools and interfaces that can be readily used by developers. Apache Cordova provides all the interfaces and plugins that the development team needs to develop an application which can then be published across multiple platforms. Cordova[8] supports seven platforms—Android, iOS, Windows, Ubuntu, Blackberry 10, WP8, and OS X. *Web View* provides user interface capabilities, *Web App* provides configurational settings for the application, and Cordova *Plugins* allow seamless communication within application components and the platform. The *Mobile OS* platform provides standardized plugins, which are regularly updated by the platform owner.

All Apache projects are required to store and host programming activities, decisions, and status of the project. Projects adhere to these requirements using mailing lists, project management and version control tools, and/or messaging platforms. In our study, we extract data from the project management tool. Specifically, we focus on this dataset because (a) all data are available, (b) the dataset consists of issues raised by active contributors, and (c) the dataset includes requests for information, bug fixes, feature requests, suggestions, and discussions. We focus on completed *user stories* that describe a specific feature request and/or issue with the application and/or platform. User stories often include very technical descriptions of requested features and supporting details. Completed user stories are suitable for this research since they provide the issue and its description addressed in the story, typically, with a solution that is provided and implemented in the application. Some stories have additional discussions on the viability of alternative solutions to the issue being discussed. Story descriptions and related comments for over 1000 stories were extracted and analysed. The data analysis is supported with documents from proposals, board reports, and project documentation. The unit of analysis is the issue reported in the story.

We analyse the data as follows. First, inspecting all stories in the database, we remove unclear or non-descriptive stories. These include stories that do not discuss any specific issue in depth, provide a link or non-conclusive short description, and/or provide a blob of program code without accompanying discussion. The user story needs to clearly present the issue at hand. As the initial inspection retained clear and descriptive stories, they were subjected to qualitative analyses. These analyses include coding the data with identification of relevant terms and definitions. Finally, inferences were derived from selected stories and triangulated from multiple sources. Through these rigorous filters, we refined the initial set of 1051 stories in order to identify

[8] The Apache Cordova application supports application development digital platforms on mobile operating systems. Application developers use the Apache Cordova framework to develop their applications as Cordova abstracts the complexity of developing specific components for different platforms. Therefore, this empirical study considers Apache Cordova as the application and mobile operating systems as platforms.

**Table 1** Research method constructs, codes, and definitions

| Construct | First Cycle Code | Operational Definition |
|---|---|---|
| Means (existing resources at hand) | Technology | Existing technological capability within the team (in this case, the community) – programming languages, tools, configuration, testing, documentation, etc. |
| | Market knowledge | Existing knowledge about the platform market (alternatives, competitors) |
| | Platform knowledge | Existing knowledge about the technological state of the platform |
| | Social Capital | Capital that the team can draw upon to append existing means |
| Platform (resources and constraints provided by the platform) | Technology (API) | Technological resources and constraints provided by the platform (APIs, programming language, setup, features) |
| | Market | Existing offerings on the platform market |
| | Value | Existing value offered by the platform to its customer (in terms of features that the users can use – tangible) |
| Aspirations | Product-market match | The features to be built in the product should match the requirement of the market |
| | Product-platform match | The product should be technologically compatible and functional on the platform |
| | Exceed Platform Value | The features being built in the application should help exceed the application the core set of value provided by the platform |
| | Novelty | Technological or feature based novelty of the application that the existing applications and platform do not cover. |
| Acceptable Risk | Commit limited resource | Commit limited technological and people resources to any given feature. |
| | Application recoverable after failure | If implementation of the given feature results in failure, it should not jeopardize entire application. |
| | Risk Analysis | Risk portfolio of an alternative are determined before decision-making. |
| Logic of Control | Logic of Control | Decision making based on factors that the team can control as opposed to prediction of future events. |
| Actions | Fixed bugs | The issues that were identified based on means and fixed. |
| | Completed Tasks | Feature requests which were identified and completed using means and acceptable risk. |
| Effects | NA | Collective documentation and understanding of which features and issues are to be addressed in the project. |
| Expanding Cycle of Resources | New technological knowledge | Identify new API's, tools, and configurations that can be used by the application. |
| | New market knowledge | Identify new requirements that the market needs. |
| | New platform knowledge | Identify new API's, tools, and configurations that are provided by the platform. |
| Converging Cycle of Constraints | Converging technological (means) constraints | Identify specific API, tool, or configuration for the application from competing alternatives. |
| | Converging feature constraints | Identify specific feature for the application from competing alternatives. |
| | Converging platform constraints | Identify specific API, tool, feature, or configurations competing alternatives provided by the platform. |

42 complete user stories with sufficient detail for full analysis. We use Atlas.ti qualitative data analysis software for our analysis. To aid our coding procedure, we developed a qualitative codebook that identifies sub-codes and operational definitions (Table 1) for each construct in our model (Fig. 4).

The sub-codes are identified from the research context, theoretical constructs, and conceptual research model. Operational definitions are identified based on the research context and prior empirical studies on effectuation (Chandler et al. 2011; Perry et al. 2012). Further, the coding scheme is flexible to add new sub-codes as they emerge from the data. The codebook guided our first-order coding. Using *descriptive* coding technique (Miles et al. 2013), sub-codes from the

codebook were applied to each story where applicable. To illustrate the coding, Appendix 1 provides several sample stories from our database and the codes that are assigned to them. To address construct validity, multiple sources of data—stories, documentation, contributor comments, board reports, and proposals—are tapped to ensure that the findings converge. Reliability of the study is addressed with (a) programmatically retrieving and storing analysed stories locally from the project management tool, (b) maintaining the qualitative codebook of codes and operational definitions, and (c) developing matrices from the labelled data.

## Results

The results of the analyses of the Apache Cordova projects are presented in Table 2 including the first cycle codes (and related constructs identified in Fig. 4) and the frequency of the codes. As the secondary data used for this analysis consist of contributors' descriptions of issues and feature requests for the Cordova applications, the data are characteristically technical in nature. This readily translates into identification of technological means available to the application development team that is specific to the application and platform. We identified 40 stories that show technological means for the development team. Available means include knowledge about market needs (feature requests), value propositions provided by the platforms, and new features that are introduced by platforms or competing applications (through developer conferences or official press releases). Similarly, the technological opportunities and limitations by platforms are discussed by contributors. Current working of API's and the value they provide to the user are discussed and coded in 23 stories. This leads to identification of limitations and opportunities that serve as value additions to the current value proposition of the platform and serve the market need. This evidence points to the nascent application market of the project. Building on existing technological and market knowledge, possible alternatives are identified. Further, these stories do not predict potential changes to the market and platform. Instead, the focus is to build the application based on current understanding of the technology, platform, and market.

The analysis also leads to identification of aspirations in the team's decision making and actions. Specifically, the application-platform match is one of the central driving forces across these stories since contributors focus on technical

**Table 2** Constructs and their frequency in the data

| Construct | First Cycle Code | Frequency |
|---|---|---|
| Means | Technology | 40 |
| | Market knowledge | 5 |
| | Platform knowledge | 20 |
| | Social Capital | 2 |
| Platform | Technology | 23 |
| | Market | 7 |
| | Value | 8 |
| Aspirations | Product-market match | 8 |
| | Product-platform match | 24 |
| | Exceed Platform Value | 14 |
| | Novelty | 15 |
| Acceptable Risk | Commit limited resource | 33 |
| | Application recoverable after failure | 5 |
| | Risk Analysis | 21 |
| Logic of Control | Logic of Control | 32 |
| Actions | Fixed bugs | 20 |
| | Completed Tasks | 11 |
| Expanding Cycle of Resources | New technological knowledge | 37 |
| | New market knowledge | 5 |
| | New platform knowledge | 21 |
| Converging Cycle of Constraints | Converging technological constraints | 24 |
| | Converging feature constraints | 9 |
| | Converging platform constraints | 11 |

aspects that lead to seamless operation between the application and platform. 24 stories are coded to identify application-platform match. Further, the analysis finds support for the aspiration of introducing novelty to the application (15 stories) and ultimately adding value to the existing value proposition provided by the platforms (14 stories). The common theme in these aspirations is identification of opportunities (limitations and/or enhancements) for value addition through existing means and platform knowledge, and introducing novel features that take advantage of the platform's opportunities.

The heuristics of acceptable risk and logic of control also find strong support in our analysis. Each story is identified and addressed by (typically) one contributor. Thus, the team is devoting limited resources for each issue and feature, and 33 stories are coded for this sub-code. Alternatives identified—do feature A or B or C—accompany risk analyses that discuss technological implications on the application and platform, novelty, and extending the platform's value proposition. 21 stories are coded to show risk analysis and identify alternatives that have acceptable risk associated with them. Further, actions identified by the team embody the logic of control and are coded in 32 stories. These include decisions based on the current means, platform knowledge, and the aspirations of the team, rather than predicting which actions would enhance the application. Finally, the application is already in use by an array of users which provide feedback to the development team. This represents a control driven approach rather than prediction based approach that would identify the goals of an application a priori.

Actions (32 stories coded) lead to intermediate effects, which are the operationalization of team aspirations. Each iteration of the Cordova application resulted in an intermediate effect that, in turn, expanded means and attenuated aspirations. Specifically, intermediate effects help identify technological avenues, tools, limitations, and features, that increase the fit and utility of the artifact. 37 stories are coded to identify expanding technological knowledge. In addition, intermediate effects improve the platform knowledge for the overall team, as new features are implemented that connect to the platform and add new value to its existing value proposition. Note that we did not code the 'effect' construct since this is a new concept in our proposed model that current development projects do not consider. Our interpretation of an effect as an intermediate software development artifact advanced during our analysis.

Overall, the frequency of sub-codes identified in our analyses justifies the conjecture that software development teams developing novel applications on digital platforms employ the constructs of effectuation even when the terms used in the processes may not exactly align with those used in effectuation context. Also, these stories span across multiple iterations. For every iteration, a set of stories represents the growing set of intermediate effects that are developed and tested. This provides feedback to the development team that expands its resources and attenuates its aspirations.

## Conclusion

Our goal is to propose an innovative and disruptive effectual approach for the development of software applications on digital platforms. Development teams must break from traditional, prediction-based methods and become more entrepreneurial as they design, implement, and deploy new applications. In this position paper, we identify the unique challenges and desired properties for the development of software applications on digital platforms which surpass the traditional conceptualizations of product-market match and project performance as key success criteria for software applications. To address these challenges, an effectual approach to software development has been proposed that is developed through lens of the theory of effectuation. Empirical analysis using qualitative secondary data from open-source projects provide evidence of its efficacy.

This study has important contributions and implications for research and practice. Building on the challenges identified for the development of novel applications on software platforms, we advance an innovative vision of software development where the software development project is seen as an entrepreneurial endeavour with the project manager and development team as entrepreneurs. An effectual approach to development of novel applications on software platforms is proposed and described. Grounded in the theory of effectuation, the approach introduces context specific constructs (platform) and theorizes and adapts existing effectuation constructs to the software development context. The effectual approach to software development introduces new constructs and feedback processes in software development research – aspirations, focus on existing resources, decision heuristics, and expanding and converging cycles. These effectual processes provide improved explanations for novel application development on software platforms where existing development approaches lack resonance with important constructs from Fig. 4, such as means, actions, and aspirations.

The effectual approach proposed in this paper contributes to practice. First, we draw attention to the development approach for novel applications on software platforms which has received limited attention in the information systems literature. Attention to development approaches on software

platforms is particularly important and timely, given the proliferation of platforms (Parker et al. 2016). Second, application producers have a direct interest in development approaches that specifically address the unique challenges offered by platform ecosystems. These interests include development of novel applications and maintenance of existing applications. Third, platform owners also benefit from the introduction of novel applications on software platforms. As the locus of innovation shifts from within the organization to a heterogeneous base of application producers, introduction of novel applications allows the platform to serve diverse consumer segments and introduces new demand within the user group.

In our discussions so far, we have refrained from exploring specific software development methods (e.g. Scrum or Kanban) in use today because the focus of this research is the broader software development approaches and the underlying theoretical underpinning adopted by these approaches. While specific agile software development methods have been shown to provide innovation (Highsmith and Cockburn 2001) in fast-paced environments, we contend that the unique, resource-constrained, challenging, and dynamic environment provided by platform ecosystems are radically different from the fast-paced environments that are of interest in prior work. An intriguing future research direction could apply the framework of effectuation theory to practice-inspired agile methods. The effectual focus on emerging control portfolios could provide a promising theory-based structure for defining and distinguishing different agile approaches to software development.

The focus of the qualitative data analyses in this paper is to identify evidence that supports the current (perhaps, unconscious) use of effectuation in open-source development projects in the Apache Cordova environment. A limitation of this study is that our data analysis is limited to qualitative secondary data (42 user stories) for available open source projects. Specifically, the software development projects studied did not use effectual concepts and terms directly. Thus, the selected user stories required subjective coding and interpretation via an effectual lens. In order to address these limitations, we developed operational definitions for effectuation constructs in the software development context and updated them as the data analyses progressed. Also, stories selected for analyses provided extended discussion on the issue at hand. Based on these analyses, we did find considerable evidence that demonstrates the wide-spread use of effectuation thinking in the Apache Cordova projects. Future studies may adopt a case study-based approach for more in-depth study of effectual software development projects.

Future research directions will study how best to build software development methods that incorporate aspects of effectual thinking. We propose three potential areas of future research:

- A clear focus on novelty is essential for entrepreneurial application success on platforms. The effectual approach to software development supports a new way to think about novelty with a focus on existing means and aspirations of the software development team.
- The proposed effectuation design cycles found in the Fig. 4 research model will require considerable thought and development to transform into practical use. We envision that software development teams will perform *tight* effectuation design cycles that allow the project's actions and deliverables to emerge in rapid but controlled ways. We expect commonalities and differences with current agile methods to be identified and studied.
- We discuss and analyze the theoretical underpinning of software development approaches using the framework of control and prediction in Fig. 3. On-going research in the management of software development projects has considered pros and cons of different approaches and their suitability to specific environments. Across these studies, the assumption is that all team members follow a single approach throughout a project. However, based on our inferences from the framework of control and prediction and results from the qualitative study, we find that team members may adhere to different control techniques at different stages in the development project. Based on the application's current state of development, team members may alter their adherence to different software development approaches. Future research can study avenues and implications of balancing control and prediction during the progress of a project.

In conclusion, as organizations find themselves operating in uncertain, risky, and dynamic environments like digital platforms, existing approaches to software development have limited applicability due to their theoretical underpinning of prediction-based approach. Effectual approaches hold promise to address this gap by accepting, in fact, welcoming the uncertainty and risky environment, and incorporating its unique characteristics in its control-based approaches. This study provides the foundation for future work that explores control-based approaches and methods in software development and other areas of research pertinent for electronic markets.

**Acknowledgements** We gladly acknowledge the contributions made to this research by Rosann Webb Collins, Diana Hechavarria, Matthew Mullarkey, Richard Will, and Balaji Padanabhan. Partial funding was received from the Gaiennie endowment in the Muma College of Business at the University of South Florida.

## Appendix 1

The sample stories in the table provide examples of the coding performed in the qualitative data analyses

| Story description | Codes |
|---|---|
| with cordova-plugin-contacts 1.1.0 “Contacts”<br>When I get the whole list of contacts It’s working but when I choose: navigator.contacts.fieldType.phoneNumbers in the options.desiredFields<br>I get properly some first contacts, then (maybe because one of my contact phoneNumber value), I get this error:<br>Error in Success callbackId: Contacts598408154:<br>SyntaxError: Unexpected token u cordova.js:312<br>Cordova plugin contacts - PhoneNumbers error | Acceptable Risk - Commit limited resources<br>Action – Fixed bugs<br>Expanding cycle of resources – new technical knowledge<br>Means – Technology |
| Add the file plugin and browser platform (edge, from github or local repo) then cordova run browser gives the following in the console:<br>Error: exec proxy not found for:: File:: requestFileSystem<br>File plugin on browser platform causes “proxy not found” error on Chrome | Action – Fixed bugs<br>Expanding cycle of resources – new technical knowledge<br>Platform - Technology |
| Most of automatic geolocation tests were pended on Android because we didn’t have the tool to detect if the tests are running on a simulator or on a real device. Now we have device.isVirtual and can use it to pend the tests only on an emulator. Make geolocation tests use device plugin to properly detect Android suimulator | Means, Action, Aspiration – Novelty, application-platform match, Logic of Control |
| I came back from the Android Dev Summit, and sure enough, I forgot about the “Do not show me again” box on permissions. We need to handle this somehow and send a different exception. We should allow developers to tell users why they need the permission, otherwise their application experience will suffer. This will be an API addtion, not a change. I don’t believe we need to go up a major version for this. Cordova does not handle use case where we need to show rationale about permissions | Acceptable Risk, Aspirations, Expanding cycle of resources, converging cycle of constraints, means, platform |
| Under Adobe AIR, you can open a connection to a SQLite db and point to an existing file. The benefit of this is that your application can ship a database seeded with data. Without this support, your application has to initialize the db via scripting. While not difficult, it does increase the application’s first run time and also complicates the code unnecessarily. I understand that this isn’t per the Web SQL spec, http://dev.w3.org/html5/webdatabase, but it could certainly be useful. Support opening a database that connects to an existing file | Acceptable Risk, Aspirations, Expanding cycle of resources, converging cycle of constraints, means, platform |
| The ALAssetsLibrary framework has been deprecated in iOS 9, replaced by the Photos.framework. Once our minimum dependency is iOS 9, move to it.<br>Usage:<br>1. iOS (CDVURLProtocol)<br>2. Camera plugin<br>3. File plugin<br>4. File Transfer plugin<br>5. Local-Webserver plugin (cordova-plugins)<br>Update deprecated ALAssetsLibrary usage in plugins | Acceptable Risk, Aspirations, Expanding cycle of resources, converging cycle of constraints, means, platform, Logic of Control |
| MediaFile.getFormatData result data was empty (filled with default “0” values) for all types of capture: image, video audio. Problem encountered on Android iOS.<br>I solved this by changing the url passed to native code from localURL to fullPath.<br>Tested with two different Android phones (5.1 4.4) one iPhone 5 (iOS 9). The fix works! MediaFile.getFormatData does not work with localURL; changed with fullPath | Acceptable Risk, Action, Expanding cycle of resources, converging cycle of constraints, means, Logic of Control |

## Appendix 2

This appendix summarizes key aspects of software development approaches considered in this position paper

| | Plan-driven | Controlled-Flexible | Ad-hoc | Effectual |
|---|---|---|---|---|
| Assumption (Environment) | Stable, well-defined market boundaries, known competitors | Dynamic, Uncertain, ambiguous, blurred market boundaries and competitors | Dynamic, Uncertain, ambiguous, blurred market boundaries and competitors | Dynamic, Uncertain, Risky, Resource constrained, blurred market boundaries and competitors |
| Assumption (Market) | Mature market | Mature but fast evolving market | Mature but fast evolving market | Nascent market, mature but fast evolving |
| Project Execution | Linear | Iterative | Iterative | Iterative (effectual cycles) |
| Key Concepts | Prediction, resource gathering, milestones | Partial prediction, resource gathering, scope boundaries, ongoing feedback | constant calibration of its course by reacting to changes in the environment | Existing means, aspirations, acceptable risk, control, effects |
| Underlying Logic | Causation | Causation and adaptability | Rapid Adaptation | Effectuation |
| Process | Define outcome (specification) based on consumer needs, gather required resources, plan milestones, execute to realize the outcome | Define partial outcome based on consumer needs, gather required resources, set scope boundaries, monitor development with feedback | Development follows changes in consumer needs | Define partial outcome using aspirations, existing means, and consumer needs, identify multiple effects from existing resources, select effect that is controllable and has acceptable risk, monitor the development based on feedback from consumer needs, aspirations, and platform |
| Decision making | Systematic information gathering and analysis, expected return of the alternative | Systematic information gathering and analysis, expected return of the alternative, feedback | Expected return | Acceptable risk and logic of control |
| Outcome | Product-specification match | Product-market match | Working product | Product-market match, *product-platform match, value exceeding platform's core proposition*, novelty |
| Use of Knowledge | Rely on existing knowledge base | Synthesize existing and new knowledge (created with every iteration) | Act on new situational knowledge | Synthesize existing and new knowledge (created with every effect) |



## References

Alt, R., & Zimmermann, H. (2014). Electronic markets and general research. *Electronic Markets, 24*(3), 161–164.

Alt, R., & Zimmermann, H. (2015). Electronic markets on ecosystems and tourism. *Electronic Markets, 25*(3), 169–174.

Austin, R., & Devin, L. (2009). Weighing the benefits and costs of flexibility in making software: Toward a contingency theory of the determinants of development process design. *Information Systems Research, 20*(3), 462–477.

Baskerville, R., Heje-Pries, J., & Madsen, S. (2011). Post-agility: What follows a decade of agility? *Information and Software Technology, 53*(5), 543–555.

Benaroch, M., Lichtenstein, Y., & Robinson, K. (2006). Real options in information technology risk management: An empirical validation of risk-option relationships. *MIS Quarterly, 30*(4), 827–864.

Boudreau, K. (2012). Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. *Organization Science, 23*(5), 1409–1427.

Bygstad, B. (2016). Generative innovation: A comparison of lightweight and heavyweight IT. *Journal of Information Technology, 32*(2), 180–193.

Chandler, G. N., DeTienne, D. R., McKelvie, A., & Mumford, T. V. (2011). Causation and effectuation processes: A validation study. *Journal of Business Venturing, 26*(3), 375–390.

de Reuver, M., Sørensen, C., & Basole, R. (2017). The digital platform: A research agenda. *Journal of Information Technology, 33*(2), 124–135.

Drechsler, A., and Hevner, A.R. 2015. "Effectuation and its implications for socio-technical design science research in information systems," in: *DESRIST*. Dublin.

Flyvbjerg, B., and Budzier, A. 2011. "Why your IT project may be riskier than you think," *Harvard Business Review*).

Ghazawneh, A., & Henfridsson, O. (2015). A paradigmatic analysis of digital application marketplaces. *Journal of Information Technology, 30*(3), 198–208.

Gill, T.G., and Hevner, A.R. 2013. "A fitness-utility model for design science research," *ACM Transactions on Management Information Systems* (4:2).

Haile, N., & Altmann, J. (2016). Structural analysis of value creation in software service platforms. *Electronic Markets, 26*(2), 129–142.

Harris, M. L., Collins, R. W., & Hevner, A. R. (2009). Control of flexible software development under uncertainty. *Information Systems Research, 20*(3), 400–419.

Hayes, R.H. 1985. "Strategic planning-forward in reverse," *Harvard Business Review* (63:6).

Hevner, A. (2018). "Intellectual control of complexity in design science research," in A. Rai, "Editor's comments: Diversity of design science research,". *Management Information Systems Quarterly*, (41: 1), March 2017), iii–xviii.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer, 34*(9), 120–127.

Kim, W.C., and Mauborgne, R. 1997. Value innovation: The strategic logic of high growth. Harvard Business School.

Kirsch, L. J. (1997). Portfolios of control modes and IS project management. *Information Systems Research, 8*(3), 215–239.

Koch, S., & Bierbamer, M. (2016). Opening your product: Impact of user innovations and their distribution platform on video game success. *Electronic Markets, 26*(4), 357–368.

Lyytinen, K., Mathiassen, L., & Ropponen, J. (1998). Attention shaping and software risk— A categorical analysis of four classical risk. *Information Systems Research, 9*(3), 233–255.

MacCrimmon, K., & Wehrung, D. (1986). *Taking risks: The Management of Uncertainty*. New York: Free Press.

March, J., & Shapira, Z. (1987). Managerial perspectives on risk and risk taking. *Management Science, 33*(11), 1404–1418.

McKelvey, B., Tanriverdi, H., & Yoo, Y. (2015). Complexity and information systems research in the emerging digital world. *MIS Quarterly.*

Miles, M., Huberman, A.M., and Saldana, J. 2013. *Qualitative data analysis: A methods sourcebook*. SAGE Publications.

Parker, G.G., Van Alstyne, M.W., and Choudary, S.P. 2016. *Platform revolution*. W. W. Norton & company.

Perry, J. T., Chandler, G. N., & Markova, G. (2012). Entrepreneurial effectuation: A review and suggestions for future research. *Entrepreneurship Theory and Practice, 36*(4), 837–861.

Sarasvathy, S. (2001). Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *The Academy of Management Review, 26*(2), 243–263.

Stanley, K.O., and Lehman, J. 2015. Why greatness cannot be planned. Springer International Publishing.

Tiwana, A. (2013). *Platform ecosystems*. Morgan Kaufmann.

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research, 21*(4), 675–687.

Weiner, M., Mähring, M., Remus, U., & Saunders, C. (2016). Control configuration and control enactment in information systems projects: Review and expanded theoretical framework. *MIS Quarterly, 40*(3), 741–774.

Wiltbank, R., Dew, N., Read, S., & Sarasvathy, S. (2006). What to do next? The case for non-predictive strategy. *Strategic Management Journal, 27*, 981–998.

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research, 21*(4), 724–735.