Privacy by Design and Generativity

Candidate Number: 22955

Abstract

Both Generativity and Privacy by Design have been a subject of intense academic focus. However, there has not been much research done on the intersection of these domains. This essay argues that both these concepts have inherent conflicts and do not sit well in the context of Software Engineering. It first constructs comparable theoretical frameworks for both Generativity and Privacy by Design before delving into prescriptive and descriptive arguments to show their incongruity.

Introduction

On 9th January, 2007, Apple's CEO, Steve Jobs introduced the iPhone. A revolutionary smartphone that changed the computing industry forever. Giving the user an ability to install additional software without much hassle, its affordances were captivated only by one's imagination. Capitalizing on the wide array of sensors, third party developers created Applications to augment the functionality of the original device. This ability to add modular components and augment functionality make iPhone, and the subsequent smartphones, 'generative' devices.

On 5th June, 2013, Edward Snowden, a National Security Agency(NSA) contractor came out with stark revelations. Among other things, he claimed that governments have been using certain smartphones for spying on civilians. Bringing the issue of privacy to the forefront, Snowden's revelations forced us to consider how vulnerable we are in this digital pervasive world. It presented a new set of challenges for engineers to design privacy friendly artefacts.

One such approach is called Privacy by Design. It has received significant academic attention and has been adopted by the industry in practice. It is not the ultimate design methodology; neither is it without its drawbacks. However, it encapsulates key privacy principles and it is flexible in its approach. Thus it makes for a good framework for analyzing underlying privacy concepts. For the purposes of this essay, I would use the term privacy and Privacy by Design interchangeably.

In this essay I would argue that in the context of software engineering, generativity and Privacy by Design have conflicting ideas. I would further flesh out these conflicts and point to the various privacy tradeoffs that need to be made when designing generative artefacts. This essay solely looks at the effect of these domains under the purview of Software Engineering. Doing this limits the context for analysis but allows me to dive into the specifics of the domain and the underlying privacy principles.

First this essay will build on frameworks of Privacy by Design and Generativity. This will bring forth the key ideas and concepts in both of these domains. The next section will delve into specific arguments to prove my point. In the concluding section, I will tie these arguments together to further form a coherent case.

Framework for Privacy by Design

Privacy by Design (PbyD) is an approach to protect privacy by taking into account various privacy considerations when designing systems/services. It aims to minimize privacy risks by incorporating privacy throughout the life cycle of the artefact(Van Audenhove et al. 2011). Although PbyD is a multi-disciplinary approach, spanning over organizational, business and technological domains, this essay looks into the application of PbyD in Software Engineering. Focussing on the framework put forward by European Union Agency for Network and Information Security (ENISA) on Privacy and Data Protection by Design(Danezis et al. 2014), it analyses the implication of various aspects of this approach with additional context.

On the social end of the PbyD spectrum, there is existing discourse on the broad themes of design in privacy, including the legalities and societal perceptions. On the technical end, PbyD is viewed as implementation of specific 'Privacy Enhancing Technologies' (PETs) such as encryption and zero knowledge proofs. This essay takes a more balanced approach towards understanding privacy in software design and engineering. It focusses on design themes and privacy principles, keeping in accord with both social and technical views.

The key PbyD strategies can be broken into two broad categories, data-oriented and process oriented, albeit their subcategories do have some overlap.

Data Oriented Strategies

Data oriented strategies are key when designing IT artefacts. The techniques discussed here require very little user involvement and they can be baked into the design of the application itself. The literature around these strategies are primarily focused on their technical implementation rather than legal and societal nuances.

Kuner (2007) describes data minimization as "restricting the processing of personal data to minimum". The idea is to restrict data collection to a bare minimum without sacrificing functionality. Reduction in data collection and processing reduces privacy risks as well. It has been argued that data minimization should not just limit to our analogue understand of the minimum required data but include the state of the art technology as well (Gürses et al. 2011). An example this is the storage of cryptographically hashed passwords instead of their plain text versions for protection in the event of a data breach.

Another strategy to enhance PbyD is the usage of Data Hiding techniques. Data Hiding is referred to as the hiding of any personal data and their interrelationships from plain view(Danezis et al. 2014). It includes data which has been collected and stored securely. This is done to enhance confidentiality and its application is context specific. This strategy helps software engineers to design artefacts that are protected from security attacks on some vulnerability which was not anticipated. An example of this is uncovering usage patterns by analyzing meta data (which could have been potentially hidden).

Data Separation is the collection and processing of data in a distributed fashion. This plays in line with the concept of locality of data(Langheinrich 2001) which states that data should be tied to the point to where it is collected. The idea behind doing so is two fold, firstly it ensures that there is no central point of failure in the event of a privacy breach. Secondly, it ensures that by combining this data with more data points, more information about the user cannot be gleaned than what was intended.

Anonymity and Pseudonymity have been subject to intense academic focus. Anonymity refers to the state of not being identifiable within a set of subjects and Pseudonymity has been defined as the use of pseudonyms as identification(Pfitzmann & Köhntopp 2001). In systems where consent is tough to garner, anonymity and pseudonymity help in protection of privacy. These concepts are intricately linked to data minimization and it has been argued that anonymity and pseudonymity are consequences of this approach(Danezis et al. 2014).

The point of anonymizing data is lost if it can be linked back to an individual. This brings us to an import design strategy of making artefacts unlinkable. Unlinkability of two or more items means that within a system, these items are no more and no less related than they are related concerning the a-priori knowledge(Pfitzmann & Köhntopp 2001). However, whether data has been made sufficiently anonymized to make it unlinkable is often a matter of debate(Langheinrich 2001). Given some anonymized data in conjunction to other data points, it could be possible to link the data back to an individual with varying degrees of accuracy. Thus software engineers employing PbyD should keep in mind the degree of anonymization so as to make data "sufficiently" unlinkable.

Another concept that helps us think about privacy during artefact design is the concept of unobservability. Where anonymization and data minimization protect the linkages between users and data, unobservability relates to hiding user data itself. It has been referred to as making data indistinguishable from noise(Pfitzmann & Köhntopp 2001). This in part overlaps with the previously discussed Data Hiding technique.

Process Oriented Strategies

There are various process oriented decisions that need to be made during the design of IT artefacts. This section talks about strategies that could help develop artefacts keeping in mind the tenants of PbyD. The strategies discussed here require significant user participation and the literature around these strategies focusses on legal and social nuances of these policies and how they tie up to various privacy concepts.

One obvious strategy is consent. Taking the consent of users when storing or processing their personal information adds to the transparency of the system(Danezis et al. 2014). However, there are added complications when seeking consent of users. For consent to hold value, it is imperative that users know what they are consenting to. Many empirical studies(Good et al. 2007; Bakos et al. 2009) point to the fact that a majority of users do not read software agreements before consenting to them. There are cases where consenting to sharing personal data is required to access sources and services(Whitley 2009). In such cases, the notion of consent becomes complicated. Langheinrich (2001) argues that providing such binary choices when asking for consent is tantamount to blackmailing!

To truly empower users with privacy, control of the information they share should be accorded to them. The notion of this control is often drawn from data protection legislation. Users (in certain cases) are given the right to view, modify or even withdraw the information that they provide. For instance, in a landmark case (Case C-131/12 2014, Google Inc. v Agencia Española de Protección de Datos) tried in the Court of Justice of the European Union, it was ruled that users had the right to demand removal of search engine results if a search of their name was made. However, the notion of control goes beyond legal frameworks and includes the ease of control that users have over their data. Thus user interaction design plays a key role in the control of privacy(Danezis et al. 2014).

A privacy strategy needs to be enforced for it have any effect. Enforcing privacy strategies and monitoring for violations adds to the accountability of the system. Danezis (2014) argues that abiding to legal requirements should be the minimum to enforce privacy strategies, proper technical protection and appropriate governance structures must be be established for effective enforcement. FTC mandates proper recourse mechanisms to ensure privacy strategies are properly enforced.

Framework for Generativity

Generativity is "a technology's overall capacity to produce unprompted change driven by large, varied, and uncoordinated audiences"(Zittrain 2006). In IS literature, generativity has been analyzed though various perspectives. From an organizational perspective, there has been work done on how firms should adapt themselves to make generative products(Schilling 2000; Garud et al. 2003). Another body of literature presents a conceptual lens to understand generative platforms and the "paradox of control"(Tilson et al. 2010). Yoo (2013) calls for a sociomateriality view of generativity. There is another body of literature that analyses the effect and power of the internet as a generative platform(Zittrain 2006; Zittrain 2008).

To understand the relationship between Privacy by Design and generativity in the context of software engineering, it is important to deconstruct the existing frameworks of generativity. This section aims to present a coherent picture of key strategies in generativity, across frameworks, only to further draw parallels with the aforementioned framework for PbyD.

Data Oriented Strategies

Data lies at the heart of Software Engineering. The production, handling, use and distribution of data in certain ways contribute to the generative nature of an IT artefact. Presented here is a mix of descriptive and prescriptive strategies, shedding further light on the role of data in a generative context.

The digitization of different kinds of data (such as maps, music, books, etc.) is referred to as homogenization of data(Yoo, Henfridsson, et al. 2010). This transformation leads to the same standard of data being stored, processed and consumed across an array of similar digital devices. This separation of content from the medium has been key to making generative artefacts. Homogenized data further develop innovative properties such as traceability, interoperability and memorability(Kallinikos et al. 2013) which come with their own set of privacy risks.

Generative artefacts are responsible for creating and building upon large amounts of data. We are currently facing a sensor explosion(Zittrain 2008) due to the dwindling cost of sensors and data transmission. Generative products create a virtuous cycle of data creation and this data is further used to develop new generative products(Yoo, Henfridsson, et al. 2010). This phenomenon has been referred to as self-referentiality(Kallinikos 2006).

One important attribute of data in generative artefacts is recombinabiltiy. Recombinabiltiy refers to the ability of data to merge with other sources with relative ease and form new combinations.

The limitless possibility of “mashups” resulting due to recombinabiltiy of data leads to further, previously unimagined, innovations(Lessig 2008). This is in part due to data homogenization and self-referentiality. One often cited example of this is the Maps technology offered by Google. Its affordances were not thought of during the time of its development. However, due to the ease with which it can be recombined with other software artefacts, it has developed into a generative platform(Yoo, Henfridsson, et al. 2010).

There has been a wave of standardization that has been brought forth due to generative platforms. This closely ties to the idea of data homogeneity. Standards that have proliferated lower the barrier to entry for third parties to build upon generative platforms(Tilson et al. 2010). One example of this is the widespread adoption of the XML based Data Typed Definitions. They can be defined and used locally, yet global standards have emerged which allow easy recombinabiltiy.

Generative digital artefacts have a set of boundary resources called Application Programming Interfaces(APIs)(Yoo et al. 2013). These end points, together with certain rules and norms allow third party developers to use the underlying resources offered by the artefact. While such interactions are standardized, they are not planned or coordinated. They allow for new forms of control(Tilson et al. 2010) and provide a mechanism for uninhibited growth. Based on the principle of layered modularity(Yoo, Henfridsson, et al. 2010), APIs are used to form dynamic and complex ecosystems in a distributed fashion.

Process Oriented Strategies

The process by which generative software is engineered has been a subject of intense academic study. Often, software engineering processes determine the policy of governance of the digital artefacts. The governance policy in turn is key factor behind the degree and the kind of generativity the software artefacts exhibit. The strategies discussed here are tied closely to data oriented strategies and share certain characteristics with them.

Generative artefacts, by design, are never truly complete. This seemingly counterintuitive idea brought forward by Zittrain (2008) suggests that even though the generative products can be used as they are, they only realize their true potential when they are used in conjunction with peripheral services built on top of the original artefact. An example of this is the modern smartphone, which although highly functional straight out of the box, has applications installed on to it to add additional affordances. This was exemplified with Apple’s introduction of the Cut-Copy-Paste functionality on its iOS platform, several iterations after the initial launch(Cohen 2009).

Having generative artefacts that are not complete by design is advantageous. It allows software engineers to have lesser specification in their design and separates the design and production phases of the artefact(Henfridsson & Bygstad 2013). This flexibility allows the design to be receptive to customer needs for a longer period of time.

A key idea in the generativity literature introduced by Tilson et al. (2010) speaks about the ‘Paradox of Control’. Bringing an organization perspective into the literature, the paradox of control talks about the nature of control exercised by companies creating generative artefacts. On one hand, companies have to proliferate standards and set bounds to give further innovation a direction. On the other hand, companies should not exercise granular control as they might prohibit third party involvement. The socio-technical dynamic established is key for making a successful generative platform. This directly ties to the theme of centralization and decentralization of resources in the platform. This paradox directly manifests itself in the design of the APIs. They encapsulate norms, behavior and rules for interaction that decide the trajectory of the generative artefact. Changes in the dynamic of control can change the nature of the platform. A stark example of this is the difference in the mobile application store of Google's Android and Apple's iOS platforms(Eaton et al. 2011).

An architectural perspective for designing generative platform is the layered modular approach. Modularity is key in explaining the generative nature of products but the addition of an extra dimension of layer, makes the generative picture more coherent. It segregates artefacts into layers of content, service, network and device(Yoo, Henfridsson, et al. 2010). This segregation helps us see clearly the nature of generative products and their relationships. We are seeing the increasing use of this architecture in the automobile industry, as cars become increasingly modular at different layers, allowing users to add functionality(Amadeo 2016).

Arguments

Argument 1

Assertion

Data homogenization has led to newer and more complex privacy concerns.

Reasoning

The homogenization of varied analogue data into digital data has been both a cause and a characteristic behind generativity(Yoo, Henfridsson, et al. 2010) while revolutionizing our perception of IT artefacts in the last half a century. We have made constant and rapid progress towards digitizing information in various domains such as books, maps, images, sound, etc. Without going into the wonders of data homogenization, I am going to analyze to two key aspects of it and their respective effects on privacy.

One effect of data homogenization is 'Traceability'. It is defined as the ability of digitized artefacts to embed in them the relationship between itself and the actors. In other words, it is the capability to serially store, identify and process encounters between events and entities(Yoo, Lyytinen, et al. 2010). Traceability, although useful and empowering to software engineers, makes data minimization and data hiding more complex. When software developers exploit traceability, they inadvertently make users more vulnerable to privacy risks. When relationships between data entities and users is traced, it can often be reengineered to reveal unintended insights about user behavior.

The notion that traceability complicates privacy is best exemplified by the use of 'cookies'. Cookies are small pieces of data that are sent from a website and stored in the user's browser. Its intended use is to customize the web experience of users by keeping track of their preferences over time. However, this seemingly harmless feature reveals web browsing patterns of the user (which, arguably, constitutes as sensitive data). Furthermore, cookies have been used for targeted advertising at the expense of user privacy(Christina 2010). Amazon has been using cookies to show differential pricing for its products depending on their browsing behavior(Zittrain 2008). To address the issue, both data oriented and policy oriented strategies have been deployed. Browsers have baked in the 'Do-not-track' feature to protect user privacy(Data Hiding)(Gilbertson 2012). On the other hand, law makers have mandated explicit user consent when they visit websites which use cookies(ICO 2015).

Another effect of data homogenization is 'Memorability'. Due to the fact that information has ubiquitous representation (0's and 1's) irrespective of the context and content, it has become astronomically easier to store data(Yoo, Lyytinen, et al. 2010). The same memory devices can be used to store various kinds of information. Previously, we needed paper to hold text, film to hold videos and vinyl records to hold audio. Now, the same hard disk inside a modern day computer can hold all of these and more. Once again, despite being a key reason behind the 'Information Revolution', it has exacerbated privacy risks.

The concept of memorability has led to centralization of data. This flies against the concept of data separation. The same general purpose device, such as a smartphone, has the capability to store all kinds of personal information. This makes the device a single point of failure and escalates the privacy risk. Separation of data is unfeasible at the device level, thus software engineers should look into PETs like encryption and zero knowledge proofs.

Argument 2

Assertion

Sensor explosion coupled with mass data aggregation is detrimental to privacy.

Reasoning

Zittrain (2008) has shed light on the era of cheap sensors and the aggregation of peer produced data. He states that since 1970s, following Moore's law, processing power has become both faster and cheaper, exponentially. At the same time, the advancements in network technologies have allowed us to communicate between devices with greater efficiency. Building on these two phenomenon, he predicted that we were entering a phase of cheap sensors that are small and accurate at the same time. With the proliferation of 'Internet of Things' (IoT), his predictions have indeed come to pass.

Arguing about privacy risks related to sensors and IoT might seem tangential to generativity, however they are deeply linked. A key feature regarding generative platforms is the self-referentiality of data(Kallinikos 2006). Self-referentiality of data talks about how the production of data is a virtuous cycle i.e. data aids in the production of even more data. Generative platforms further leverage this to increase recombinabiltiy and allow further participation in the network. A recent example of this phenomenon is the health platforms launched by Apple and Google to aggregate personal health data(Dwoskin 2014). This data is further used by third party developers and medical professionals to deliver personal health insights(Amadeo 2014).

Sensor explosions has huge upsides, we are seeing regular objects, leveraging the power of the network and data to augment functionality. Cars equipped with proximity sensor are aiding in parking, the gyroscopes on smartphones monitors everyday activity, even smart thermostats that learns your preferences over time! However, they present huge privacy risks.

Sensors allow for the collection of vast amounts of personal data, often without proper consent. Consent gets more complicated as data becomes more personal. Often users agree for collection of private information in one context but generative platforms have the potential to use it in different, unintended, and sometimes unacceptable contexts. This conflict is best captured by the use of personal health data by insurers to provide better rates(Jurgens 2015).

Security is another huge concern with embedded sensors. Depending on the design, networked sensors are vulnerable to attacks. Since most of the sensors are linked to the same network, one vulnerable sensor could potentially infect the network. The data collected by these sensors often provide meaningful insights when paired with other data points. This recombinabiltiy of data is damaging to the notion of unobservability and unlinkability.

Sensor explosion presents huge design challenges for software engineers. Aside from baking in security measures in size and cost constrained devices, they also need to take into consideration the security issues raised by other vulnerable devices in the same network. Concerns pertaining to how their data is collected, secured and processed need to be addressed. Collectively, they have to engineer consent mechanisms where the user is made aware of both the use and potential uses of the data being collected.

Argument 3

Assertion

The nature of modularity in generativity and PbyD is fundamentally different.

Reasoning

The Oxford dictionary defines modularity as ‘Employing or involving a set of standardized parts or independent units as the basis of design or construction’. Starting from here, both PbyD and Generativity draw from this design principle in very different ways. I would argue this in two distinct bits. Firstly, I would explain how both of these concepts employ modularity as a core principle. Then I would proceed to draw the differences in their approach towards modularity.

PbyD does not explicitly advocate the use of modularity as a principle. However, in its implementation, key themes of modularity can be observed. The key idea behind Data Separation is the storage and processing of data in a distributed way. To create these data silos, it is important to deploy modularity when designing the data structure. Software engineers have to break down data into modules before deciding how they could be stored in a distributed fashion without breaking functionality(Hughes & Shmatikov 2004). The process of breaking up data before storing is called ‘Data Sharding’. Another reason to design modular systems under the paradigm of PbyD is access control. Software engineers use access control to limit access that different users have on the same system. Modular architectures aid in the design of access control by separating the system into discrete chunks before assigning access rights to different users and applications. For example, modern Operating Systems reserve certain critical actions for the owner (root user, administrator) of the system to provide added security.

A major portion of literature around generativity explicitly considers modularity as a key theme. Yoo, Henfridsson, et al. (2010) point to layered modular architectures as a way of understanding generative platforms. They propose breaking generative artefacts into layers and further subdividing layers into modules. The most interesting implication of this approach is Zittrain's view that modular products are never complete. They bank on external actors to augment functionality depending on user need. Having the capability of designing independent modules and attaching it to incomplete artefacts drives generativity. The idea of recombinabiltiy of data is also contingent on modular designs. The loose coupling of data wherein their combining it with other data adds newer context and meaning is part of an overall modular design.

The difference in modular approaches of generativity and PbyD is nuanced. On one hand, generativity uses modularity to open up and allow external participation. On the other hand, PbyD uses modularity to close down and restrict unauthorized usage. PbyD advocates strong control over design of the system to ensure security and privacy which goes against Zittrain's idea of starting with incomplete systems and depending on (often untrusted) external actors to add functionality. Generative modularity encourages combining data to gain newer insights whereas PbyD modularity advocates for data separation and data hiding.

Argument 4

Assertion

Control is perceived as widely different things in generativity and PbyD and even at their points of intersection, they contradict each other.

Reasoning

Control in PbyD is a policy oriented strategy that empowers users to take control of their personal information. The literature around control also talks about the nature and degree of accorded to users. Legal frameworks mandate a basic level of control in all products, however, PbyD encourages software engineers to go beyond this and also make this control simple and easy to exercise(Danezis et al. 2014). Consent and control are deeply interlinked as well. Similar empirical tests can be used to determine the effectiveness of both control and consent. For instance, the procedures and implication of revoking of control/consent act as good proxy in determining the efficacy of control(Whitley 2009).

Control in generativity, or lack thereof, refers to how open organizations make their generative platforms(Tilson et al. 2010). As pointed to before, there exists a paradox of control wherein firms have to set bounds and constraints and at the same time not exercise granular control (so as to discourage external participation).

This clears up the difference between the notion of control between generativity and PbyD. Control in generativity has to do with the 'openness' of the artefact and control in PbyD has to do with the power accorded to users over their own information. There are however some linkages between the varied interpretations of control, they have a negative effect on the other. To illustrate how organizations loosening control lead to restricted control for users, I would use the example of Google's Android and Apple's iOS operating systems.

Extensions of Arguments 3 and 4 under the mobile operating system paradigm

The mobile operating systems, Android and iOS, have been referred to as mankind's biggest architectural feat. Justifiably so, both of these operating systems have revolutionized how we interact with digital ecosystems. Interestingly, the central argument of this essay is best exemplified in the analysis of both of these platforms.

Analyzed through the lens of Paradox of Control, iOS exercises strict control over the platform when compared to Android (Tilson et al. 2012). This is exhibited through various policies and their different modular architectures. Putting this in tandem with a comparative study(Han et al. 2013) of the security features of Android and iOS, we see how control and modularity shape privacy and generativity.

Apple has a dedicated team to review application submissions before users can download them. Google on the other hand performs automated security checks before putting an application on their store. By putting applications through greater scrutiny, Apple ensures that applications on their store are secure and do not pose a privacy risk. Due to lax control, Android offers more choice for users on their store than iOS.

Application sandboxing is a modular architectural principle wherein each separate application has restricted permission in their ability to interact with the other applications and the Operating System at large. iOS uses application sandboxing while Android does not. This limits the damage if a malicious application is introduced in the system. This however severely restricts generativity and disallows applications to augment functionality onto one another. For instance, iOS disallowed third party keyboards for years in fear of compromising security(Olivarez-Giles 2014). This reinforces my argument about how modularity differs in the content of generativity and privacy.

API design in both of these operating system reveal the difference in the nature of control and modularity. Android APIs are more powerful than their iOS counterparts because of the high degree of system flexibility it enjoys. It is relatively easier for Android applications to leverage APIs to make system wide changes. By not exercising control, Google has pushed for a wider adoption across many device manufacturers allowing for enthusiasts to tinker with their devices. On the downside, Android has been targeted with malwares and virus which compromise device security(Ahmad et al. 2013).

Conclusion

Arguments 1 and 2 bring out descriptive implications of generativity and prove how they go against various PbyD principles. In these arguments, potential solutions (or at least the existing problems) to existing risks privacy risks are fleshed out. However, these solutions are tougher to implement than they are made out to be. Aside from huge technical issues, markets often constrain software engineers to skimp on privacy friendly technologies in favour of cheaper and generative solutions.

Arguments 3 and 4 take prescriptive characteristics to show how PbyD and Generativity treat similar concepts in different veins. This only becomes clearer when these principles are elucidated upon when comparing Android and iOS. The tradeoffs between choosing generativity over privacy and vice-versa become clear. The presence of these tradeoffs reinforce the idea of generativity and PbyD having ideological conflicts.

Standardization is one aspect of generativity that aids in PbyD. The lack of standardization hurts software engineers to build privacy friendly solutions because of the larger realm of standards they need to cater to(Pearson & Benameur 2010). Generativity proliferates standardization of various aspects(Yoo, Henfridsson, et al. 2010). For instance, having similar cryptographic standards across platforms would help engineers bake in encryption without running the risk of breaking functionality.

There are a few distinct limitations of this essay. Both PbyD and generativity are ginormous domains and have implications and uses above and beyond software engineering. This essay only focuses on the design aspect of software engineering when looking at both of these domains. Even though the key theme might be evident in different aspects of these domains, their implications may be very different.

Another possible limitation is the implicit understanding of privacy as an ideal. With our changing attitudes towards privacy, deemed as Privacy 2.0 by Zittrain, many arguments made by the essay and most of the cited literature would have to cede considerable ground. Societal perception to privacy is adapting to the digital world. Collectively, people find data collection and sharing increasingly acceptable(Zittrain 2008). If we are moving towards this ideal, recalibrating our notion of privacy would help us better understand generative artefacts and how we can design for privacy.

These limitations aside, the analysis of literature coupled with examples from the industry overwhelmingly bring out the clash between key ideas behind Generativity and Privacy. Prioritizing one over the other when designing Software Engineering artefacts would have grave consequences. Further research is needed to point towards the costs of prioritizing and a framework to maintain the balance between these opposing concepts.

References

Ahmad, M.S. et al., 2013. Comparison between android and iOS Operating System in terms of security. In *Information Technology in Asia (CITA), 2013 8th International Conference on*. IEEE, pp. 1–4.

Amadeo, R., 2016. CarPlay vs. Android Auto: Different approaches, same goal. *ArsTechnica*.

Amadeo, R., 2014. Google releases Google Fit SDK along with special version of Android L. *ArsTechnica*.

Van Audenhove, L. et al., 2011. Privacy by Design: an alternative to existing practice in safeguarding privacy. *info*, 13(6), pp.55–68.

Bakos, Y., Marotta-Wurgler, F. & Trossen, D., 2009. Does Anyone Read the Fine Print? Testing a Law and Economics Approach to Standard Form Contracts. *IDEAS Working Paper Series from RePEc*.

Christina, T., 2010. How Advertisers Use Internet Cookies to Track You. *Wall Street Journal*. Available at: http://www.wsj.com/video/how-advertisers-use-internet-cookies-to-track-you/92E525EB-9E4A-4399-817D-8C4E6EF68F93.html [Accessed February 23, 2016].

Cohen, P., 2009. Cut and Paste, MMS Highlight IPhone 3.0 Improvements. *PCWorld*.

Danezis, G. et al., 2014. Privacy and Data Protection by Design - from policy to engineering.

Dwoskin, E., 2014. Apple's Next Big Focus: Your Health. *The Wall Street Journal*. Available at: http://www.wsj.com/articles/apples-next-big-focus-your-health-1409959072.

Eaton, B. et al., 2011. *Dynamic structures of control and generativity in digital ecosystem service innovation: the cases of the Apple and Google mobile app stores*,

Garud, R., Kumaraswamy, A. & Langlois, R., 2003. Managing in the Age of Modularity: Architectures, Networks, and Organizations.

Gilbertson, S., 2012. Google Chrome Adds "Do Not Track" Privacy Tools. Available at: http://www.wired.com/2012/11/chrome-23-skidoos-out-of-beta/.

Good, N.S. et al., 2007. Noticing notice: a large-scale experiment on the timing of software license agreements. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 607–616.

Gürses, S., Troncoso, C. & Diaz, C., 2011. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14, p.3.

Han, J. et al., 2013. Comparing mobile privacy protection through cross-platform applications.

Henfridsson, O. & Bygstad, B., 2013. The Generative Mechanisms of Digital Infrastructure Evolution. *MIS Quarterly*, 37(3), p.907.

Hughes, D. & Shmatikov, V., 2004. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer security*, 12(1), pp.3–36.

ICO, 2015. Guide to the Privacy and Electronic Communications Regulations. Available at: https://ico.org.uk/media/for-organisations/guide-to-pecr-2-1.pdf.

Jurgens, B., 2015. How IoT Will Change Your Relationship With Insurance. *TechCrunch*.

Kallinikos, J., 2006. *The consequences of information : institutional implications of technological change*, Northampton, MA: Northampton, MA : Edward Elgar.

Kallinikos, J., Aaltonen, A. & Marton, A., 2013. The Ambivalent Ontology of Digital Artifacts. *Mis Quarterly*, 37(2), pp.357–370.

Kuner, C., 2007. *European data protection law: corporate compliance and regulation*, Oxford University Press.

Langheinrich, M., 2001. Privacy by design—principles of privacy-aware ubiquitous systems. In *Ubicomp 2001: Ubiquitous Computing*. Springer, pp. 273–291.

Lessig, L., 2008. *Remix: Making art and commerce thrive in the hybrid economy*, Penguin.

Olivarez-Giles, N., 2014. The iOS 8 Features We're Most Excited About. *Wall Street Journal*.

Pearson, S. & Benameur, A., 2010. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, pp. 693–702.

Pfitzmann, A. & Köhntopp, M., 2001. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*. Springer, pp. 1–9.

Schilling, M.A., 2000. Toward a general modular systems theory and its application to interfirm product modularity. *Academy of management review*, 25(2), pp.312–334.

Tilson, D., Lyytinen, K. & Sørensen, C., 2010. Research commentary-digital infrastructures: the missing IS research agenda. *Information systems research*, 21(4), pp.748–759.

Tilson, D., Sorensen, C. & Lyytinen, K., 2012. Change and Control Paradoxes in Mobile Infrastructure Innovation: The Android and iOS Mobile Operating Systems Cases. *System Science (HICSS), 2012 45th Hawaii International Conference on*, pp.1324–1333.

Whitley, E.A., 2009. Informational privacy, consent and the "control" of personal data. *Information security technical report*, 14(3), pp.154–159.

Yoo, Y. et al., 2013. The Generative Architecture of a Digital Ecosystem: A Network Perspective.

Yoo, Y., Lyytinen, K.J., et al., 2010. The Next Wave of Digital Innovation: Opportunities and Challenges: A Report on the Research Workshop'Digital Challenges in Innovation Research'. *Available at SSRN 1622170*.

Yoo, Y., 2013. The Tables Have Turned: How Can the Information Systems Field Contribute to Technology and Innovation Management Research? *Journal of the Association for Information Systems*, 14(5), pp.227–236.

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

Zittrain, J., 2008. *The future of the internet--and how to stop it*, Yale University Press.

Zittrain, J.L., 2006. The generative internet. *Harvard Law Review*, pp.1974–2040.