Researching educational apps: Ecologies, technologies, subjectivities and learning regimes

Mathias Decuypere
KU Leuven, Belgium

mathias.decuypere@kuleuven.be

Keywords: Apps, Science and technology studies, Learning to code

To cite this article:

Decuypere, M. (2019). Researching educational apps: Ecologies, technologies, subjectivities and learning regimes. *Learning, Media and Technology*. DOI:10.1080/17439884.2019.1667824

Available at:

https://doi.org/10.1080/17439884.2019.1667824

Abstract

Apps are becoming increasingly important in education. However, critical educational research has hardly ever undertaken up-close analyses of educational apps. This article provides an extensive analysis of one 'learning to code' app called Grasshopper. Drawing from the field of Science and Technology Studies, the article adopts a theoretical framework that situates app analyses at the dimensions of apps' situated ecologies, their platform and algorithmic technologies, the enacted user subjectivities and, as the sum of these previous dimensions, the projected learning regimes. Making use of the methodological entry points of app websites, stores and interfaces, the article makes visible and analyzes the app's infrastructural settings, actively invokes different app situations, and uses these to advance the inquiry by offering a multidimensional point of view that navigates different scalar dimensions of educational apps. In doing so, the article examines learning to code through its concrete sociomaterial practices, and the specific pedagogies and sorts of education that are employed in order to bring about such learning.

Introduction

With the dominant app distribution stores having debuted only in 2008, apps have very quickly become mundane software, and it is precisely because of this mundaneness that they are now situated in nearly each aspect and facet of contemporary society, including education (Morris and Elkins 2015). Educational apps are a distinctive category in app stores, tailored to all levels of the educational field (from kindergarten to higher education; from formal to non- and informal learning) and comprising over hundreds of thousands of apps with many different forms of projected usage (Zosh et al. 2017). . The educational app field is characterized by a considerable diversity, and is often roughly divided into three general categories: apps that focus on the acquisition and training of specific *skills* (for instance aiming at rote memorization or skill-and-drill exercises), of providing designated *content* (giving access to and enabling the study of specific types of data, information, or knowledge) or of offering specific *functions* (assisting students in productively performing tasks in the classroom) (Cherner, Dix, and Lee 2014). As far as skill- and content-based apps is concerned, their specialized nature generally implies that the type of learning that is envisaged is highly specific and focused on delineated and predetermined aspects or subjects. This has been designated as *microlearning*: learning through relatively delimited amounts of information, characterized by quick interactions between app and learner, and by dividing learning activities into small, manageable chunks that can be consumed 'on the go' (Godwin-Jones 2011). Next to these skill- and content-based apps, other apps are tailored to the enhancement of pupils' and teachers' productivity (e.g. Google's popular G Suite app bundle). Such function-based apps are not directly targeting the learning of pupils and students, but rather assist in creating digital artifacts such as textual and/or visual documents, multimedia presentations, etc. (Cherner, Dix, and Lee 2014; Lindh and Nolin 2016). In addition hereto, a fourth group of educational apps, which seek to manage pupils' behavior and aim to tie the classroom with the parental home (e.g. ClassDojo), has recently emerged (Williamson 2017).

At the same time, the proliferation and sheer abundance of educational apps has given rise to several concerns. Many apps are unclear, unreflexive or untested as far as their educational design and underlying didactic philosophies is concerned, leading Hirsh-Pasek et al. (2015:3) to the conclusion that the educational app industry in practice amounts to 'a vast, unplanned experiment' on its – often very young – user base (see equally Callaghan and Reich 2018). Equally, concerns have been raised in view of the often opaque business models of apps, and more particularly of apps' tendencies to surreptitiously collect and mine user data for commercial purposes, thereby turning private data into an exploitable resource (Herold 2016; Lindh and Nolin 2016). Furthermore, it has been argued that skill- and content-based apps tend to reduce education and learning to its most traditional, salient and visible components such as the test, the quiz or the lecture, thus offering environments that are largely built on technological affordances rather than on educational rationales (Salomon 2016).

Despite these important general concerns, critical educational research has not yet often been directed towards an up-close study of educational apps. Curiously, whereas recent years have seen a general upsurge of critical studies that seek to analyze (and counter) the instrumentation, instrumentalism and rhetoric reigning in the broad field of educational technology (Eynon 2018), these studies have largely sidestepped the specific field of educational apps. Whilst some arguments about the educational relevance of app analysis have already been made (e.g. Albury et al. 2017) and whilst educational apps have already been the subject of some critical scrutiny (e.g. Williamson 2017), critical studies of educational apps are still few and far between.

This article makes an extensive analysis of one educational app called *Grasshopper*. Grasshopper is a mobile microlearning app targeting mature individual users. It is a free application destined to teach its users how to code in JavaScript, and is exclusively available on mobile devices (Android & iOS). Grasshopper does have a website, however, where some general information is being given about the aim and functionalities of the app and its projected user base.[1] Grasshopper has been named likewise as a tribute to Grace Hopper, who was an early pioneer

[1] https://grasshopper.codes/

in computer programming. The app is developed by Area 120, a division of Google that is circumscribed as a workshop developing speculative projects and experimental products.[2] This makes Grasshopper related to, but different from Google's broader attempts to offer initiatives with educational purposes to schools, parents, classes and pupils the like (Williamson 2016b; Lindh and Nolin 2016). Grasshopper launched in June 2017, and has since then been downloaded over a million times.[3]

In analyzing Grasshopper, this article is situated within a growing body of studies that take a critical interest in the contemporary upsurge of 'learning to code' cultures (e.g. Kitchin and Dodge 2011; Edwards and Carmichael 2012). Together with other major tech giants, over the last years Google has been promoting 'learning to code' both by trying to influence educational policies to include coding in educational curricula and by offering financial stimuli to specific initiatives that stimulate a learning to code agenda. As Williamson (2016a:41) states, the learning to code movement 'has been translated from a grassroots campaign into a relatively stable and coherent policy agenda in a remarkably concentrated period, yet the actors mobilizing it into education policy, the material practices of coding promoted through its pedagogies and its wider connections to changing techniques of governance remain under-researched'. Rather than focusing on large-scale agendas, this article focuses on these concrete (socio)material practices and pedagogies of coding, and does so by investigating the ecologies, technologies, subjectivities and learning regimes present in a learning to code app.

STS and apps: Ecologies, technologies, subjectivities, and learning regimes

The theoretical approach of this paper is grounded in Science and Technology Studies (STS). One of the central characteristics of STS is their adoption of a relational ontology, which does not prioritize social or material dimensions in researching a setting (such as an educational app). Instead of focusing exclusively on social and/or material dimensions, these approaches place analytical focus on the relations between actors, and contend that it are precisely these relations that are constitutive in the shaping of sociality, materiality and/or technicality (Decuypere and Simons 2016; Latour 2005). That is to say, the point of departure is that practices are never pregiven – rather, they are *shaped* in and through the relations that are formed between different actors. Practices, thus, do not exist prior to their operations: they are realized or *enacted* in the course of their effectuation (Woolgar and Lezaun 2013). As such, STS operates as sensitizing device that allows to focus on the relational features of apps – in particular how the relational interplay between technologies and individuals shapes and makes different sorts of users, forms of learning, and conceptions about education (Decuypere 2019a; Fuller 2006).

Accordingly, STS considers apps as socio-technical devices, that is, as assemblages of actors that act and make their users act. Apps *act*: they convey particular messages about education and learning, stage the learning process in specific ways, embed sociality through dedicated plug-ins, give direct feedback to the learner through algorithms and machinic leaning, etc. In doing so, apps aim to create and prototype digital forms of education that are appropriate for achieving the learning they envisage (such as learning to code). Yet, these actions need to be tied to what these doings *make their users do*: through acting, apps install and appeal to specific self-understandings of learners making use of these apps. In other words, apps equally *enact*: they attempt to make students think in particular desired ways, seek to understand themselves and the world in certain predefined manners, and consequentially inscribe not only particular visions about education and learning and what it means to learn and be a learner today, but equally install a way of acting upon that world (Callon and Muniesa 2005; Williamson 2016b; Decuypere 2019b). In that respect, STS considers apps not merely as the technological backdrop that allows social dynamics or forms of learning to take place. Apps are not neutral and not to be considered as 'simply there, irrelevant, or

[2] https://area120.google.com/

[3] As with all apps, usage statistics are difficult to obtain, but some information about the Android version can be found here: https://www.appbrain.com/app/grasshopper%3A-learn-to-code-for-free/com.area120.grasshopper

designed in only one way imaginable' (Gillespie 2015:1). Rather, of central importance is to focus on what apps and their constituent socio-technical elements are doing as part of situated practices. In sum, apps are not just simply in the world, *they make different sorts of worlds* – i.e. they enact specific practices, sorts of users and forms of learning.

In order to disentangle the socio-technical operations performed by apps, STS foregrounds four theoretical dimensions that constitute the framework of this article. First, the *ecologies* of apps can be scrutinized, that is, their situatedness within larger wholes (e.g. websites, apps stores) and what this relational embeddedness and infrastructural positioning tells us about how apps are inscribed and positioned within distinct thematic, discursive, and algorithmically-tailored and curated fields. Put differently, apps never exist solely on themselves but are always ecologically embedded in fields that partly shape and stage how an app needs to be approached and understood (Hörl 2017; Williamson 2017). Next to ecologies are the *technologies* that are operational in app practices, such as algorithmic and platform technologies. Whereas these technologies are increasingly argued to be dispersed, not directly amenable to scrutiny, and opaque from the point of view of their technical details or their underlying operations (Ziewitz 2016), STS focuses not so much on what algorithms and platforms *are* as on what they *do* (Introna 2016; Bucher 2018). Platforms should be understood in the architectural sense of the word here, i.e. as mediating structures that are implicated in establishing the design of apps, and more particularly in how they are being composed and consisting of different areas (e.g. communicative, communal vs. instructional, individual) within the app (Gillespie 2010). As such, the attention is not so much on the technical intricacies of an app's architecture, but rather on the effects that this platform architecture generates. Likewise, the analytical focus is not on the algorithm itself or where it needs to be situated precisely. Rather, the focus is on how these algorithmic technologies, once operational, start to make different versions of reality (Bucher 2018:56). A third dimension consists of the *subjectivities* that are being shaped in relation to these ecologies and technologies. Apps do not function in a vacuum: they have particular sorts of projected users in mind and, hence, always have a specific understanding about ideal learners or students. These ideal users are not only assumed, they are equally actively being created: technological environments such as app interfaces actively *configure* or *script* particular kind of users (Woolgar 1990). A fourth and last dimension – and as the result of the previous three dimensions – is that educational apps always install a particular *learning regime* within the confines of the app itself: a particular understanding about what learning is and about the specific forms of learning that are considered to be worthwhile – actively shaped in and inscribed through app functionalities (Decuypere 2019b; Williamson 2017).

Together, these four dimensions form a heuristic vantage point that can be deployed in order to scrutinize educational apps in an up-close and critical manner. In the next section, we elucidate the methodological design constructed based on these four dimensions.

Multi-situated app studies: Three entry points

In this article, we adopt three methodological entry points of inquiry in order to scrutinize the multi-situatedness of apps. We start with a brief analysis – combining and juxtaposing multimodal textual, visual and navigational elements – of the app's *website* to disentangle how Grasshopper is being positioned in, and staged by, a specific discursive-semiotic field (Knox and Bayne 2013; Decuypere 2016). After that, we adopt two entry points as suggested by Dieter et al. (2018), namely that of the app store (i.e. how apps are relationally embedded within a store's infrastructure) and that of the app interface (i.e. what happens on the screen when one is using an app). Acting as obligatory passage points (Callon 1986), *app stores* function as 'key gatekeepers (…) by setting up the rules for app creation, sorting, and distribution' (Dieter et al. 2018:2). Researching how app stores index apps and sort them into various categories 'opens possibilities to generate research situations that provide insights into the relational and infrastructural situatedness of apps' (ibid.:4). Apps, hence, are not to be conceived as having a singular, fixed form, but rather as practices that can be situated in a variety of places, connected to different categories, as well as related to many possible topics. This multiple situatedness gives us the means to visualize the

ecological (store) milieu in which educational apps are situated. In order to do so, we made use of the *Google Play Similar Apps* tool developed by the Digital Methods Initiative.[4] This tool extracts details of individual apps and collects similar apps that are suggested by the store and does so, importantly, without being tied to a singular user's individual data and digital identity.[5] First, the tool generated 50 similar apps to Grasshopper. In a second move, we ran the tool again but this time taking the apps generated in the first step as point of departure. This resulted in a network that showcases a relational field of more than 700 similar apps, which provides insight in Grasshopper's surrounding milieu, that is, how it is relationally embedded and positioned within a broader field of similar apps.[6]

While the previous entry points allow to research an app's situated ecologies by focusing on the discursive-semiotic staging of apps (site) and how they are algorithmically brought together and ordered (store), the third entry point focuses on the operations performed within the *interface* of the app itself (once downloaded). Concretely, we made use of the walkthrough method, a methodological approach to study app environments that is situated within the broad STS tradition (Light, Burgess, and Duguay 2018). The walkthrough method consists of a direct engagement with the app's interface. One of its most salient features is that it *slows down* (inter)actions that are associated with normal app use: whereas normal app use is characterized by fast and fickle user engagement, the walkthrough method allows to analyze an app's architecture and interface in a systematic way in order to disentangle the app's *environment of expected use* and foreground the characteristics of the interface (Dieter et al. 2018). The term 'environment of expected use' is used in order to refer to what app providers anticipate to be typical (and desired) user behavior and more broadly their intentions and visions illuminated through an app's interface. Specifically, walking through apps 'requires the researcher assume a user's position while applying an analytical eye to the process of acquiring the app, registering, accessing features and functionalities and discontinuing use' (Light, Burgess, and Duguay 2018:891). Reminiscent of STS ethnographies, the walkthrough was effectuated through extensive periods of observation, where we visited the app on a daily basis for a period of five weeks.[7] During these five weeks, the researcher participated as a user who followed one core module of the app ('fundamentals') and who visited the social sections of the app on a passive basis (i.e., without actively contributing to discussions). Three notebooks were used in the process: a notebook consisting of fieldnotes (notes and memos supplemented with screenshots documenting the research process) generated on a continual basis; a notebook acting as logbook of the observed events; and a notebook that sought to give first accounts of what was being observed (Latour 2005). In our walking through Grasshopper, we focused on algorithmic and platform technologies, processes of subjectification through user configuration and the installment of desired forms of learning present in and enacted through the app. In that respect, the focus was not only on textual elements, but equally on visual elements (pictures, buttons, menu structures, etc.) and more importantly on how form, style and content constantly interact with each other in order to shape specific users, forms of learning, and conceptions about education (Law 2009; Decuypere 2016).

[4] https://wiki.digitalmethods.net/Dmi/ToolGooglePlaySimilar/

[5] In other words, the Google Play Similar Apps tool allows to investigate which apps are algorithmically associated with which other apps *without* making use of a user's personal data. Our analysis is not immediately directed at inquiring the mutability of algorithms based on personal recommender systems – we do not intend to analyze how algorithms' output shifts, and is co-shaped by, users' moving through data (Mackenzie 2015). Rather, of particular interest here is the agency that is present within algorithmic ordering itself *in so far as* this ordering is displaying the relational embeddedness of the app in a store network of similar apps. Whilst stabilizing the mutability of algorithmic agency, this allows to make the processes of algorithmic associating, and the effects that such associating brings about, amenable to empirical scrutiny (Dieter et al. 2018; Ziewitz 2016).

[6] Network visualizations were constructed in Gephi (gephi.org) using the ForceAtlas algorithm (Bastian, Heymann, and Jacomy 2009). Figure 2 shows this network with a filter of 2, i.e., only apps with more than one relation are being displayed. Size of the nodes is set proportionally to their betweenness centrality. In order to perform a visual network analysis, app type was manually categorized (and colored) based on app's descriptions as found in the Play Store (Decuypere, 2019c).

[7] Between September and October 2018. All footage discussed in this article (website, store and interface) was collected within that timeframe. Figure 1 consists of two screenshots from the Grasshopper website (https://grasshopper.codes/); Figures 3–6 consists of screenshots taken within the Grasshopper app itself.

Combined, the three entry points allow to make visible and analyze apps' infrastructural settings, to actively invoke different app situations, and to use these to advance our inquiry by offering a multidimensional point of view that navigates different scalar dimensions of educational apps.

Website

On its website, Grasshopper is variously staged. To start with, the website presents the app as a *persuading device* that seeks to convince its users of being capable to successfully complete a JavaScript coding journey. Both textually and visually – displaying routes to follow, mountains to climb, and bridges to cross – the website stages Grasshopper as a device that will assist its users in completing their proverbial journey and that will, eventually, allow one to 'graduate' with fundamental coding skills (Figure 1a). Naturally, Area 120 does not have the means to issue official diplomas. However, the usage of traditional educational terms such as 'graduating' and coding 'curricula' seeks to convince users of the possibility of positively completing a 'coding journey' (cf. infra). At the same time, persuading is equally effectuated by stressing prospective app users that coding launches new careers, expands one's network, and opens new doors. Interestingly, at this point the website invokes the image of the diploma, but this time in a negative way: in the coding profession, diploma's *aren't* needed. This conveys the message that the professional coding sector is less rigid, more innovative and more welcoming for new developments, and hence, more empowering, than the traditional educational sector (cf. the visual depiction of an open door and the textual description 'newer learners like you' in 1b). The coding sector, thus, is promoted as a professional sector in which learning through Grasshopper allegedly suffices to become more employable. In that sense, the website enacts a specific technocratic imaginary, infusing thinking about learning with assumptions that apps outdate traditional educational systems in favor of offering quick and easy technological solutions to specific challenges (such as learning to code) (Williamson 2016b).

Promotion and persuasion are furthered by framing the app as a *playful device*: time and again, coding is portrayed as fun, as an addictive playground or as a skill that everybody can master. Additionally, the website stages Grasshopper as a *teacher* who is giving you personal feedback and who is there to advance your own personal learning. This reinforces the imaginary that the technicities and intricacies of learning to code shouldn't preclude the opportunity to make it enjoyable, brake it into digestible parts, and tailor all of this to the personal requirements of the individual learning, thus solving the challenge of learning to code easily with the assistance of a simple, straightforward app. The effect of such visual and textual staging is that Grasshopper is being elevated to an authoritative position, because it guarantees to make learning to code at once educational and employable, and digestible and playful. As such, the website stages Grasshopper as an *achievement technology*, assuming that coding knowledge is singular and universal, and that the pace at which one acquires such knowledge can be expanded and accelerated by making learning an enjoyable, fun, and easy endeavor (Chang 2019).

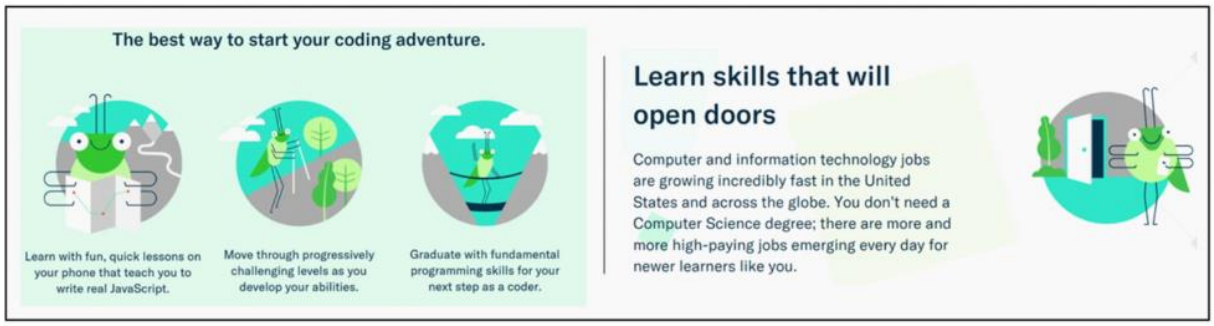

Figure1. Grasshopper framed as navigating (left; a) and as empowering (right; b) device.

Thus far, we only discussed visual and textual elements present on the website. Interestingly, Grasshopper's website is nearly entirely self-contained (i.e., does not refer to other websites). However, in one designated place the website explicitly navigates the user to an external website. On its FAQ-section, where the question is raised what

to do after having used Grasshopper, the website links to the MOOC platform Coursera and states that "We've partnered with Coursera… Coursera is a revenue sharing affiliate partner, so if you sign up, you'll be helping us continue to grow Grasshopper. Thanks.".[8] At this point, then, the website is not only staging Coursera as authoritative platform; it is at once showing a glimpse of Grasshopper's economical functioning and business model (see below) *and* positioning Grasshopper within a broader ecology of (open) educational initiatives (Decuypere, 2019b).

Store

This discursive-semiotic staging is being reinforced once one enters Grasshopper's app store page, iterating the same directional ('begin your coding journey'), playful ('move through progressively challenging levels') and personalizing ('real time feedback guides you like a teacher'; 'we can't wait to meet you') utterances.[9] Next hereto, another dimension of Grasshopper's ecology consists of how it is algorithmically ordered within the store, that is, which other apps it presumably resembles (the 'similar apps'-section of an app page in the store). Developers play a key role here: they can assign their app to several designated categories. Next to developers, equally general usage patterns play a role: users' engagement with similar sorts of apps are co-determining how resembling apps are algorithmically ordered (Dieter et al. 2018). As outlined above, we created a Grasshopper similarity network (Figure 2), which displays how Grasshopper is ecologically positioned within a relational web of similar apps. Not surprisingly, the majority of apps are coding and programming apps (purple; main body of Figure 2). Most of these coding apps are equally focusing on learning to program in Java. It is equally clear that Grasshopper takes a central position and is very regularly referred to. Yet, algorithmic store ordering equally associates other app categories as similar: open education and generic lifelong learning apps (orange; left-hand side), language learning apps (green; primarily bottom left corner), apps related to programming microcontrollers (pink; top), picture-taking and processing apps (blue; scattered throughout). Displaying the relational infrastructure of Grasshopper (and the organizational logic of the store itself), Figure 2 not only shows how coding apps are related to each other, it equally shows the algorithmic associations of coding apps with other categories. The point here, then, is not to make a general case for the alleged powerfulness of algorithms (Ziewitz, 2016). Instead, Figure 2 concretely shows how algorithmic store ordering, *irrespective of individual recommender systems*, makes learning to code similar to presumably neighboring areas as language learning, tinkering with microcontrollers (e.g. Raspberry Pi), lifelong learning, etc. – thus creating an ordering which is contingent yet significant. The crux of the matter is that this ordering is not innocuous but impactful: it determines what algorithms deem to be *equivalent* (sorts of) apps, based on developer feedback and general user patterns. Thus, such ordering enacts a specific *space of equivalence* in which coding apps are not only associated with other coding apps, but equally with specific other categories (implying at once that the mentioned categories such as language learning are as technical as learning to code is, and that non-mentioned app categories are not of interest to the user browsing similar apps).

This interpretive flexibility in how Grasshopper is algorithmically being portrayed and made equivalent, makes that it functions as a *boundary object*. For instance, it is at once portrayed as an instantiation of learning to code and of lifelong learning apps, or to say it more forcefully, it is assigned to, but at once equally occupies and creates, a boundary space it shares with other apps (Star 2010). In sum, and next to app staging through persuasion and promotion, store ordering illustrates that singular apps do not have one predetermined form, and need to be investigated from a variety of perspectives (see conclusion).

[8] https://support.grasshopper.codes/t/super-charge-your-coding-skills/58
[9] https://play.google.com/store/apps/details?id=com.area120.grasshopper

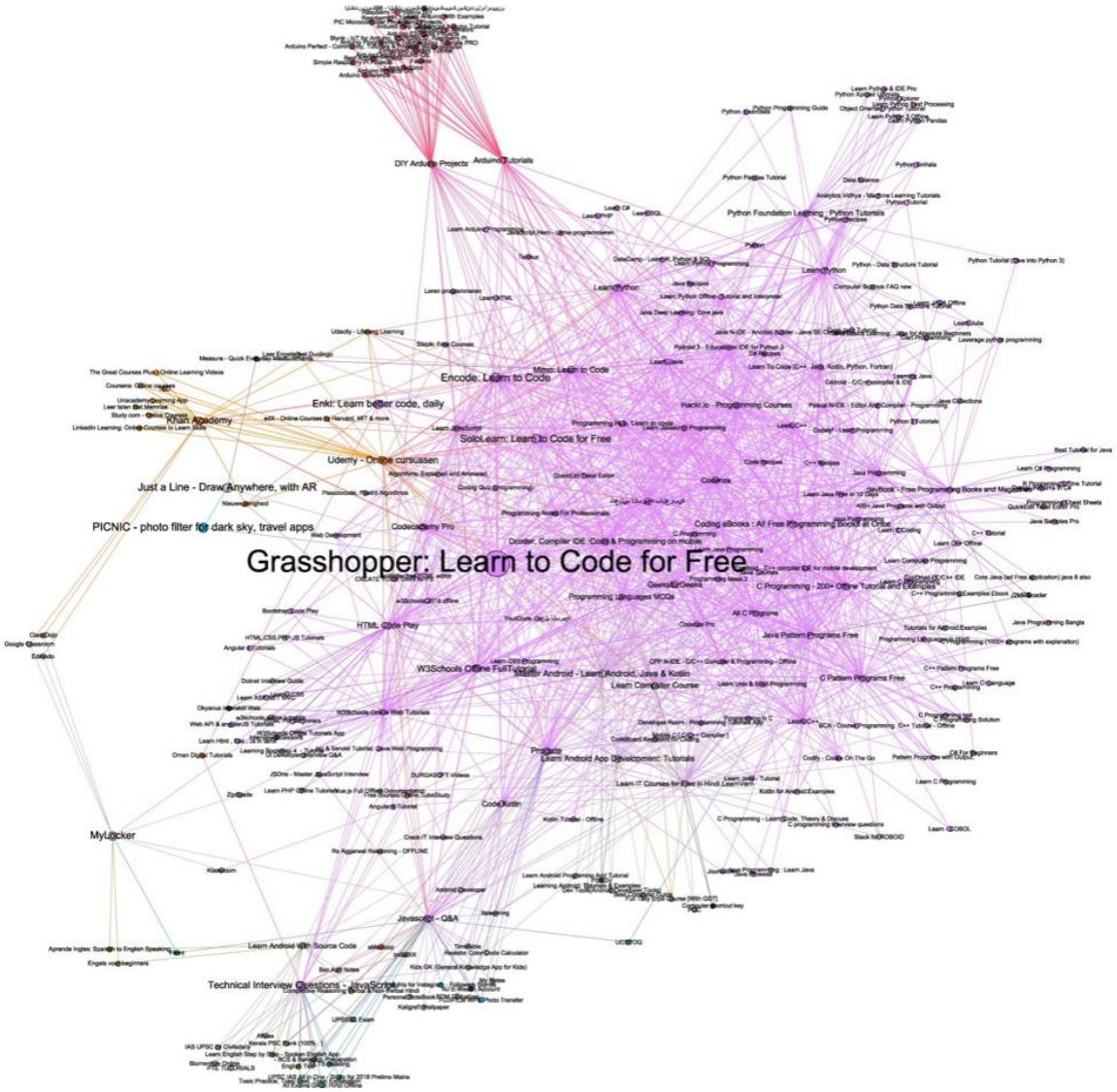

Figure 2. Similarity network.

Interface

Technologies

Grasshopper's interface broadly consists of two platforms: an instructional ('courses') and a social ('support forum') one (3a). The instructional platform is the main area intended to teach how to code in JavaScript, largely consisting of puzzles to solve, which are subsequently followed by some conceptualizing information about what the user did precisely in order to come to a solution (and occasionally complemented by a quiz or optional content). The social platform, conversely, is designed to operate as a resource when one gets stuck. Its structure mirrors the structure of the instructional platform: each puzzle has its own forum thread, where users, and the developmental team, help each other out.

Before being given access to these platforms, users first need to go through a registration procedure, which frames the user explicitly as a beginner and which further reinforces the message that Grasshopper's main finality is to let its users learn (3b). Consistent with its website, the app frames coding as a journey, which is furthermore depicted – both in text and by visuals – as agreeable and something one learns by doing and by 'fiddling around'(again shying the user away from possible perceptions that coding is abstract, technical, and hard to learn) (3c). Likewise, the app

constantly stresses that 'making mistakes is impossible', and if you do answer something incorrectly, you are assured that 'you will get it next time'.

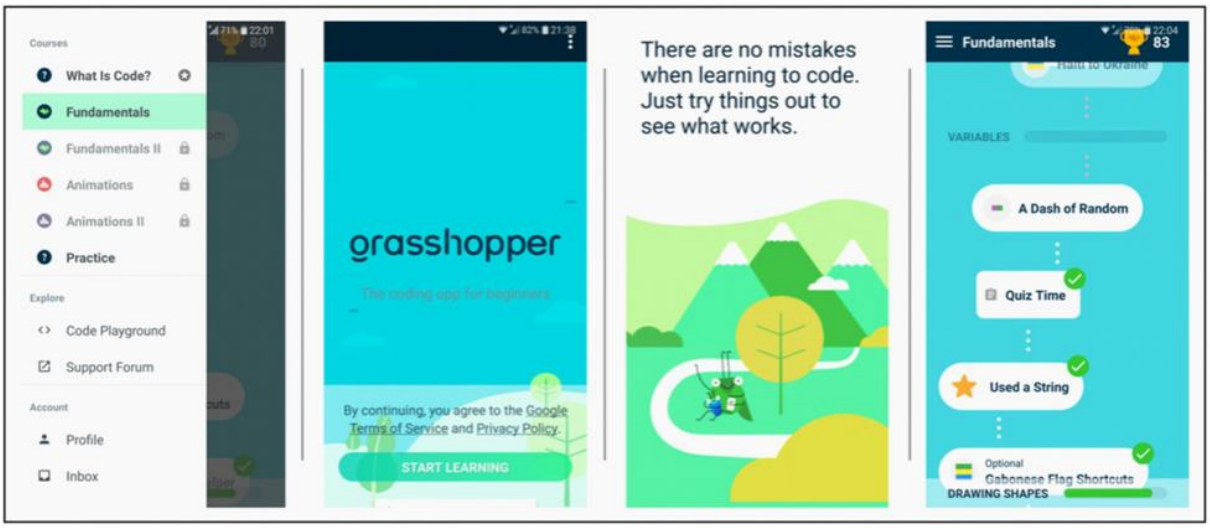

Figure 3. Grasshopper's general menu (a); welcome screen (b); prompt framing coding as a journey where one can fiddle around (c); and instructional menu (d).

As far as the environment of expected use of the instructional platform is concerned, one of the central features of Grasshopper's interface is the assumption and inscription of logical, linear progression. As Figure 3a shows, most sections of the app only unlock once one has worked through previous sections. Within each section, the learning journey of the user constitutes the core feature of the app interface, as each puzzle, quiz, or moment of explanation is placed on a vertically moving line – the higher one goes, the more progress has been made (3d). Additionally, this progression is further inscribed by means of horizontal markers (green for unfinished modules, gray for upcoming modules).

As stated, one of Grasshopper's central features are its coding puzzles. Each puzzle has its own finality, yet most of the time the general purpose of a puzzle is to make the user acquainted with new aspects or principles about coding in JavaScript, or to further elaborate upon a skill already introduced. A prototypical puzzle starts with a given set of instructions that the user needs to follow (4a) and then proceeds to an answer section that most of the time contains some preloaded code already (4b). Answers always consist of an amount of code the user has to write. However, users are not compelled – and actually not allowed – to write coding instructions from scratch. Instead, Grasshopper's interface provides a 'coding keyboard' where possible relevant pieces of code are already offered to the user. The task to fulfill, then, amounts to tap the right boxes so that the right piece of code is being compiled.

Two other major features of the course platform are connected to correctly finishing puzzles. When a puzzle has been solved, a pop-up notification informs the user what one has just produced precisely (4c). Lastly, the course platform equally contains a trophy section (4d) where one's achievements are being visualized. This section equally functions as a sort of diary, containing unlocked coding concepts and JavaScript keys.

The social platform, on the other hand, operates as a support forum if one gets stuck, but equally acts as feedback device where nuisances, perceived inconsistencies, and app issues in general can be communicated to the app developers, which are present very prominently (e.g. 5a). Next hereto, the social platform equally allows users to communicate with each other and help each other out (5b). Yet, users are not entitled to assist in any possible manner: there are rules tied to the usage of the platform in order for it to function as a 'civilized public space' (5c).

Additionally, new users are restricted in what they are entitled to do and allowed to say by limited functionalities: the more users interact on the social platform, the more badges they earn, the more functionalities are unlocked (5d). Put differently, the environment of expected use present on this platform images sociality as a civilized undertaking where users follow set rules and are forbidden to fully participate until they have gathered a prior necessary amount of experience. This imagined sociality closely resembles fairly traditional takes on socialization, where education (i.c. the app) only allows one to become a fully competent member of a certain group after a sufficient amount of disciplining (Foucault 1977). Furthermore, specifically with respect to this app environment, it is clear that app functioning is not self-evident and not operating automatically: it needs to be heavily curated, both in a technical (disabling functionalities by default) and in a social (creating behavioral rules or editing/deleting messages where necessary) sense.

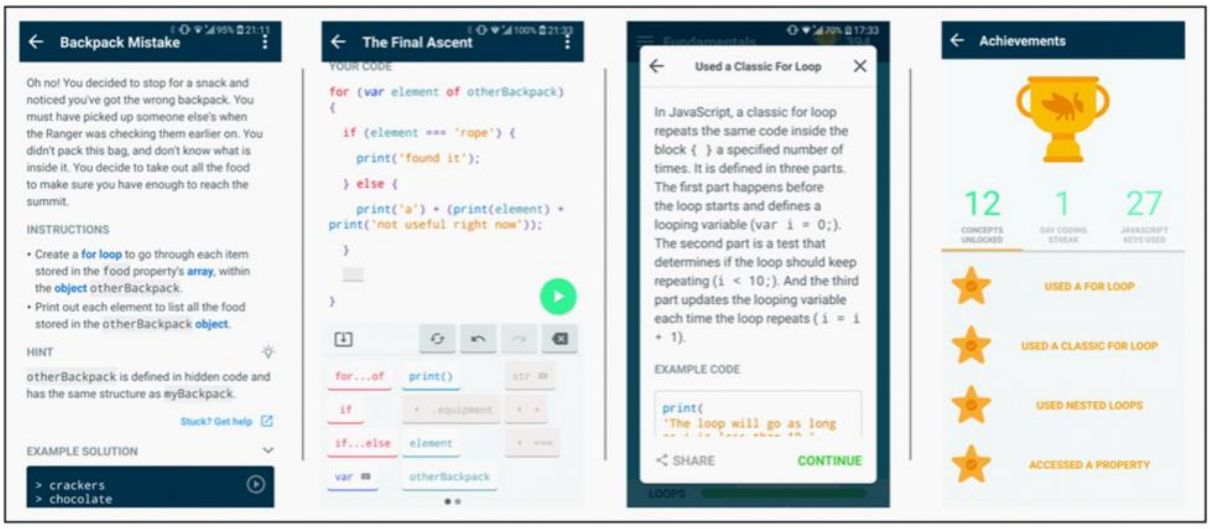

Figure 4. A puzzle's instruction section (a); answer section (b); information section (c); and achievement section (d).

On both platforms, algorithms are active in order to shape and tailor a personal learning experience. Running in the background of Grasshopper's instructional platform, algorithms allow the app to make two types of real-time decisions. These decisions are based on a continuous logging of users' performances and can either deliver appropriate feedback when the student is in the middle of solving a puzzle (6a-b), after solving a puzzle correctly (6c), or can select an appropriate next puzzle when the user has completed a current puzzle (3d). This last type of algorithmic decision making allows users who perform well to skip some puzzles and offer extra puzzles of the same difficulty level to not-proficient users (Malysheva 2017).[10] When inserting incorrect code, the concrete output of this algorithmic decision making regularly consists of boxes that instruct you how to move forward (e.g. blue boxes in 6a-b), but it can equally be a more abstract, coded-like statement that seems to come right from the computer or coding system itself (e.g. red box in 6b). On the social platforms, there are equally algorithms operable in order to assist the user who gets stuck. Interestingly, these algorithms are sometimes made explicitly visible in the form of a chatbot that can be talked to in-app, but that can equally e-mail you afterwards (6d). Thus, algorithms operate on at least three places. First, within puzzles, they log and track activity in real-time, so that they might give automated feedback on the spot. Second, between puzzles, algorithms establish which puzzle is being displayed next depending on the user's previous responses. Third, outside puzzles, the user might be given automated feedback through e-mails and through chats with bots.

[10] As a single user walking through Grasshopper, it is impossible to track down the concrete functioning of each algorithm. Rather, conform STS principles, the focus is on the *effects* of such algorithmic agency. In order to give the reader a general idea about the algorithms' functioning, the following brief description is based on Malysheva (2017), a Google employee who has provided some insight into how Grasshopper's algorithms function.

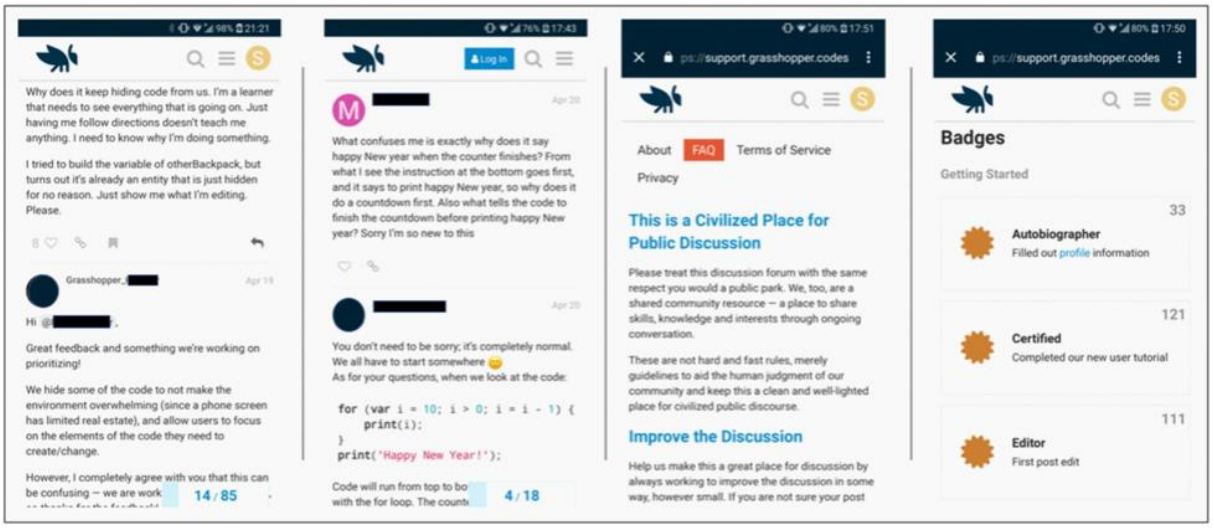

Figure 5. App developer feedback (a); User feedback (b); Social platform rules (c); Badge section (d).

These platforms and algorithms operating within Grasshopper have been described in a distinctive manner here, yet when using the app these different operations occur largely simultaneously and thus regularly mix with one another. The slowing down of app use that is characteristic of the walkthrough method, enables to identify three tensions that are inherently present within these different operations, and that might go by unnoticed in regular app use. A first tension is situated at the level of standardization versus customization. Grasshopper first of all seeks to create learners who experience an app that is customizable to their needs, in a way that is proper for them, at their own personal pace, and who can additionally rely on the continuous guidance of algorithms (6a-b; see above) as well as on the back-up of both other learners (5b) and a whole Grasshopper team (5a) when necessary. At the same time, however, learners are equally embedded in an environment with a substantial amount of standardization: the coding journey might be faster or slower paced, it still remains the same coding journey (3d). Something similar applies to Grasshopper algorithms themselves: even though these algorithms are associated with a personalization of the learner experience, their mode of operation is such that they offer predetermined standardized feedback instructions to prompt the learner (Malysheva 2017). Additionally, projected users are equally substantially standardized, in the sense that what they can do is stringently delineated and based on pre-programmed, moral principles of good (online) conduct (Light, Burgess, and Duguay 2018).

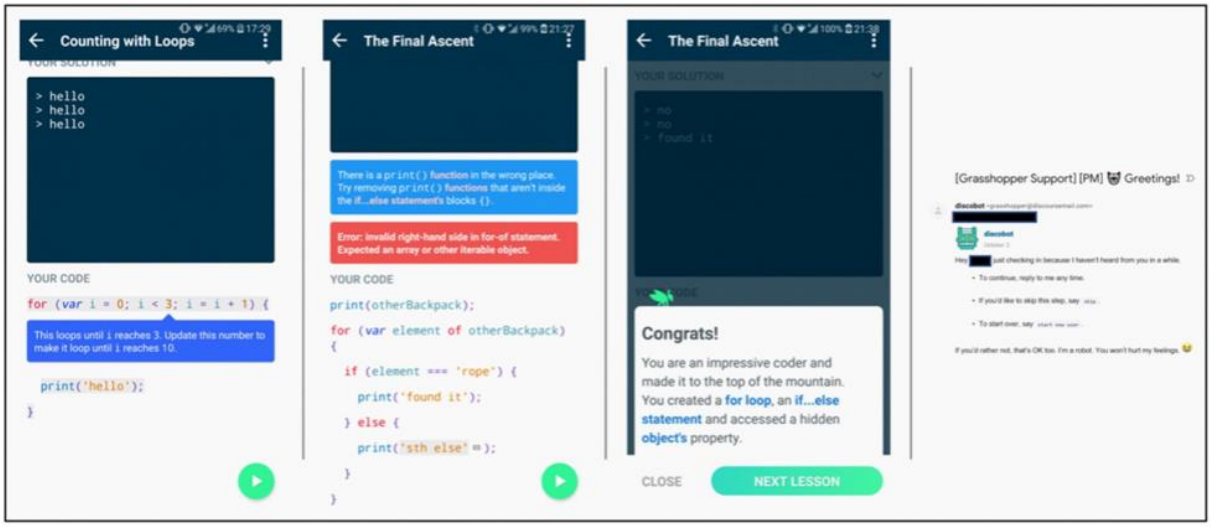

Figure 6. Algorithmic feedback within puzzle solving (a); After incorrect (b) and correct (c) puzzle solving; Personal e-mail received from chatbot.

Secondly, there is a tension between simplicity and attractivity on the one hand, and complexity and abstraction on the other. Grasshopper aims to counter the complex and technical nature of coding with an interface that induces positive affect, feel-good sentiments and a putting at ease of the user (see 'site'). This is effectuated by ensuring that coding is not about failing or succeeding, but rather about tinkering until one discovers something that works. Equally, the kind of feedback that is being given – both algorithmic and by the support team – is always highly positive and supportive (e.g. 5a;6c). Thirdly, and closely associated herewith, is a tension between personalization and responsabilization. As discussed at length already, Grasshopper makes it very clear that learning is conceived as a personal matter that might nevertheless be enhanced by tailored algorithmic pieces of feedback and human support. As a consequence, it is the user who has the final authority over what one needs, wants, and learns – or who might even decide *not* to learn. Thus, tailored personalization inevitably comes with a price: it implies that learners bear the full responsibility for their own learning. This further implies that there is no one else to attribute success or acquired knowledge (but equally failures and mistakes) to than oneself, and hence, nobody other to blame when learning does not come about (Decuypere 2019b; Simons 2018).

In sum, this section argued that technologies are no neutral backdrop operating in the background of an app and unequivocally leading to learning. Rather, platforms and algorithms *act*: they show as much as they hide, they aesthetically and discursively seduce and seek to lure the user into using the app, and they install rules of conduct inscribing what users can and are allowed to do.

Subjectivities

Grasshopper users are faced with specific expectations regarding how to behave when using the app. This was already evidenced by the civilized behavior that is expected when visiting the support platform, but the shaping of a particular type of user is by no means confined to the inscription of some community rules. In general, Grasshopper users are configured in at least four different ways.

First of all, Grasshopper users are heavily *sculpted*. That is to say, they are configured to abide to the app's expectations. Such sculpting is not necessarily as manifest as instructing specific community rules and concomitant civilized behavior, but can equally be hardly noticeable and merely 'nudge' the user in a particular way of conduct (and thinking about oneself) (Souto-Otero and Beneito-Montagut 2016). This is, for instance, the case when users tap for help and consequently receive some additional elucidation, but the solution itself remains blurred. Such blurring of code nudges the user in trying it oneself once again based on the additional instructions one has just received. Another example of such sculpting are push notifications, which one voluntarily receives each day at a particular (self-chosen) time, and which give messages like:

```
Got 5 minutes?
Your next lesson is Post Climb Postcard.
```

or

```
if (you.wantToLearnToCode)
{ tapHere(); learnJavaScript(); }
```

Whereas such notifications sculpt the user variously as a learner who can learn coding in a very short amount of daily time and as a learner who is seduced by the technocratic metaphor of language as code, what they have in common is that they prompt the user to get back to 'coding work' and hence, that they appeal to the diligence and self-understanding of the user as someone who is willing to learn. Importantly, such sculpting is not deterministic and can easily fail: users regularly involve in unruly behavior, for instance by explicitly questioning the app's instructional methods and design (5a) or by simply not complying to the installed standard of daily activity (Nissenbaum 2011).

Secondly, users are configured as subjects who are in need of (exclusively) positive feedback and reinforcements, and hence, of the message that they are constantly performing well. This *inculcating* of progress in the user works in several ways (instant feedback; achievements; badges). Each of these ways operates according to a different logic: instant feedback ('look at you go'; 'you learned so much') scripts users as persons who need to be constantly informed about their positive performances (6c). Conversely, gamified achievements enact the figure of the competitive learner who is awarded trophies (4d), whereas badges (6d) are more compliant to the idea of the coder as a navigator of a particular journey. Common to all these inculcating operations is that they script the user as someone's whose performance needs to be constantly positioned (vis-à-vis own progression, technical requirements, etc.), yet – and on top of that – always in a reinforcing way (Simons 2018).

Thirdly, Grasshopper at once *codes* and *decodes* its users. Users are *coded*: they are ostensibly configured in such a way that they are expected to act, function and think in an algorithmic manner themselves. More precisely, users are enquired to start to operate according to the lines of the *if…then* reasoning that is inherent to algorithmic functioning itself (Bucher 2018). Especially on the instructional platform, users are coded as 'automators' who need to tap required code boxes apparent on the coding keyboard (4b), or seem to be addressed directly by the algorithm itself (red box in 6c). However, users are equally constantly *decoded*, in the sense that each of their actions is logged, traced, and made amenable to analysis and algorithmic action (Herold 2016). Both operations share the technocratic contention that user behavior and technical monitoring go hand in hand and need to subsist in symbiotic fashion.

Lastly, users are continuously *framed* – that is, they are constantly being addressed in particular ways and put in specific positions. Firstly, in the process of registering users are explicitly framed as beginners who will undertake a coding journey that is planned by the app in advance. This framing strictly destines the app to be only useful for specific kinds of people (Hacking 2007). Secondly, users are configured to operate in a framework of rules (e.g. linear progression) and conduct (e.g. civilized behavior) that is not negotiable and that delimits the space of action of the user. Thirdly, on the social platform users are framed as users with a *reputation* towards other users and the Grasshopper team: the more one has been active through placing reactions, the higher one is ranked in the platform's social 'hierarchy' and the more visible one becomes (by being put up front when commenting on a message and by having distinctive badges). Such hierarchic framing of the learner stands in sharp contrast with the more general contentions of the app, which stress that everybody is able to learn to code (hence enacting a more equalizing user base).

In sum, these various ways of configuring the user exist next to the identified technological operations (and associated tensions) acting within Grasshopper-. Again, such configuring is not neutral or innocuous: configuration actively *enacts* (projects and shapes) ideal app users, to whose image actual learners are expected to conform.

Conclusion: Apps and learning regimes

Apps are increasingly important in formal and non-formal educational settings. They constitute a significant field where new methods and technologies to educate continue to emerge and where a considerable amount of experimentation on its users base exists. However, at present little research critically considers apps or explores how they are positioned within the wider ensemble of educational technologies. This article sought to address this gap by analyzing the sociomaterial practices of coding (and the pedagogies that are employed in order to do so) inscribed in, and performed by, Grasshopper. The analysis focused on the three methodological entry points of app website, store and interface (as environment of expected use), and has as such refrained from observing actual user engagement with the app itself (even though traces hereof can be retrieved from the analysis of the social platform; e.g. 5a-b). Thus, user engagement could serve as another entry point, substantiating and disconnecting actual usage from expected usage, possible modes of resistance and friction, and perhaps even equally the remodifying of the app itself (cf. Nissenbaum 2011).

To conclude, through its inquiry of selected entry points, the present article has shown, firstly, how Grasshopper is embedded within technocratic imaginaries that frame coding at once as a technical, empowering, fun, easy and addictive endeavor. In that respect, Grasshopper exemplifies the rise of a techno-solutionist way of thinking that assumes that technical skills and computational thinking will enable to solve societal and educational problems the like (Williamson 2016a; Morozov 2014). Furthermore, the app is at once staged as a tool for learning and as a tool that escapes the alleged slowness, rigidity and formality of the traditional educational field. As an innovative learning tool, Grasshopper is positioned at the nexus of two increasingly interconnecting domains: the (foremost technical) field of what *can* be done, and the (foremost educational) field of what *should* be done with new technologies in the educational field. In practice, however, the connecting of these two domains often implies that the educational concern is being subsumed under the technical one: what should be done, is then equivocated with what is technologically possible (Salomon 2016). In as far as this article has shown that learning to code is primarily effectuated through chopping down educational activities in segmented series of minor instructions, and coupled to a fairly simplistic model of behavior reinforcement (achieved by real-time, almost exclusively positive feedback, algorithmic decision making and continuous tracking), it has explored the contours of a technical learning regime that strives to be entirely self-contained (Watters 2017; Chang 2019). Closely related hereto is the observation that Grasshopper's business model is anything but clear. As discussed, users are enticed to subscribe to an affiliated and revenue sharing proprietary platform after having finished Grasshopper. However, one could additionally surmise that the main currency users are paying with are the data they provide to Google about how one learns (about coding). At the very least, the Grasshopper user base is offering Google a tremendously fine-grained view on how learners go about when they learn a technical subject as coding. As such, not unlike other educational apps, one might conjecture that Grasshopper potentially operates as exploiter of the learning patterns and behaviors of its users (Fuchs, 2012; Lindh and Nolin, 2016).

Secondly, the article made clear that apps shouldn't be approached as instruments that merely instill proclaimed learning content in their users. Rather than that, the article substantiated the various ways in which proclaimed user subjectivities are being inscribed and scripted in the app's environment of expected use. Operating as an active device that is nested in various ecologies, Grasshopper should be understood as a technology that is multiple, in the sense that it instills other forms of learning into its learners as well. For instance, digital sociality is being explicitly and implicitly inscribed in Grasshopper's code. The same could be said about the rewarding and the (de)coding of learners, which all seek to bring about pre-specified desired effects. Importantly, this not only applies to user learning, but equally to platform and algorithmic learning. Just as user learning is always at once technical learning, platform and algorithmic learning is social through and through: curated by individuals, commented upon by users, reprogrammed where necessary by developers, and so on. At present, little is understood about the sociality that is inscribed *within* the algorithmic and within platforms more broadly (Seaver 2018). Yet, the constant evolutionary nature – short update cycles, iterative development, direct user feedback – of apps makes them ideal to analyze such socio-technical agency (Dieter et al. 2018). Machine learning and algorithmic adaptation produce different sorts of users as much as they are themselves the product of user feedback, baseline visions about the social world and education held by development teams and their presuppositions of what learning is and entails (Beer 2017; Souto-Otero and Beneito-Montagut 2016). Thus, research on educational technologies needs to take into account both the *technicality of the social* and *the sociality of the technical* – and preferably how both are constantly co-producing each other.

References

Albury, Kath, et al. 2017. "Data Cultures of Mobile Dating and Hook-up Apps: Emerging Issues for Critical Social Science Research." *Big Data & Society* 4 (2): 205395171772095. doi:10.1177/2053951717720950.

Bastian, Mathieu, Sebastien Heymann, and Mathieu Jacomy. 2009. "Gephi: An Open Source Software for Exploring and Manipulating Networks." *Third International AAAI Conference on Weblogs and Social Media*, 361–62. doi:10.1136/qshc.2004.010033.

Beer, David. 2017. "The Social Power of Algorithms." *Information, Communication & Society* 20 (1): 1–13. doi:10.1080/1369118X.2016.1216147.

Bucher, Taina. 2018. *If...Then: Algorithmic Power and Politics*. Oxford University Press.

Callaghan, Melissa N, and Stephanie M Reich. 2018. "Are Educational Preschool Apps Designed to Teach ? An Analysis of the App Market." *Learning, Media and Technology* 43(3): 280–93. doi:10.1080/17439884.2018.1498355.

Callon, Michel. 1986. "Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay." *In J. Law, Power, Action and Belief: A New Sociology of Knowledge?*: 196–223.

Callon, Michel, and Fabian Muniesa. 2005. "Economic Markets as Calculative Collective Devices." *Organization Studies* 26 (8): 1229–50. doi:10.1177/0170840605056393.

Cherner, T, J Dix, and C Lee. 2014. "Cleaning up That Mess: A Framework for Classifying Educational Apps." *Contemporary Issues in Technology and Teacher Education* 14 (2): 158–93.

Decuypere, Mathias. 2016. "Diagrams of Europeanization: European education governance in the digital age." *Journal of Education Policy* 31 (6): 851-872.

Decuypere, Mathias. 2019a. "STS in/as education: Where do we stand and what is there (still) to gain? Some outlines for a future research agenda." *Discourse: Studies in the Cultural Politics of Education* 40 (1): 136-145.doi: 10.1080/01596306.2018.1549709

Decuypere, Mathias. 2019b. "Open education platforms: Theoretical ideas, digital operations and the figure of the open learner." *European Education Research Journal* 18 (4): 439-460. doi:10.1177/1474904118814141

Decuypere, Mathias. 2019c. "Visual network analysis: A qualitative method for researching sociomaterial practice." *Qualitative Research*. doi:10.1177/1468794118816613.

Decuypere, Mathias, and Maarten Simons. 2016. "Relational thinking in education: Topology, sociomaterial studies and figures". *Pedagogy Culture & Society* 24 (3): 371-386.

Dieter, Michael, et al. 2018. "Store, Interface, Package, Connection : Methods and Propositions for Multi-Situated App Studies." CRC Media of Cooperation Working Paper Series.

Edwards, Richard, and Patrick Carmichael. 2012. "Secret Codes: The Hidden Curriculum of Semantic Web Technologies." *Discourse: Studies in the Cultural Politics of Education* 33 (4): 575–90. doi:10.1080/01596306.2012.692963.

Eynon, Rebecca. 2018. "Into the Mainstream : Where next for Critical Ed Tech Research ?" *Learning, Media and Technology* 43 (3): 217–18. doi:10.1080/17439884.2018.1506976.

Foucault, M. 1977. *Discipline & Punish: The Birth of the Prison*. doi:10.2307/2065008.

Fuchs, C. 2012. "Google Capitalism, TripleC: Communication, Capitalism & Critique." Open Access Journal for a Global Sustainable Information Society 10 (1): 42–48.

Fuller, Steve. 2006. The Philosophy of Science and Technology Studies. New York: Routledge.

Gillespie, Tarleton. 2010. "The Politics of 'Platforms.'" *New Media and Society* 12 (3): 347–64.

doi:10.1177/1461444809342738.

Gillespie, Tarleton. 2015. “Platforms Intervene.” *Social Media and Society*, no. April-June: 1–2. doi:10.1177/2056305115580479.

Godwin-Jones, R. 2011. “Emerging Technologies: Mobile Apps for Language Learning.” *Language Learning & Technology* 15 (2): 2–11.

Herold, Benjamin. 2016. “Google Acknowledges Data Mining Student Users Outside Apps for Education.” https://blogs.edweek.org/edweek/DigitalEducation/2016/02/google_acknowledges_data_mining_GAFE_users.html.

Hirsh-Pasek, Kathy, Jennifer M. Zosh, Roberta Michnick Golinkoff, James H. Gray, Michael B. Robb, and Jordy Kaufman. 2015. “Putting Education in ‘Educational’ Apps: Lessons From the Science of Learning.” *Psychological Science in the Public Interest* 16 (1): 3–34. doi:10.1177/1529100615569721.

Hörl, Erich. 2017. “Introduction to General Ecology: The Ecologization of Thinking.” In General Ecology: The New Ecological Paradigm, edited by Erich Hörl and James Burton, 1–74. London: Bloomsbury Academic.

Introna, Lucas. 2016. “Algorithms , Governance , and Governmentality : On Governing Academic Writing.” *Science, Technology & Human Values* 41 (1): 17–49. doi:10.1177/0162243915587360.

Kitchin, Rob, and Martin Dodge. 2011. *Code/Space: Software and Everyday Life*. Cambridge, MA and London: MIT Press.

Latour, Bruno. 2005. *Reassembling the Social: An Introduction to Actor-Network-Theory*. Oxford University Press.

Law, John. 2009. “Actor Network Theory and Material Semiotics.” In *The New Blackwell Companion to Social Theory*, edited by Bryan Turner, 141–58. Chichester: Wiley-Blackwell.

Light, Ben, Jean Burgess, and Stefanie Duguay. 2018. “The Walkthrough Method: An Approach to the Study of Apps.” *New Media and Society* 20 (3): 881–900. doi:10.1177/1461444816675438.

Lindh, Maria, and Jan Nolin. 2016. “Information We Collect: Surveillance and Privacy in the Implementation of Google Apps for Education.” *European Educational Research Journal* 15 (6): 644–63. doi:10.1177/1474904116654917.

Mackenzie, Adrian. 2015. “The Production of Prediction: What Does Machine Learning Want?” *European Journal of Cultural Studies* 18 (4–5): 429–45. doi:10.1177/1367549415577384.

Malysheva, Yana. 2017. “Grasshopper ’s Event System: Defining and Readcting to Noteworthy Features of Student Code.” In *IEEE Blocks and beyond Workshop*, 111–12.

Nissenbaum, Helen. 2009. Privacy in Context: Technology, Policy, and the Integrity of Social Life. Stanford: Stanford University Press.

Salomon, Gavriel. 2016. “It’s Not Just the Tool but the Educational Rationale That Counts.” In *Educational Technology and Polycontextual Bridging*, edited by E Elstad, 149–61. Sense.

Seaver, Nick. 2017. "Algorithms as Culture: Some Tactics for the Ethnography of Algorithmic Systems." Big Data & Society, July-December: 1–12.

Simons, Maarten. 2018. “Refiguring the European Student: Mixed Transnational Feelings.” In *Critical Analyses of Educational Reforms in an Era of Transnational Governance*, edited by E. Hultqvist, S. Lindblad, and Thomas Popkewitz, 169–84. Springer.

Souto-Otero, Manuel, and Roser Beneito-Montagut. 2016. “From Governing through Data to Governmentality through Data: Artefacts, Strategies and the Digital Turn.” *European Educational Research Journal* 15 (1): 14–33. doi:10.1177/1474904115617768.

Star, Susan. 2010. “This Is Not a Boundary Object : Reflections on the Origin of a Concept.” *Science, Technology &*

Human Values 35 (5): 601–17. doi:10.1177/0162243910377624.

Wade Morris, Jeremy, and Evan Elkins. 2015. "FCJ-181 There's a History for That: Apps and Mundane Software as Commodity." *The Fibreculture Journal*, no. 25. *The Fibreculture Journal*: 63–88. doi:10.15307/fcj.25.181.2015.

Watters, Audrey. 2017. "Dunce's App. How Silicon Valley's Brand of Behaviorism Has Entered the Classroom." *The Baffler*. https://thebaffler.com/latest/behaviorism-education-watters.

Williamson, Ben. 2016a. "Political Computational Thinking: Policy Networks, Digital Governance and 'Learning to Code.'" *Critical Policy Studies* 10 (1). Routledge: 39–58. doi:10.1080/19460171.2015.1052003.

———. 2016b. "Silicon Startup Schools: Technocracy, Algorithmic Imaginaries and Venture Philanthropy in Corporate Education Reform." *Critical Studies in Education* 59 (2): 218–36. doi:10.1080/17508487.2016.1186710.

———. 2017. "Decoding ClassDojo: Psycho-Policy, Social-Emotional Learning and Persuasive Educational Technologies." *Learning, Media and Technology* 42 (4): 440–53. doi:10.1080/17439884.2017.1278020.

Woolgar, Steve. 1990. "Configuring the User: The Case of Usability Trials." The Sociological Review 38 (1): 58–99.

Woolgar, Steve, and Javier Lezaun. 2013. "The Wrong Bin Bag: A Turn to Ontology in Science and Technology Studies?" *Social Studies of Science* 43 (3): 321–40. doi:10.1177/0306312713488820.

Ziewitz, Malte. 2016. "Governing Algorithms: Myth, Mess, and Methods." Science, Technology, & Human Values 41 (1): 3–16. doi:10.1177/0162243915608948.

Zosh, Jennifer, Sarah Lytle, Roberta Golinkoff, and Kathy Hirsh-Pasek. 2017. "Putting the Education Back in Educational Apps: How Content and Context Interact to Promote Learning." In *Media Exposure during Infancy and Early Childhood*, edited by R. Barr and DN. Linebarger, 259–82. Springer. doi:10.1561/2200000016.