Software and Systems Modeling
https://doi.org/10.1007/s10270-021-00897-8

REGULAR PAPER

Modelling on mobile devices

A systematic mapping study

Léa Brunschwig[1] · Esther Guerra[1] · Juan de Lara[1]

Received: 22 January 2021 / Revised: 22 May 2021 / Accepted: 26 May 2021

Abstract
Modelling is central to many disciplines in engineering and the natural and social sciences. A wide variety of modelling languages and tools have been proposed along the years, traditionally for static environments such as desktops and laptops. However, the availability of increasingly powerful mobile devices makes it possible to profit from their embedded sensors and components (e.g. camera, microphone, GPS, accelerometer, gyroscope) for modelling. This has promoted a new range of modelling tools specially designed for their use in mobility. Such tools open the door to modelling in dynamic scenarios that go beyond the capabilities of traditional desktop tools. For example, modelling in mobility can be useful to design smart factories on-site, or to create models of hiking routes while walking along the routes, among many other scenarios. In this paper, we report on a systematic mapping study to identify the state of the art and trends in modelling on mobile devices. The study covers both research papers and modelling apps from the Android and iOS stores. From this analysis, we derive a classification for mobile modelling tools along three orthogonal dimensions, discuss current gaps, and propose avenues for further research.

Keywords Model-driven engineering · Modelling tools · Mobile devices · Systematic mapping study

1 Introduction

Modelling captures the essence of a system for a given purpose, like simulation, understanding, discussion, analysis or testing [100]. For this reason, models are used in many disciplines. For example, in engineering, models serve as design and blueprints of the artefacts to be built [7], while in the natural sciences, models are used to describe and analyse existing phenomena—like the evolution of animal populations or the orbit of planets—for purposes like prediction or understanding [74].

Software engineering is no exception to the use of models, as they are used for both project management (e.g. planning) and technical tasks (e.g. requirements and design) [41,58]. Actually, in some paradigms like the model-driven engineering (MDE), models are the main development artefacts [87] and serve not only as documentation but also to describe, simulate, verify, validate and generate code for the application under development, among other activities [41]. By focussing on the essential aspects of the system under construction, engineers deal with less accidental details (i.e. those that are unnecessary for the task at hand [18]), and therefore, productivity and quality can get improved [49].

In the context of software engineering, modelling can be conducted using either general-purpose or domain-specific languages (DSLs) [49,103]. The former are languages—like the UML [98]—that permit describing any kind of application, while the latter target a narrow domain, like tax calculation, logistics planning, or conversational adventure games [49]. DSLs are sometimes oriented to domain experts, who may lack a technical background, to empower them to accomplish small development tasks [11].

Communicated by Steffen Zschaler.

✉ Léa Brunschwig
lea.brunschwig@uam.es

Esther Guerra
esther.guerra@uam.es

Juan de Lara
juan.delara@uam.es

[1] Computer Science Department, Universidad Autónoma de Madrid, Madrid, Spain

Published online: 29 June 2021

Traditionally, computer-assisted modelling has taken place in a static setting, using tools that run on desktop computers or laptops. However, modelling in mobility is becoming an attractive possibility since a large percentage of the world population owns a smartphone,[1] and the capabilities of mobile devices are rapidly increasing in terms of both computing power and variety of embedded sensors and components (e.g. camera, GPS, microphone, accelerometer, gyroscope, lidar). Moreover, some researches have shown that modelling can benefit from mobility and context [102], e.g. for designing domotic buildings on-site [88]; for smart city planning, where the position of sensors within a city must be precisely determined; or in the area of tourism, where tourist guides create touristic routes while walking through the city, and tourists can download the route models and rate their spots in-place while visiting the city [20]. Thus, some approaches have recently emerged to facilitate domain-specific modelling using the capabilities of mobile devices. CEL [53], DSL-comet [102], FlexiSketch [106], HoloFlow [90], Metaphore [88] and NetSketcher [10] are some of the tools that tackle this challenge.

The purpose of this paper is to provide an overview of the state of the art regarding mobile modelling tools. To this aim, we have performed a systematic review of the literature in academic publication venues, and have analysed existing tools found in digital app distribution platforms, like App Store[2] for iOS, or Google Play[3] for Android. Based on this review, we classify the existing works and tools along three main orthogonal dimensions, identify gaps, and propose research opportunities.

The rest of this paper is organized as follows. First, Sect. 2 introduces related works. Then, Sect. 3 describes the methodology of our study and states research questions. Next, Sect. 4 categorizes the identified relevant works according to the features a mobile modelling tool may have. Section 5 answers the research questions and discusses trends, gaps and research directions. Finally, Sect. 6 summarizes the main findings and concludes the paper.

2 Related works

To the best of our knowledge, there is no previous systematic mapping study on approaches to modelling using mobile devices. Hence, this section positions our work with respect to systematic reviews and surveys on other topics related to both mobiles and modelling.

MDE has been widely used to automate the development of mobile applications, and several surveys on the proposed approaches have been published [33,83]. However, our focus is not on the use of MDE for mobile app development, but on using mobile devices for modelling. Similarly, there are mapping studies on different aspects of mobile development (like testing [111]) and types of mobile apps (like those based on machine learning [82]). However, our interest is on mobile apps supporting modelling. To the best of our knowledge, ours is the first mapping study on this topic.

There is a recent trend for making modelling environments available through the web. Tools like AToMPM [24] or WebGME [63] support modelling via web browsers, and the so-called *low-code* development tools allow the creation of software applications using visual diagrams and forms via cloud-based environments [86]. Certainly, it is possible to use a web browser from a mobile device, but our focus is on native mobile apps that profit from the distinguishing features of mobile devices for modelling.

Once modelling is done in mobility, the smart devices can exploit the data provided by their sensors to enrich the modelling experience (e.g. to adapt the model depending on the user location). There are surveys on modelling context awareness [15] and context-aware systems [9]. However, context adaptation and context awareness are just one aspect of mobile modelling. Moreover, we are only interested in adaptation if it occurs in the context of mobile apps for modelling.

Beyond model construction, mobile modelling may use models at run-time [16]. For example, a touristic app can offer a DSL to create and use models of touristic itineraries. There are surveys on models at run-time, like [13]. However, we only want to look at approaches using models at run-time that also support mobile modelling.

Collaboration may also be important in mobile modelling [101]. There are surveys on collaborative modelling [32] proposing classification criteria. Again, our focus is on collaboration in conjunction with mobility, and so we discard works on collaborative modelling that do not occur on mobile devices.

Domain-specific modelling and DSLs have been used for end-user development (see e.g. the mapping study in [11]). As in the previous cases, our study shall only include end-user development methods if they target modelling activities within mobile devices.

Finally, note that the use of touch-enabled devices does not mandatorily imply the mobility aspect. For example, [28] and [76] propose systems based on tabletops that support collaboration to model simultaneously on the same screen. We exclude this kind of works from our study, as we only

[1] According to https://www.bankmycell.com/blog/how-many-phones-are-in-the-world 45,04% of the population is estimated to own a smartphone in 2020 (around 3.500.000.000 of smartphone users).

[2] https://www.apple.com/app-store/.

[3] https://play.google.com.

consider touch-enabled modelling tools for mobile devices (tablets or smartphones).

3 Research method

The goal of this study is to understand the current state of modelling using mobile devices, the rationale of the approaches, and the features of these tools. Specifically, we pose the following four research questions:

> **RQ1:** *What are the domains where mobile modelling has been applied?*
> **RQ2:** *What are the motivations argued for mobile modelling?*
> **RQ3:** *What are the characteristic features of existing mobile modelling approaches?*
> **RQ4:** *What are the gaps in current approaches, and the research opportunities in mobile modelling?*

To answer these questions, we have conducted a systematic mapping study (SMS) [79] in the published literature to identify and classify the relevant works. Moreover, we have enriched the obtained results with the analysis of existing tools in app distribution platforms like Apple's App Store (for iOS devices) or Google Play (for Android devices).

In the following, we describe how we identified the relevant papers and apps. First, Sect. 3.1 presents the used search query. Then, Sect. 3.2 details the databases sought and the search protocol. Finally, Sect. 3.3 reports on the paper and app selection procedure.

Table 1 Keywords used for the first search string

Mobile development		MDE
Mobile	**AND**	Model-driven engineering
OR mobile development		**OR** domain-specific language
OR mobile programming		**OR** domain-specific modeling language
OR mobile app*		**OR** domain-specific modeling language
OR android		**OR** MDE
OR iOS		**OR** DSL
		OR DSML

Table 2 Keywords used for the second search string

Mobile development		MDE
Mobile device*	**AND**	Model-driven engineering
OR tablet*		**OR** domain-specific language
OR smartphone*		**OR** domain-specific modeling language
OR mobile development		**OR** domain-specific modeling language
OR mobile programming		
OR mobile app*		
OR android		

3.1 Search string selection

According to the SMS method [79], the first step is to identify search strings and relevant keywords for our topic. Since our research involves MDE and mobile development, we decided to build a search string using these two lexical fields, obtaining the query of Table 1.

The query on Scopus and Web of Science (WoS) retrieved 11.032 and 6.158 results, respectively. These large numbers can be explained because the acronyms *MDE*, *DSL* and *DSML* are frequently used in topics unrelated to our SMS, and the term *mobile* is too generic.

Hence, we refined the search query as shown in Table 2. Specifically, we removed the acronyms and the terms that returned many false positives, and replaced *mobile* by *mobile device**, *tablet** and *smartphone**. The keyword *iOS* was removed from the query because it returned many papers from the IOS press editorial unrelated to our topic. This new query returned 205 papers on Scopus and 263 on WoS. Despite these numbers are more reasonable, while screening the retrieved papers, we realised that many were either out of scope or related to code generation for mobile apps and not for modelling on mobile devices.

After analysing the abstract of some relevant papers, we decided to specialize the MDE lexical field to focus on modelling editors and hence narrow the search. Regarding the lexical field of mobile development, we deleted the term *mobile programming* because it introduced too much noise, and added the term *social network** to catch modelling approaches based on conversation within social networks [78] (cf. Fig. 9) since social networks are heavily used as native tools within mobile devices. Table 3 shows the final search string selected for the SMS.

In addition, to select the relevant non-academic tools available in the app stores, we had to define an adequate protocol. Since these stores have no advanced search mechanisms, we looked for recommended tools similar to relevant tools we had already identified, like DrawExpress [29] and Lucidchart [57], and we also used search keywords like *diagram*, *UML*, *model*, *modelling*, *DSL* and *domain-specific model*.

Table 3 Keywords used for the final search string

Mobile development		MDE
Mobile device*	**AND**	Visual environment*
OR tablet*		**OR** graphical editor*
OR smartphone*		**OR** graphical environment*
OR mobile development		**OR** visual method*
OR mobile app*		**OR** visual editor*
OR android		**OR** model* tool*
OR social network*		**OR** model* editor*
		OR graphical model* language*
		OR gesture-based model*
		OR software design notation*

3.2 Databases and search protocol

Smart mobile devices are now part of our daily life, but they are a new technology within computer science history. Thus, it is logical not to consider papers published prior to 2000. Our temporal landmark is the iPhone first generation, launched in 2007, which is the first contemporary smartphone. To make sure not to miss relevant papers, we set our search range from January 2005 to November 2020 (the date we performed the query).

We applied the selected search string on Scopus, Web of Science, the ACM Digital Library, IEEE Xplore and SpringerLink. Each database has its own query syntax and search fields. Table 4 shows the search fields that we used in each case. In all databases but SpringerLink, we sought for articles whose title, abstract or keywords contain at least one term related to mobile development, and at least one term related to MDE, as specified in Table 3. SpringerLink does not allow choosing the fields the query is applied on, hence we looked for the terms in the entire document.

After the paper retrieval, we merged the results and removed duplicates. The inclusion criteria to select the pertinent publications were the following:

- The paper is written in English;
- The approach is related to modelling;
- The approach is available on, or designed for, mobile devices.

Conversely, we excluded the papers that did not meet any of the inclusion criteria, that propose an MDE approach to generate mobile apps from a desktop tool, or that present drawing (but not modelling) tools.

In addition to papers, we sought for existing tools (apps) within distribution stores to obtain a panorama of the current practice. We did our search in December 2020 on App Store and Google Play. We did not consider the Huawei AppGallery[4] and the Microsoft Store[5] because we did not have the devices to run the tools. We selected the tools according to the following inclusion criteria:

- The tool is rated 3 stars or above;
- The tool has been downloaded at least 50.000 times (only relevant on Google Play);
- The tool supports modelling;
- The tool has a free version.

The first two criteria ensure a minimum quality and usage for the apps. To capture further relevant tools, we also considered other apps recommended by the store ("You might also like" in App Store and "Similar apps" in Google Play).

Table 4 Search fields in the considered databases

Database	Search fields
Scopus	Title OR Abstract OR Keywords
Web of science	Title OR Abstract OR Keywords
ACM digital library	Title OR Abstract OR Keywords
IEEE Xplore	Title OR Abstract OR Keywords
SpringerLink	Entire document

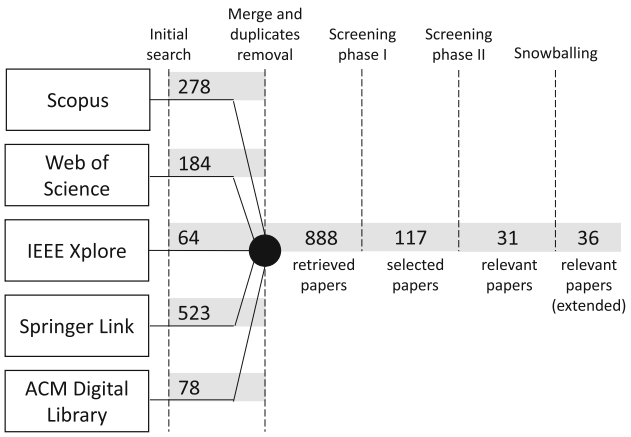


Fig. 1 Paper selection process

3.3 Paper and tool selection

Figure 1 shows the followed paper selection process. First, the initial search on the five databases retrieved 1.127 papers. After removing duplicates, 888 unique papers remained. Then, we conducted two screening phases. In the first phase, each one of the three authors of this paper (one PhD student, two professors) analysed the title and abstract of the papers to

[4] https://appgallery.huawei.com/.

[5] https://www.microsoft.com/en-us/store/apps/windows-phone.

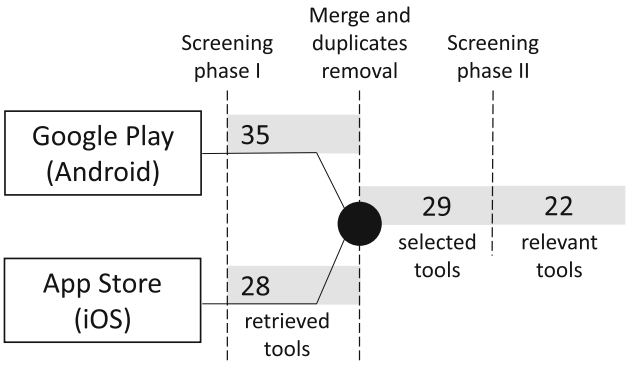


Fig. 2 Process of tool selection

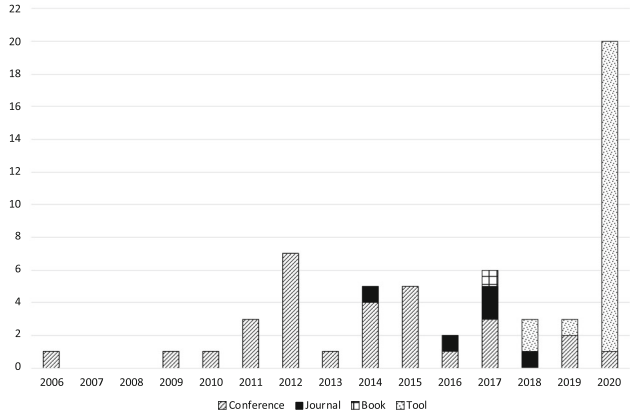


Fig. 3 Temporal and venue distributions of the relevant papers and tools

classify them as irrelevant or (potentially) relevant. A paper was selected if at least one author marked it as relevant. This resulted in the selection of around 13% of the retrieved papers. The resulting set is named "selected papers" in Fig. 1. In a second screening phase, each selected paper was read carefully to assess its relevance. Around 26% of the papers were deemed relevant. We call this set of filtered papers "relevant papers". Finally, we performed a snowballing process [104], looking in the list of references of the relevant papers others that might fit in our study. This process resulted in 5 additional papers. Altogether, the 36 final papers account for 25 different approaches. In the study, when there are several papers covering the same approach, we take all of them into account but cite just the most recent one.

With respect to tools, we followed the process depicted in Fig. 2. First, we applied the search protocol on the two app stores. In this case, the figure does not show the number of retrieved apps, since the stores do not provide this information. Then, we conducted two screening phases. In the first one, we read the tool description to discard any tool clearly out of the scope of our study. After this first screening, we obtained 63 tools in total. Several tools were available for both iOS and Android, so after removing duplicates, we ended up with a set of 29 unique "selected tools". In the second phase, we installed and tested each app, obtaining a filtered set of 22 "relevant tools".

Figure 3 breaks down the number of relevant papers by type of venue and year, and the number of relevant tools by the year of their last release. The graphic shows that most papers were published at conferences, which might be indicative of being an area in evolution, not mature yet. Regarding tools, most of them are actively maintained and released their last version during the year 2020. Only a few tools have their last version from 2018. This shows that there is interest and practical value on apps for modelling in mobile devices.

4 Analysis and classification of approaches to modelling on mobile devices

Building on [102], we classify the approaches to modelling on mobile devices along three main dimensions, summarized in the feature diagram [47] of Fig. 4:

- *Modelling language* deals with aspects related to the language(s) supported by the approach, like the style of its syntax, its semantics, or the support for fine-grained access control to language elements depending on the user.
- *Language definition* classifies the approaches depending on whether they support a fixed set of languages (e.g. UML, BPMN), or if they allow defining new languages for domain-specific modelling.
- *Tooling* comprises features of the modelling tool itself, such as its deployment architecture, the collaboration support, the interoperability with other tools, and the way users can interact with the tool for modelling.

In the next subsections, we describe and refine these dimensions, and classify the relevant works along them.

4.1 Modelling language

In this section, we classify the approaches based on the modelling languages they support. This includes the language concrete syntax (Sect. 4.1.1), the extended modelling capabilities to capture informal model information (Sect. 4.1.2), the language semantics (Sect. 4.1.3), and the ability to define fine-grained access control policies for different user types (Sect. 4.1.4).

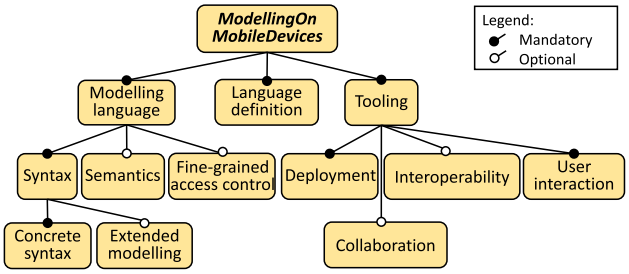


Fig. 4 Dimensions of modelling tools for mobile devices

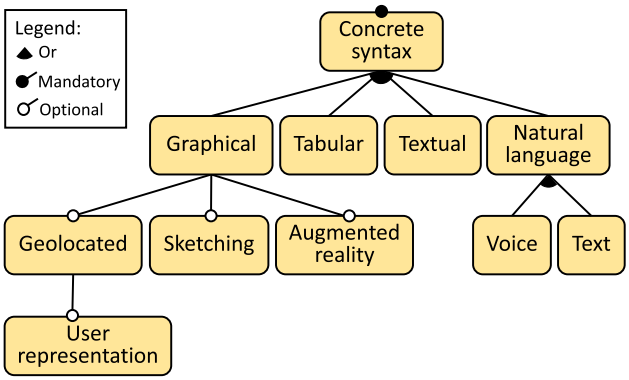


Fig. 5 Feature model for concrete syntax of modelling languages on mobiles

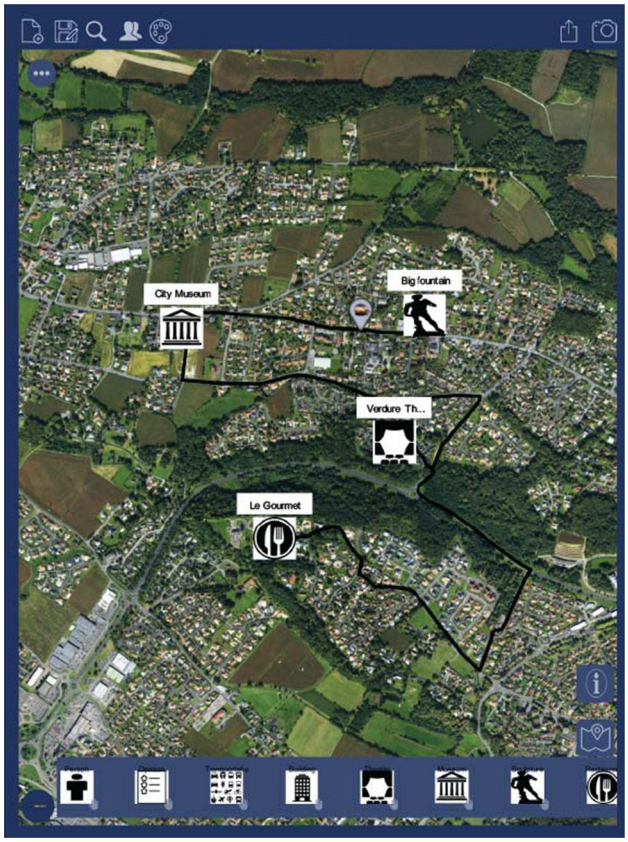


Fig. 6 Screenshot of DSL-comet illustrating a geolocated graphical concrete syntax [20]

4.1.1 Concrete syntax

Modelling languages comprise an *abstract syntax*, which defines the language primitives, properties, relationships and constraints regarding the use of the syntax [17]. Besides, languages have a *concrete syntax*, which describes how the models of the language are to be visualized. Traditionally, in desktop computers, the concrete syntax of most modelling languages is either graphical (e.g. UML sequence diagrams) or textual (e.g. Kermeta [45]). In the case of modelling on mobile devices, this syntax has to be chosen carefully since the particularities of the device may influence the user interaction and experience. Figure 5 shows a feature diagram with a classification of concrete syntaxes tailored to mobile devices.

In mobile devices, a graphical concrete syntax may be *geolocated*. In such a case, the model elements are displayed on a map, and their position coordinates are reified as graphical attributes. This syntax can eventually represent the user's position on the map (feature *user representation*). Figure 6 illustrates a geolocated DSL for modelling touristic routes atop the DSL-comet tool [20,102]. Objects of the DSL are geopositioned, and links follow the map roads and streets. The user is represented within the model as a pin (close to the object named *Big fountain*).

A graphical concrete syntax may support *sketching* to mimic the pen-and-paper feel of traditional modelling via a touch-screen. As an illustration, Fig. 7 shows a screenshot of the FlexiSketch tool [106] during a modelling session in the home networking domain.

Augmented reality (AR) [8] permits superimposing a computer-generated image on a view of the real world. AR in mobile devices is possible using their camera to superimpose virtual objects, which can be interacted via the device touch screen using technologies such as ARKit [5] and ARCore [4]. Another option is the use of head-mounted devices (glasses) like HoloLens [40] or Magic Leap [60], where the interaction occurs via gestures. Modelling languages with an AR graphical syntax provide an immersive model representation, as they can be used to model close to the system under study by overlaying the model elements onto real-world elements. Figure 8 shows a screenshot of HoloFlow [90], a modelling tool to configure IoT devices and workflows by the use of AR atop HoloLens.

Tabular syntaxes represent the model elements in a matrix or in menus. While this kind of concrete syntax is not specific to mobile devices, it can be useful to maximize the space in the reduced size of mobile screens.

Finally, *natural language* concrete syntaxes enable modelling via written *text* or *voice*. As an example, Fig. 9 shows

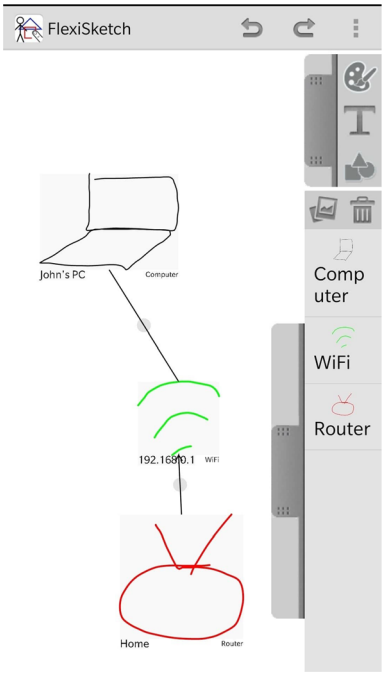


Fig. 7 Screenshot of FlexiSketch illustrating a sketch-based concrete syntax [106]

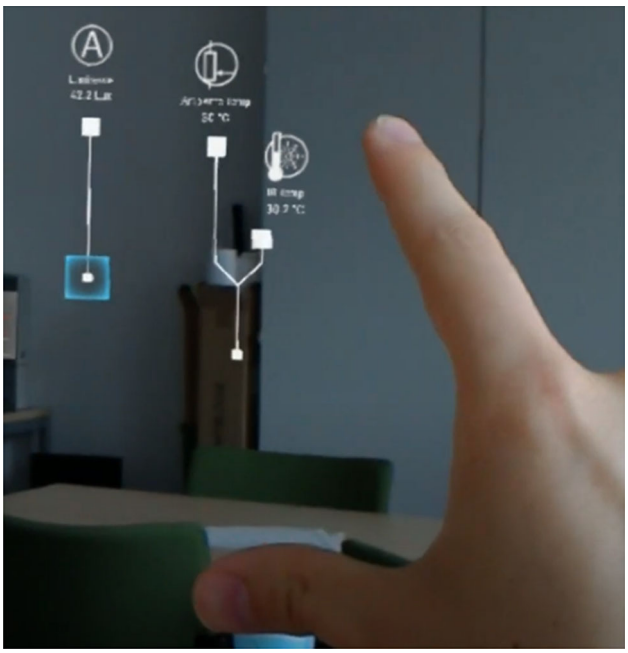


Fig. 8 Screenshot of HoloFlow illustrating an AR-based concrete syntax [90]

a screenshot of the Socio tool [78], which permits building models within a social network via conversation in (written) natural language with a bot. In the figure, the user inputs the sentence *"The order contains a payment method, which can be paypal, credit card or cash"*, and the bot reifies the user utterance as a model in the abstract syntax.

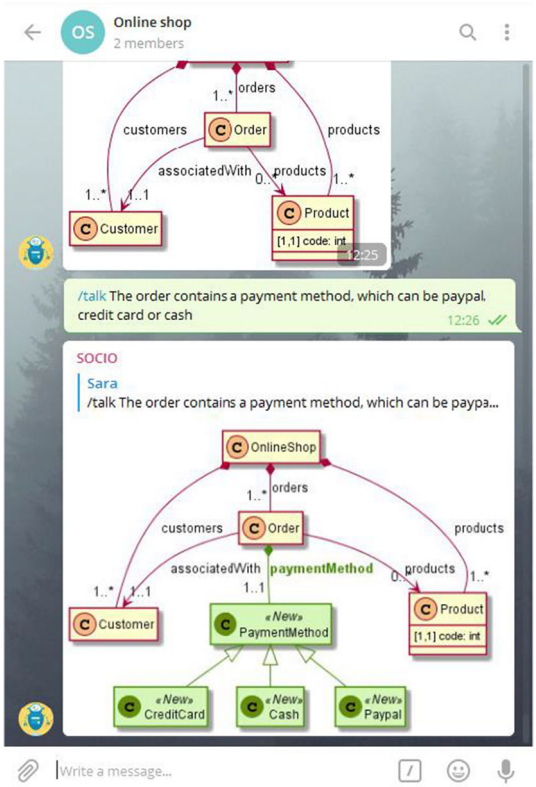


Fig. 9 Screenshot of Socio illustrating modelling via natural language conversation [78]

Table 5 classifies the relevant papers for our study according to the concrete syntax features in the feature model. We observe the following aspects:

Graphical concrete syntax. Graphical concrete syntaxes are used by the great majority of the tools (all except 4). DSL-comet has the particularity of using geolocated concrete syntax with user representation, and so, the model elements are placed on a map and get geolocated (cf. Fig. 6). All approaches provide a 2D representation, except Mind Mapping 3D [70], which uses 3D diagrams.
Sketching is supported by roughly 23% of the approaches with a graphical syntax. Specifically, Buchmann et al. [22], Calico [62], CollabTouch [46], DrawExpress [29], FlexiSketch [106], the Horus Method [2], Lekh Diagram [52], MobiDev [89], NetSketcher [10], and Sá et al. [85] support a sketch-based syntax which fits well in the touch-based screens of most devices. In addition, MobiDev proposes to sketch using pen and paper or a blackboard, take a picture with the mobile, and translate the drawing into a digital model automatically via shape recognition. The approach by Sá et al. [85] is aimed at designing prototypes of mobile apps in-situ, and it allows augmenting the sketches with behaviour (audio, video, and image elements).

Table 5 Classification of relevant papers according to the supported concrete syntax

Tool	Graphical	Tabular	Textual	NL
Approaches from the literature				
Archinotes [99]	•			
BPMN-Tool [84]	•			
Buchmann et al. [22]	Sketching			
Calico [62]	Sketching			
CEL [53]		•		
CollabTouch [46]	Sketching			
DSL-comet [102]	Geolocated User repr.			
Epidosite [54]			•	
FlexiSketch [106]	Sketching			
HoloFlow [90]	AR			
Horus Method [2]	Sketching			
icebricks [12]	•			
López-Jaquero et al. [56]		•		
Ma et al. [59]	•			
Metaphore [88]	•			
MicroApp [31]	•			
MobiDev [89]	Sketching	•		
NetSketcher [10]	Sketching			
Nolte et al. [75]	•			
Pounamu/Thin [112]	•			
Puzzle [26]	•			
Sá et al. [85]	Sketching			
Socio [78]	•			Text
TouchDevelop [95]			•	
YinYang [65]	•			
Tools from app stores				
Astah* UML Pad [6]	•			
Database Designer [50]	•			
DrawExpress [29]	Sketching			
Flowdia Diagrams [30]	•			
Halna Mind [38]	•			
Inspiration Maps [44]	•			
JSON Designer [25]	•			
KnowledgeBase Builder [43]	•			
Lekh Diagram [52]	Sketching			
Lucidchart [57]	•			
miMind [66]	•			
Mind Mapping 3D [70]	•			
Mind Meister [67]	•			
Mind Vector [68]	•			
Mindly [69]	•			
MindMaster [71]	•			
MindNode [72]	•			
Mindomo [73]	•			
OrgChart [77]	•			
PureFlow [1]	•			
SimpleMind [91]	•			
XMind [108]	•			

While graph-like diagrammatic notations are the norm, there are some exceptions. In particular, MicroApp [31], Puzzle [26] and YinYang [65] share a peculiar graphical style that consists of jigsaws representing concepts to be assembled following a specific logic. This is at the border between graphical and textual concrete syntaxes, and it is similar to the Scratch programming language [61]. Figure 10 illustrates this syntax with a simple example, inspired by MicroApp and Puzzle, for sending an SMS with a picture. The SMS jigsaw requires three other jigsaws, whose type is given by the colour (in the example, green corresponds to phone book actions, blue to text-related actions, and yellow to picture-related actions). Only jigsaws with the same colour can fit. Thus, for sending an SMS, the SMS jigsaw requires a target contact, a text and a picture, but the latter needs to be taken and saved in the gallery before.

Finally, HoloFlow [90] is the only modelling tool supporting a concrete syntax rendered in AR, where the model elements are attached to real-life objects.

Tabular concrete syntax. Tabular concrete syntaxes organize the model elements in a matrix or in menus, which is a popular representation for databases. Three approaches support this syntax. CEL [53] is an alternative to graphical class models that employs a tabular syntax. Elements are neither sketched nor displayed diagrammatically, but they are inserted into cells, which can be linked to represent relationships. This frees the user from arranging the model elements graphically, which can be cumbersome given the reduced screen size of some mobile devices. López-Jaquero et al. [56] provide a tabular concrete syntax organized in menus, where the users drag and drop the elements into different categories. Similarly, MobiDev also relies on menus to represent models.

Textual concrete syntax. Textual syntaxes are often discouraged for mobile devices due to the small size of their screen. However, Epidosite [54] and TouchDevelop [95] have tailored textual editors for letting novice programmers write scripts on mobile devices. The former has a focus on IoT devices, and the latter enables developing small general-purpose apps.

Natural language concrete syntax. A concrete syntax based on natural language permits using voice or text for creating and editing models. Socio [78] is the only tool that offers this kind of syntax. Models are constructed by conversing with a chatbot within social networks like Twitter or Telegram (cf. Fig. 9). The approach is not based on textual commands for creating elements, but on textual requirement descriptions that the bot interprets to create the abstract syntax of a model, which is shown to the user as a response. This prevents the users from dealing with the layout of the models.

4.1.2 Extended modelling

Some modelling tools permit users to include informal drawings or annotations on top of their models to convey additional meaning. We call this capability *extended modelling*. Figure 11 shows a feature diagram covering this aspect of a language syntax. This is an optional feature, whereby users are allowed to enrich models with *drawings* or *annotations* to informally convey extra information. Drawings refer

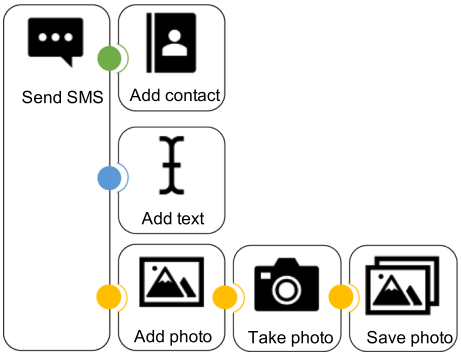


Fig. 10 Example schema representing a jigsaw-based concrete syntax

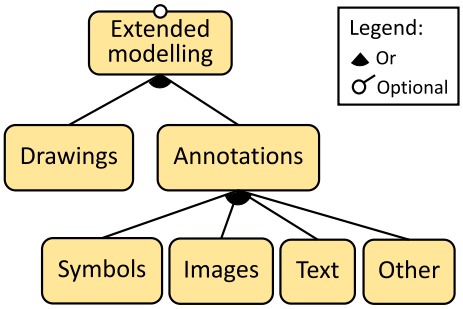


Fig. 11 Feature model for extended modelling on mobiles

to any kind of sketching that can be done over a model, like circling an element, or pointing by drawing an arrow. This can be useful in collaborative modelling sessions by enabling users to point to different parts of the model in a flexible way. Annotations are pieces of information that can be attached to model elements in the form of *symbols*, *images*, *text* comments or others. For instance, they can be used to report the model modality (e.g. fidelity of elements, uncertainty or purpose) [93] or to provide additional context to model elements by means of different media (e.g. pictures, audio). In general, extended modelling is only possible if the language syntax provides this extra flexibility.

Table 6 shows the approaches that support extended modelling. The low number of approaches may be due to the fact that most of them are meant to be used with no collaboration, or with live collaboration where users are physically close or use a communication tool like Skype, Microsoft Teams or Zoom. In these situations, extended modelling may not be seen as relevant. However, apart from collaboration, extended modelling can also help to increase the flexibility in modelling, as it enables documenting context, additional information about model elements, or tasks to be done in the next modelling session.

Annotations. All tools having extended modelling capabilities support some kind of annotation. Among them, Archinotes [99], Calico [62], DSL-comet [102], the Horus Method [2] and Lucidchart [57] provide collaboration mechanisms where different stakeholders can edit the same model at the same or different time. Archinotes permits attaching text, audio or video annotations to models. Calico provides the possibility to flag a diagram as "DO NOT ERASE" to let other users know that they should not alter its content. DSL-comet supports the use of annotations during live collaboration. The annotations can be notes, which can be geolocated and contain images, as well as temporal markers ('?' and '!') for catching the attention of other users during the collaboration. Horus allows sharing comments in natural language and attaching illustrations to model elements. Lucidchart permits attaching textual comments to model elements, in which case, a symbol is displayed to allow other users to answer the comment by clicking on the symbol. Mind Meister [67] gives the possibility to rate elements of a mind-map and add comments. Finally, Database Designer [50] and Sá et al. [85] do not support collaborative modelling, but users can add textual notes on the canvas for the former one, and on the created cards on the latter one.

Table 6 Classification according to the extended modelling ability of the approach

Tool	Drawings	Annotations
Approaches from the literature		
Archinotes [99]		•
Calico [62]	•	•
DSL-comet [102]	•	•
Horus Method [2]		•
Sá et al. [85]		•
Tools from app stores		
Database Designer [50]		•
Lucidchart [57]		•
Mind Meister [67]		•

Drawings. Informal drawings are less common in mobile modelling tools, being supported just by Calico and DSL-comet. The former also permits handwriting.

4.1.3 Semantics

Modelling languages encompass semantics, describing the meaning of models. Figure 12 categorizes the aspects of semantics that are relevant for our study. A modelling tool may or may not provide explicit semantics for models, and so, this feature is optional in the figure. If it does provide semantics, then, this can be either *generative* or based on *execution*. In the first case, semantics is implicitly defined via a code generator. In the second case, semantics is defined via execution of (part of) the model, for example via simu-

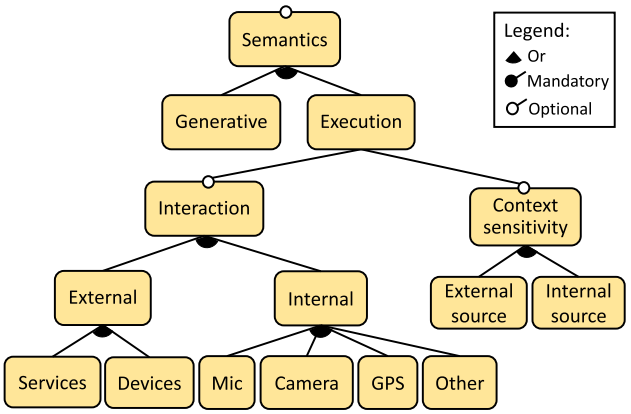


Fig. 12 Feature model for semantics of modelling languages on mobiles

Table 7 Classification according to the interaction capabilities

Tool	External		Internal
	Services	Devices	
Approaches from the literature			
Archinotes [99]	•		
DSL-comet [102]	•		•
Epidosite [54]	•	•	•
HoloFlow [90]		•	
Metaphore [88]		•	•
MicroApp [31]	•	•	•
Puzzle [26]	•	•	•
Sá et al. [85]			•
TouchDevelop [95]	•		•

lation (i.e. successive model transformation steps) or model interpretation.

For the purpose of our study, we identify two pertinent features of the model execution: *interaction* and *context sensitivity*. The interaction feature refers to the ability of a modelling language to communicate with *external services* (e.g. a remote weather forecast API, a social network), with *external devices* (e.g. IoT sensors for domotic systems), or with *internal* services of the mobile device (e.g. microphone, camera, GPS, or others). These interactions will occur during the model execution. Table 7 shows the approaches that support interaction.

The second feature of execution semantics we are interested in is context sensitivity, which refers to the ability to react to the context (e.g. changing the model) during model execution. The behaviour of a context-sensitive language relies on a context source and follows contextual rules. Context sources can be *external sources* or *internal sources*. The former are based on the interaction with remote components, such as APIs or IoT devices. The latter refer to internal components and sensors of the device on which the language is deployed, like the battery or the device network connection. Table 8 shows the context-sensitive approaches supporting the definition of contextual rules.

Table 8 Classification according to the sensitivity to the context

Tool	External source	Internal source
Approaches from the literature		
Epidosite [54]	•	
HoloFlow [90]	•	
MicroApp [31]	•	•
Puzzle [26]	•	
TouchDevelop [95]		•
YinYang [65]		•

In the following, we comment on the features related to semantics that the analysed tools provide.

Generative semantics. Among all the analysed tools, only three are generative: CEL [53] transforms its models into Java, Objective C or C++ code skeletons; Database Designer [50] generates SQL files; and JSON Designer [25] produces JSON code as this is a tool for visualizing and designing JSON structures.

External interaction. All approaches in Table 7 can interact with external services or devices, but the one by Sá et al. [85]. Specifically, Archinotes [99] permits sharing models on Facebook groups for notifying model changes to the active users on this social network. DSL-comet [102] provides an API broker for retrieving information from web services. Epidosite [54] leverages smartphones as hubs for IoT automation by means of scripts that can incorporate third-party mobile apps and web services. HoloFlow [90] supports interaction with IoT devices via AR. Metaphore [88] can detect items of the real world using external Bluetooth Low Energy (BLE) beacons as well as internal sensors of the device, and it assigns the items an ontological type and domain properties. MicroApp [31] and Puzzle [26] are similar to Epidosite, but instead of scripts, the users compose jigsaws that may use web services, domotic services and native services of the mobile. TouchDevelop [95] also provides access to web services, but it does not support interaction with IoT devices.

Internal interaction. All tools in Table 7 can interact with the internal components of the device, except Archinotes and HoloFlow. This interaction can serve different purposes. For geolocating model elements, DSL-comet uses the GPS of the smartphone, and Metaphore uses several sensors of the mobile device (e.g. accelerometer, gyroscope, compass, GPS). Epidosite permits using native services of the mobile (e.g. GPS, clock) in scripts. Similarly, MicroApp and Puzzle can incorporate native services (e.g. accelerometer, gyroscope, GPS, camera, microphone) in its jigsaw language. Sá et al. use the microphone and camera of the device to augment sketches with

input/output elements. Finally, TouchDevelop provides access to the internal services of the mobile as well.

Context sensitivity. Table 8 shows the tools that have sensitivity to the context. Epidosite and HoloFlow have been designed to interact with IoT devices, and so, their context sources are external. MicroApp can handle contextual stimuli from both internal and external sources. It adapts its jigsaw language depending on the network connection or user position, and users can use this context information to implement micro apps or benefit from other apps installed on the device, web services and IoT devices. Puzzle retrieves external context information from web services, IoT devices and other apps installed on the device like SMS messaging. Finally, TouchDevelop and YinYang [65] only employ internal sources. The former uses native functionalities of the device like the accelerometer or the GPS, and the latter can use conditional statements for adapting the situation within its script like a regular programming language.

4.1.4 Fine-grained access control

A model may need to be manipulated by multiple stakeholders. In such a case, some modelling languages may need mechanisms to control the access to their models or define different privileges depending on the user (e.g. some users may create models, while other users can only read the existing models).

DSL-comet [102] is the only tool with access control management at the modelling language level. The tool permits defining roles with different model editing permissions, and their assignment to users. As an example, let's consider the tourism DSL defined in DSL-comet in [20] (cf. Fig. 6). It defines the user roles "touristic guide" and "tourist". Touristic guides are allowed to create, modify and delete touristic route models, while tourists only have permission to consult (but not to change) existing route models. In addition to permissions at the DSL level, DSL-comet also proposes to grant permissions on the model editor functionalities. For instance, touristic guides are allowed to initiate a live collaboration session but tourists can only join existing sessions.

Some of the other approaches (Archinotes [99], Astah* UML Pad [6], BPMN-Tool [84], icebricks [12], Lekh Diagram [52], Lucidchart [57], miMind [66], Mindomo [73], Mind Mapping 3D [70], Mind Meister [67], Mind Vector [68], Mindly [69], MindMaster [71], MindNode [72], OrgChart [77], Pounamu/Thin [113], XMind [108]) provide access control at the tool level (e.g. via credentials when logging in the tool) but this does not have an impact on how the supported modelling languages are to be used.

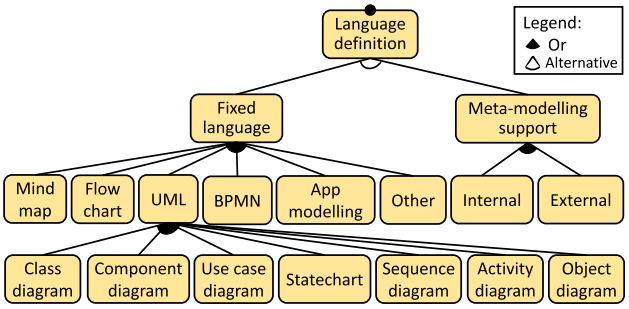


Fig. 13 Feature model for language definition dimension

4.2 Language definition

In the previous subsection, we have analysed the different aspects of the supported modelling languages. Now, we study how these languages are defined. As Fig. 13 shows, we have identified a dichotomy between tools that support fixed languages (feature *fixed language*) and those which offer users the possibility to create their own modelling languages (feature *meta-modelling support*). For the latter cases, we distinguish whether the language definition occurs within the mobile tool itself (feature *internal*) or externally, likely in a web-based or desktop application (feature *external*).

In Tables 9 and 10 , we can see that most approaches support one or a set of fixed languages, while only six approaches offer meta-modelling support. In the following paragraphs, we analyse these two tables in more detail.

Fixed language. Regarding tools and approaches focusing on fixed languages, works from the literature support mostly UML or BPMN, while mind-maps and flow charts are prominent in the tools from app stores. Some tools support fixed, custom languages specific to a domain. This is the case of Database Designer [50], which is specialized for database design; or MobiDev [89], which is devoted to user interface design. Other tools offer general-purpose notations, like CEL [53], which proposes a light version of UML class diagrams; JSON Designer [25], which is a JSON visualization and design tool; and OrgChart [77], which simplifies the organization chart maintenance in a tree structure. Other tools like Calico [62] and Lekh Diagram [52] favour sketch-based free modelling, not being constrained to a particular notation. In addition, the latter tool provides facilities for recognition of shapes and arrows for several notations, and allows creating objects by drag and drop from predefined palettes.

Another category of tools provide users with mobile editors for creating small apps like Epidosite [54], MicroApp [31], Puzzle [26], Sá et al. [85], TouchDevelop [95] and YinYang [65].

Table 9 Classification of papers according to their supported languages

Tool	UML	BPMN	Mind Map	Flowchart	App Modelling	Other
Approaches from the literature						
Archinotes [99]	Class, Component					
BPMN-Tool [84]		•				
Calico [62]						•
CEL [53]						•
CollabTouch [46]		•				
Epidosite [54]					•	
HoloFlow [90]		•				
Horus Method [2]						•
icebricks [12]		•				
Ma et al. [59]	Class, Use case, Component					
MicroApp [31]					•	
MobiDev [89]						•
NetSketcher [10]		•				
Nolte et al. [75]		•				
Puzzle [26]					•	
Sá et al. [85]					•	
Socio [78]	Class					
TouchDevelop [95]					•	
YinYang [65]					•	
Tools from app stores						
Astah* UML Pad [6]	Class					
Database Designer [50]						•
DrawExpress [29]	Class, Use case, Sequence, Statecharts	•	•	•		•
Flowdia Diagrams [30]	Class, Use case, Sequence, Activity	•	•	•		•
Halna Mind [38]			•			
Inspiration Maps [44]			•			
JSON Designer [25]						•
KnowledgeBase Builder [43]			•	•		
Lekh Diagram [52]				•		•
Lucidchart [57]	Class, Component, Use case, Object, Activity, Sequence	•	•	•		•
miMind [66]	Class, Statecharts		•	•		•
Mind Mapping 3D [70]			•			
Mind Meister [67]			•			
Mind Vector [68]			•			
Mindly [69]			•			
MindMaster [71]			•			
MindNode [72]			•			
Mindomo [73]			•			
OrgChart [77]						•
PureFlow [1]				•		
SimpleMind [91]			•			
XMind [108]			•			

While most tools support just one language, a few of them support several languages. Specifically, DrawExpress [29], Flowdia Diagrams [30], Lucidchart [57] and miMind [66] offer a large variety of languages from UML or network designs to business processes and mind-maps.

Internal meta-modelling support. There are just two approaches that enable meta-modelling on the mobile device. In FlexiSketch [106], modelling and meta-modelling can be done in any order. For example, a user can sketch a model informally, and then upgrade some of the elements as concepts of a language, adding them to the palette (cf. right part of Fig. 7). Metaphore [88] is a tool for domain-specific, positioning-based modelling. In a first step, a domain meta-model needs to be defined, and its classes can be attached to BLE beacons. Then, when a model is created, its objects can be attached to a physical position via the sensors of the mobile (GPS, compass) or BLE beacons.

External meta-modelling support. There are four approaches that permit defining languages in external (typically desktop) tools, which produce the necessary infrastructure to run the domain-specific editor on a mobile device. In DSL-comet, the meta-models of the DSLs are defined using the eclipse modelling framework

Table 10 Classification according to the meta-modelling support

Tool	Meta-modelling support
Approaches from the literature	
Buchmann et al. [22]	External
DSL-comet [102]	External
FlexiSketch [106]	Internal
López-Jaquero et al. [56]	External
Metaphore [88]	Internal
Pounamu/Thin [112]	External

(EMF) [23]. These meta-models are annotated with information about graphical syntax, geopositioning, access control or external interaction, and then uploaded on a server to allow their use inside DSL-comet.

While DSL-comet interprets the language definition on the mobile, the other approaches with external meta-modelling support generate artefacts to be run on the device. In particular, the meta-modelling tool Pounamu/Thin [113] produces Java-based multi-view editors for graphical modelling languages. It was initially designed as a desktop application, which later was extended to generate web and (Nokia) mobile clients [36,112,113] (and see also the follow-up tool Marama/Thin [35]). López-Jaquero et al. [56] synthesize multi-touch domain-specific editors. They use EMF to define domain meta-models, which need to be complemented with presentation models to specify the icons and fonts to represent the different object types. Then, they use a set of heuristics to generate a touch interface based on menus and forms. Finally, Buchmann et al. [22] propose a framework to create sketch-based editors for Android devices. The framework, which requires defining the meta-models in Java, has been used to extend the UML tool Valkyrie [21] with sketching capabilities.

Please note that, while some works have automated the creation of domain-specific mobile modelling tools (e.g. for UML [22] or for activity planning [56]) using external generative approaches, these generated tools would need to be uploaded into an app store for their distribution. Instead, internal meta-modelling mechanisms and external interpreted approaches result in truly mobile meta-modelling tools.

4.3 Tooling

After designing the modelling language of a mobile modelling solution, the next logical step is to decide where and how it will be used, i.e. we need a modelling tool. In this section, we analyse tooling aspects of the retrieved approaches in relation to their deployment (Sect. 4.3.1), collaboration support (Sect. 4.3.2), interoperability mechanisms (Sect. 4.3.3) and user interaction (Sect. 4.3.4).

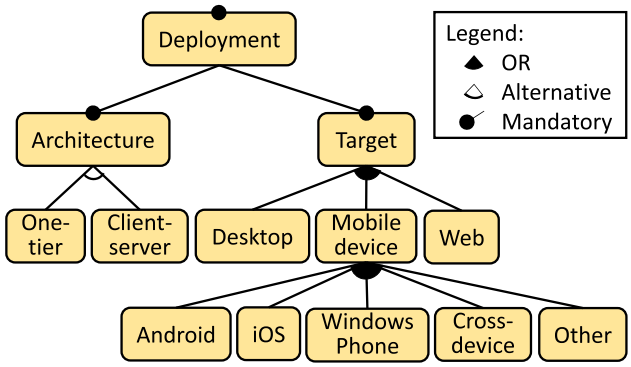


Fig. 14 Feature model for tool deployment

4.3.1 Deployment

The deployment of a modelling tool is an important issue since it may affect the usability and target audience of the tool. Figure 14 depicts the options that we consider for deployment. First, we look at the *architecture* of the solution, which can be *one-tier* (i.e. the tool runs stand-alone) or *client-server* (i.e. the tool is a client that requires communication with a server). Second, we look at the deployment *target*, which can be a *desktop* computer, a *mobile device*, the *web*, or several of them. When the deployment is on mobile devices, we distinguish the target platform (*Android*, *iOS*, *Windows Phone*, *Other*) or whether the approach is *cross-device* (i.e. it can run on several systems). With *Other*, we may refer e.g. to phone platforms (like the old Nokia's Symbian) or to head-mounted mobile devices like HoloLens.

Table 11 classifies the surveyed tools according to their deployment.

One-tier architecture. The tools that do not use external services or remote collaboration typically have one-tier architectures. These tools are Astah* UML Pad [6], Buchmann et al. [22], CEL [53], Database Designer [50], DrawExpress [29], FlexiSketch [106], Flowdia Diagrams [30], Halna Mind [38], Inspiration Maps [44], JSON Designer [25], KnowledgeBase Builder [43], Lekh Diagram [52], Mind Mapping 3D [70], Mindly [69], MobiDev [89], NetSketcher [10], PureFlow [1], SimpleMind [91] and YinYang [65]. An exception is Metaphore [88], which communicates using Bluetooth with external devices (BLE beacons).

Client-server architecture. The other tools use client-server architectures either because they have a backend or a remote database (BPMN-Tool [84], Calico [62], DSL-comet [102], Lucidchart [57], MindMaster [71], Mind Meister [67], Mind Vector [68], MindNode [72], Nolte et al. [75], OrgChart [77], Pounamu/ Thin [112], Socio [78]), or, and maybe in addition, because they communicate with external services (Archinotes [99],

Epidosite [54], HoloFlow [90], MicroApp [31], Puzzle [26], TouchDevelop [95]). In Table 11, we mark as *unknown* five academic works for which we were not able to identify the architecture.

Mobile device. Table 11 shows that, unsurprisingly, most of the tools are deployed in mobile devices. We can distinguish three categories that justify this choice in the case of academic tools.

The first reason is the desire to create editors along with their compilers or interpreters for developing on mobile devices. This includes several tools like Epidosite [54], MicroApp [31], MobiDev [89], Puzzle [26], TouchDevelop [95] and YinYang [65]. According to recent studies [97,105], the number of mobile users in 2020 is higher than the number of desktop users. Thus, mobile apps may bring development activities closer to a wider audience.

The second reason is the need for mobility, e.g. in situations where access to a desktop is not possible. This includes scenarios in which modelling occurs at the location of the system being modelled (e.g. a smart factory, a farm). Archinotes [99] enables collaboration between stakeholders who might be in different geographical areas and might not have access to a desktop. Icebricks [12] was initially implemented as a web application, but now it has been adapted as a hybrid app for modelling on mobile devices. This is also the case for BPMN-Tool [84]. López-Jaquero et al. [56], DSL-comet [102], Metaphore [88], Pounamu/Thin [112] and Sá et al. [85] are designed for modelling on-site at the system location. The approach by López-Jaquero et al. is illustrated with an app for the treatment of people with acquired brain injury by psychologists or physiotherapists, who need to model while being with their patients. DSL-comet permits geolocating model elements on a map, with constraints relative to their geoposition. Sá et al. want to provide the user with a tool for creating prototypes at the same location where the real app will be used. Metaphore can position the elements of models in the real world using BLE beacons that communicate with the application via Bluetooth. The tool is useful to create models for domotics, event planning or construction, where the user needs to be at the location of the system being modelled to profit from the short-distance communication capability of the Bluetooth.

The third reason is the need to support tactile interaction. This is the case of sketching tools like Buchmann et al. [22], Calico [62], CEL [53], CollabTouch [46], DrawExpress [29], FlexiSketch [106], Lekh Diagram [52] and NetSketcher [10]. All these tools except Calico, CEL and NetSketcher are also deployed on desktop computers to support cross-device collaboration with a desktop computer, an interactive whiteboard or a tabletop.

Table 11 Classification according to deployment

Tool	Architecture	Target		
		Web	Desktop	Mobile device
Approaches from the literature				
Archinotes [99]	Client-server			•
BPMN-Tool [84]	Client-server			iOS
Buchmann et al. [22]	One-tier		•	Android
Calico [62]	Client-server			•
CEL [53]	One-tier			iOS
CollabTouch [46]	Unknown		•	•
DSL-comet [102]	Client-server			iOS
Epidosite [54]	Client-server			Android
FlexiSketch [106]	One-tier		•	Android
HoloFlow [90]	Client-server			Other (HoloLens)
Horus Method [2]	Unknown	•		
icebricks [12]	Unknown	•		Cross-device
López-Jaquero et al. [56]	Unknown			•
Ma et al. [59]	Unknown	•		
Metaphore [88]	One-tier			iOS
MicroApp [31]	Client-server			•
MobiDev [89]	One-tier			Android
NetSketcher [10]	One-tier			•
Nolte et al. [75]	Client-server	•		
Pounamu/Thin [112]	Client-server	•		Other (Nokia's Symbian)
Puzzle [26]	Client-server		•	Cross-device
Sá et al. [85]	Client-server		•	Windows Phone
Socio [78]	Client-server	•	•	Cross-device
TouchDevelop [95]	Client-server			Windows Phone
YinYang [65]	One-tier			•
Tools from app stores				
Astah* UML Pad [6]	One-tier		•	iOS
Database Designer [50]	One-tier			Android
DrawExpress [29]	One-tier			Cross-device
Flowdia Diagrams [30]	One-tier			Cross-device
Halna Mind [38]	One-tier			Android
Inspiration Maps [44]	One-tier			iOS
JSON Designer [25]	One-tier			iOS
KnowledgeBase Builder [43]	One-tier			Android
Lekh Diagram [52]	One-tier			Cross-device
Lucidchart [57]	Client-server	•		Cross-device
miMind [66]	Client-server		•	Cross-device
Mind Mapping 3D [70]	One-tier			Cross-device
Mind Meister [67]	Client-server	•		Cross-device
Mind Vector [68]	Client-server	•		Cross-device, Other (HoloLens)
Mindly [69]	One-tier		•	Cross-device
MindMaster [71]	Client-server	•	•	Cross-device
MindNode [72]	Client-server		•	iOS
Mindomo [73]	Client-server	•	•	Cross-device
OrgChart [77]	Client-server		•	iOS
PureFlow [1]	One-tier		•	iOS
SimpleMind [91]	One-tier		•	Cross-device
XMind [108]	Client-server	•	•	Cross-device

A "•" on column *Mobile device* indicates that the deployment platform is unknown

We also hypothesize on the reasons why the tools in the app stores exist. The first reason is to permit their use on mobiles as this is a popular platform (e.g. Database Designer [50], Flowdia Diagrams [30], Halna Mind [38], Inspiration Maps [44], JSON Designer [25], KnowledgeBase Builder [43] and Mind Mapping 3D [70]). In other cases, it is to enable modelling across multiple platforms (e.g. Astah* UML Pad [6], Lucidchart [57], miMind [66], Mind Vector [68], MindNode [72], Mindly [69], OrgChart [77], PureFlow [1] and SimpleMind [91]). The last reason is for adapting a well-known desktop/web application to a mobile version, in order to widen the existing user community of the tool.

Web. Table 11 shows that three tools can only be deployed as web applications. The Horus Method [2] allows draw-

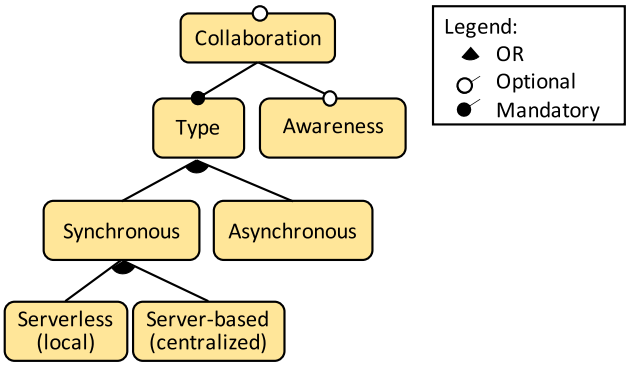


Fig. 15 Feature model for collaboration on mobile tools

ing elements on mobile and on desktop browsers, as well as drag and drop; Nolte et al. [75] present a prototype for browsers of large interactive display walls, digital tabletops and personal mobile devices; and Ma et al. [59] propose a web architecture based on HTML 5. We have included these tools in the survey because they are specially designed for their use on mobile devices, and their authors wanted them to be cross-platform, with no need for installation on the devices.

Other. HoloFlow [90] is not deployed on phones or tablets but on smart glasses, more specifically on Microsoft's HoloLens. This gives an immersive modelling experience, whereby the user needs to be at the system location to create virtual objects attached to real-life objects. Mind Vector [68] also offers a version of the tool, called Holo Mind Vector, that is deployed on HoloLens.

The tools Socio [78], MindMaster [71], Mindomo [73] and XMind [108] are deployed on the three regular platforms (web, desktop and mobile devices). Socio [78] can be used on social networks (Twitter and Telegram) which are available via desktop applications, mobile browsers and dedicated mobile apps.

4.3.2 Collaboration

Collaboration is an optional functionality, useful in many modelling scenarios. Figure 15 depicts the features that we consider regarding collaboration in mobile modelling tools. While these aspects are enough for our purposes, the interested reader can consult other more detailed taxonomies devised for classifying desktop and web-based collaborative modelling approaches like [32,64].

Modelling is often a collaborative activity that involves many actors [32]. The best example is the traditional way of modelling using pen and paper or a whiteboard, where several stakeholders meet at the same location to sketch diagrams [110]. An app could be used in this scenario if the same model can be accessed by several users. This scenario employs *synchronous* collaboration because different stakeholders collaborate simultaneously on the same model. Synchronous collaboration can be *serverless*, or local, if the stakeholders are located closely and they either share the same screen or connect their devices using an ad-hoc server via Bluetooth or Wi-Fi. Alternatively, synchronous collaboration can be *server-based*, in which case, the stakeholders can be at different locations when having a live collaboration session. In contrast, in *asynchronous* collaboration, the stakeholders can work on a model at different moments, likely offline, and the model gets synchronized when its changes are committed to the server.

Table 12 Classification of approaches according to collaboration support

Tool	Type		Awareness
	Asynchronous	Synchronous	
Approaches from the literature			
Archinotes [99]	•	Centralized	•
BPMN-Tool [84]		Centralized	
Calico [62]	•	Centralized	•
CollabTouch [46]	•	Serverless	•
DSL-comet [102]	•	Serverless	•
FlexiSketch [106]	•	Serverless	•
Horus Method [2]	•	Centralized	
NetSketcher [10]	•	Serverless	
Nolte et al. [75]	•	Serverless	
Pounamu/Thin [112]	•	Centralized	
Socio [78]	•	Centralized	•
Tools from app stores			
Lucidchart [57]	•	Centralized	•
Mind Meister [67]	•	Centralized	

In both types of collaboration, it is also interesting to study the capability of the tools to make users *aware* of each other actions. According to Dourish and Bellotti [27], *awareness* in the context of collaborative applications refers to mechanisms for "*understanding the activities of others, which provides a context for your own activity*". In collaborative systems, awareness mechanisms have been defined using a variety of means, such as informational (e.g. where collaborators inform each other of the activities performed), role restrictive (e.g. by assigning roles to users, which are then responsible for certain tasks), and others (see [3] for a comprehensive classification of awareness mechanisms).

Collaboration can profit from the extended modelling capabilities discussed in Sect. 4.1.1. Moreover, it triggers some questions regarding consistency, concurrency and security, the last of which can be tackled via access control, as explained in Sect. 4.1.4.

Table 12 classifies the approaches that support some form of collaboration with respect to the features of Fig. 15. Most collaborative tools offer both synchronous and asynchronous modes.

Asynchronous collaboration. The tools that feature asynchronous collaboration have different motivations for this choice:

- Allowing users to work at different moments and share their work on a repository. This is the case of Archinotes [99], DSL-comet [102], Lucidchart [57], Mind Meister [67], NetSketcher [10] and Pounamu/Thin [112].
- Enabling individual work to be merged with the work of other collaborators during synchronous collaborative sessions. This is the case of Calico [62], CollabTouch [46], FlexiSketch [106] and Nolte et al. [75].
- Preventing information loss in case of an internet outage, as in the Horus Method [2]. This tool has a smart internal cloud storage mechanism that updates the application when the device is online and there is an update available on the server.

Serverless synchronous collaboration. We identify three categories of serverless approaches attending to their techniques and supported scenarios:

- Creation of an ad-hoc network hosted on a device, to which other devices connect via Wi-Fi, like in DSL-comet [102] and NetSketcher [10]. In DSL-comet, users collaborate via a token-based protocol, where the user initiating the modelling session grants the token for model modification to the users in the session. Modelling in NetSketcher is concurrent, and users are identified with a name and colour to know who sketched what.
- Support of large screens like interactive tables and whiteboards, enabling several users to work on the same device at the same time. This is the case of CollabTouch [46] and Nolte et al. [75].
- FlexiSketch [106] supports the creation of an ad-hoc network hosted on a desktop and the display of the model on a large screen (like in the previous category), but each user interacts from a mobile device (as in the first category).

Server-based synchronous collaboration. If collaboration is server-based, the mobile clients have access to the models shared on a server. Archinotes [99], Calico [62], the Horus Method [2], Lucidchart [57] and Mind Meister [67] support concurrent editing. In Socio [78], the users can collaboratively and synchronously edit models via natural language messages within a social network. In addition, the tool has a model server and a mechanism supporting the creation of branches with model variants [78]—like in code versioning systems—to enable asynchronous collaboration.

Awareness. The tools that provide awareness mechanisms only apply them in the context of synchronous collaboration. In some cases, awareness is limited to broadcasting each user action on the model (like in the case of CollabTouch [46] and Lucidchart [57]). However, some tools provide additional functionalities. Archinotes [99] tracks all the changes within a live session to enable consulting them later. Calico [62] shows the name of the user currently working on a canvas. In DSL-comet [102], only one designated person can edit at a time, but the other users can use extended modelling features to react to these changes. FlexiSketch [106] implements a non-optimistic locking mechanism to prevent potential inconsistencies and conflicts by prohibiting the modification of the same element by more than one user. In this situation, the element is shown with a red background and does not react to the inputs of other users. Socio [78] is used within social networks via NL messages. Hence, every user can read and reply to the messages of other users, since collaboration occurs within a chat room. In addition, the messages that trigger the creation of model objects are attached to the objects, and it is possible to obtain statistics of the percentage of authorship of every user.

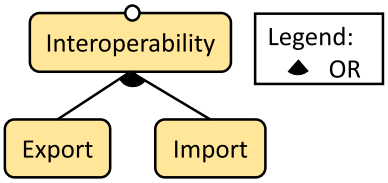


Fig. 16 Feature model for interoperability of mobile modelling tools

4.3.3 Interoperability

An important aspect for the success and acceptance of a modelling tool is its ability to interoperate with other tools, especially via modelling standards and widely accepted storage formats, like the XML meta-data interchange (XMI) [107].

Figure 16 shows the two aspects we consider for interoperability: whether the models built with the tool can be exported to other platforms, such as desktop modelling tools (feature *export*), and whether models built in other applications can be imported into the mobile modelling tool (feature *import*).

Table 13 shows the approaches with interoperability capabilities, which are very few.

Import. Only three tools provide import facilities from standard formats: DSL-Comet [102] can import XMI models, Database Designer [50] can import database models from SQL, and JSON Designer [25] can import JSON files from both the mobile device and a URL. In addition, Lucidchart [57] can import diagrams with the Microsoft Visio proprietary format.

Table 13 Classification of mobile modelling tools according to their interoperability

Tool	Export	Import
Approaches from the literature		
BPMN-Tool [84]	GraphML	
CEL [53]	C++, Java, Objective C	
DSL-comet [102]	XMI	XMI
FlexiSketch [106]	GOPPRR	
Horus Method [2]	PNML	
NetSketcher [10]	XPDL	
Socio [78]	XMI	
Tools from app stores		
Database Designer [50]	SQL	SQL
JSON Designer [25]	JSON	JSON
Lucidchart [57]	Visio	Visio

Export. The file formats used to export the models are very diverse, but we distinguish three groups. The first one includes tools using file formats and standards specific to a narrow domain or particular tool. This includes NetSketcher [10] with the serialization format for BPMN called XML Process Definition Language (XPDL) [109]; the Horus Method [2] with the Petri Net Markup Language (PNML) [80]; FlexiSketch [106], which exports models and meta-models as XML files that can be converted into the GOPPRR format of MetaEdit+ [48]; and Lucidchart, which allows exporting diagrams with the Microsoft Visio format.

The second group comprises tools generating code. This is the case of CEL [53], which can export models into C++, Java or Objective C; and Database Designer, which exports database models into SQL.

The last group consists of the tools that export to generic formats, like XMI, GraphML [34] or JSON. Socio [78] and DSL-comet export models to XMI; BPMN-Tool [84] exports BPMN models as GraphML documents; and JSON Designer exports models into JSON files.

4.3.4 User interaction

Finally, we analyse the user interaction possibilities for modelling in mobility. These are depicted in Fig. 17.

The traditional computer-assisted way to building models in desktop applications is the use of keyboard and mouse. In the case of mobile devices, they lack a mouse, so interaction with the *touch screen* or the (virtual) *keyboard* are used instead. Moreover, modelling in a mobile device can benefit from its embedded *sensors*, like the camera (e.g. for augmented reality), the microphone (e.g. for voice-based interaction), the GPS (e.g. for the geolocation of model elements), the accelerometer, or other sensors (e.g. for the positioning of elements via Bluetooth). In addition, the user interaction of some tools is *customizable*, for example, concerning the definition of gestures needed to create or delete model elements on a mobile device.

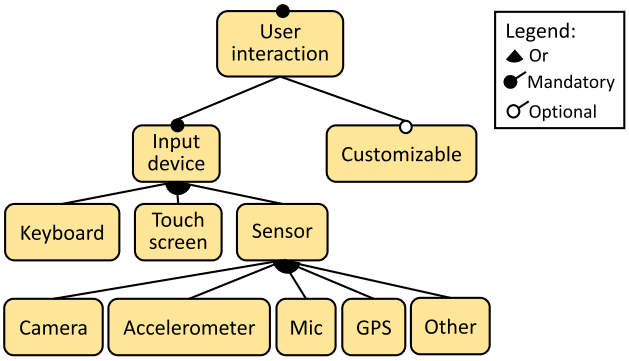


Fig. 17 Feature model for user interaction in tooling

Table 14 classifies the approaches based on their user interaction capabilities. Almost all of them rely on the touch screen and the keyboard for the interaction. Hence, in the following, we comment on the tools that only support one of these two forms of interaction, as well as on tools making use of sensors.

Keyboard. Pounamu/Thin [112] and Socio [78] limit their interaction to the keyboard. Pounamu/Thin is one of the oldest approaches from the days when touch-enabled devices were not available, but it runs on Nokia devices from 2006. Socio supports interaction via textual natural language. For this purpose, it uses just the keyboard, but since it is a modelling chatbot, it has the potential to use spoken natural language via the microphone as well.

Touch screen. Calico [62] does not use the keyboard, but it relies purely on sketching, to emulate the feel of modelling on a whiteboard.

Sensor. Some approaches enable the use of sensors as input device, most notably the camera. HoloFlow [90] needs the camera for using augmented reality, and it features gesture recognition for interacting with the model. DSL-comet [102] permits annotating the model elements with pictures taken with the camera, as well as using the GPS of the device for geopositioning model elements. Archinotes [99] and Sá et al. [85] can use the camera and the microphone, the former for annotating model elements with pictures, videos and audio files, and the latter for adding the expected input/output of behaviours. MobiDev [89] proposes to use the camera to take a picture of a model sketched on paper and then convert it to the tool digital format. Metaphore [88] uses BLE beacons, the compass and the accelerometer to position selected model elements. CollabTouch [46] uses the brightness sensor of the camera and the accelerometer to detect if a tablet is touching the table screen, meaning that the user aims to merge his/her model with the group's model.

Table 14 Classification of approaches according to their user interaction capabilities for modelling

Tool	Custom.	Input device		
		Sensor	Touch screen	Keyboard
Approaches from the literature				
Archinotes [99]		Camera, Mic	•	•
BPMN-Tool [84]			•	•
Buchmann et al. [22]	•		•	•
Calico [62]			•	
CEL [53]			•	•
CollabTouch [46]		Camera, Accelerometer	•	•
DSL-comet [102]		Camera, GPS	•	•
Epidosite [54]			•	•
FlexiSketch [106]			•	•
HoloFlow [90]		Camera, Other		
Horus Method [2]			•	•
icebricks [12]			•	•
López-Jaquero et al. [56]			•	•
Ma et al. [59]			•	•
Metaphore [88]		Accelerometer, Other	•	•
MicroApp [31]	•		•	•
MobiDev [89]		Camera	•	•
NetSketcher [10]			•	•
Nolte et al. [75]			•	•
Pounamu/Thin [112]				•
Puzzle [26]			•	•
Sá et al. [85]		Camera, Mic	•	•
Socio [78]				•
TouchDevelop [95]			•	•
YinYang [65]			•	•
Tools from app stores				
Astah* UML Pad [6]			•	•
Database Designer [50]			•	•
DrawExpress [29]			•	•
Flowdia Diagrams [30]			•	•
Halna Mind [38]			•	•
Inspiration Maps [44]			•	•
JSON Designer [25]			•	•
KnowledgeBase Builder [43]			•	•
Lekh Diagram [52]			•	•
Lucidchart [57]			•	•
miMind [66]			•	•
Mind Mapping 3D [70]			•	•
Mind Meister [67]			•	•
Mind Vector [68]			•	•
Mindly [69]			•	•
MindMaster [71]			•	•
MindNode [72]			•	•
Mindomo [73]			•	•
OrgChart [77]			•	•
PureFlow [1]			•	•
SimpleMind [91]			•	•
XMind [108]			•	•

Overall, the interaction possibilities in mobile devices are many and can exploit not only touch gestures but also information from sensors of the mobile device. For this reason, some frameworks have been proposed to facilitate the configuration of the interaction for arbitrary mobile apps (not specifically for modelling). As an example, the toolbox Milkyway [51] permits prototyping the interaction of collaborative mobile apps based on the information of the device sensors and programming by example. The toolbox has been used to provide some interesting interaction capabilities to the Business Model Canvas (BMC), such as initiating a collaboration between several users when detecting certain blocks using image recognition, or sharing and merging content by pouring it from one device to another. Please note that the latter gesture is also used by CollabTouch [46] for a similar purpose.

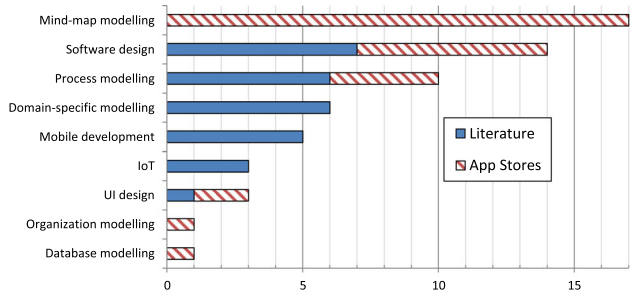


Fig. 18 Domains targeted by the approaches and tools

Customizable. Only two tools support customizability. On the one hand, MicroApp [31] allows personalizing the touch gesture required for triggering the modelled micro apps. On the other hand, the approach of Buchmann et al. [22] supports the use of the *GestureBuilder* Android app to specify sketching gestures to create the different object types. These gestures are then recognized by the synthesized domain-specific editors.

5 Discussion

This section discusses the results of our mapping study by answering the research questions outlined in Sect. 3. Section 5.1 answers RQ1 (*What are the domains where mobile modelling has been applied?*), Sect. 5.2 answers RQ2 (*What are the motivations argued for mobile modelling?*), Sect. 5.3 answers RQ3 (*What are the characteristic features of existing mobile modelling approaches?*), and Sect. 5.4 answers RQ4 (*What are the gaps in current approaches, and the research opportunities in mobile modelling?*). The first three questions cover the why and how of the current practice, and the fourth question tries to look beyond that. Finally, Sect. 5.5 summarizes the main take aways of the analysis and Sect. 5.6 argues on possible threats to the validity of our study.

5.1 RQ1: What are the domains where mobile modelling has been applied?

Table 15 presents a summary of the application domains the analysed tools have tackled, and Fig. 18 displays this information graphically. As depicted in Fig. 18, the existing approaches in the literature have been applied to six application domains: software design (7), process modelling (6), domain-specific modelling (6), mobile development (5), IoT (3), and UI design (1). Regarding tools from app repositories, they target mind-map modelling (17 tools), software

Table 15 Domains targeted by the approaches and tools

Domain	Approaches from literature	Tools from app stores
Mind-map modelling		DrawExpress [29], Flowdia Diagrams [30] Halna Mind [38], Inspiration Maps [44], KnowledgeBase Builder [43], Lekh Diagram [52], Lucidchart [57], miMind [66], Mind Mapping 3D [70], Mind Meister [67], Mind Vector [68], Mindly [69], MindMaster [71], MindNode [72], Mindomo [73], SimpleMind [91], XMind [108]
Software design	Archinotes [99], Buchmann et al. [22], Calico [62], CEL [53], FlexiSketch [106], Ma et al. [59], Socio [78]	Astah* UML Pad [6], DrawExpress [29], Flowdia Diagrams [30], JSON Designer [25], Lekh Diagram [52], Lucidchart [57], PureFlow [1]
Process modelling	BPMN-Tool [84], CollabTouch [46], Horus Method [2], icebricks [12], NetSketcher [10], Nolte et al. [75]	DrawExpress [29], Flowdia Diagrams [30], Lekh Diagram [52], Lucidchart [57]
Domain-specific modelling	Buchmann et al. [22], DSLcomet [102], FlexiSketch [106], López-Jaquero et al. [56], Metaphore [88], Pounamu/Thin [112]	
Mobile development	MicroApp [31], MobiDev [89], Sá et al. [85], TouchDevelop [95], YinYang [65]	
IoT	Epidosite [54], HoloFlow [90], Puzzle [26]	
UI design	Calico [62]	DrawExpress [29], Lekh Diagram [52]
Organization modelling		OrgChart [77]
Database modelling		Database Designer [50]

design (7), process modelling (4), UI design (2), organization modelling (1) and database modelling (1). Some of the approaches and tools serve different domains.

Mind-map modelling is the most widely targeted domain by tools from app stores. One possible reason of its prevalence is that this domain may be especially suitable for mobile devices due to the simple syntax of mind-maps, and the possibility for outlining and collecting ideas everywhere (without the need for a desktop computer). Mind-maps might be popular in app stores because their use does not require from specialized knowledge from users, and the generality of the notation makes it applicable to different areas. Conversely, other more specialized domains, like mobile development, IoT, or support for domain-specific modelling are only targeted by academic approaches. A possible reason is that they are more demanding for the users and focus on narrower domains, which is the opposite scenario than for mind-maps.

Software design, process modelling and UI design are domains covered by both approaches in the literature and app store tools. Software design is normally approached by supporting UML diagrams (cf. Table 10), but in the case of Archinotes [99], it targets architectural design. We have distinguished software design from some of its subareas, like UI design (typically supported by sketching of UI mockups) and IoT (with tools offering different ways to customize the interaction with external devices).

5.2 RQ2: What are the motivations argued for mobile modelling?

We have elicited from the papers the motivations that authors have argued for developing modelling tools on mobile devices. Figure 19 summarizes the motivations, where the horizontal axis corresponds to the number of approaches that give a particular motivation. The graphic only considers academic papers, since tools from app stores do not provide any explicit motivation.

We can see that the most recurrent motivation is collaboration (11 out of 25 approaches), which is mentioned by several approaches in the domains of process modelling, software design and domain-specific modelling. The possibility to use the application in mobility from everywhere (8) and facilitate end-user development to novice users (7) are also popular motivations. In two cases, end-user development targets mobile development, and the rationale is to use the same device to both develop and test mobile applications. In general, most end-user development tools provide graphical languages because they are easier to use in the reduced screen of mobile devices. However, we also find jigsaw-like and textual languages for the development of small applications by non-programmers on the mobile. Even though one can argue that these languages may be used on desktop computers or the web, the authors of the papers explicitly decided to focus the deployment on mobile devices because the targeted audience has more chances to access a mobile device than a computer.

On-site modelling is mentioned in the fourth place (4) in the following scenarios: the agile creation of models while interviewing customers or domain experts, the configuration of IoT devices in-place by interacting with their sensors, and providing virtual objects with precise coordinates from the real world (e.g. to build models of convention centers or smart factories). The latter is very much related to indoor positioning, which is also given as motivation by 1 work.

The fifth reason for mobile modelling is the high number of smartphone users (3). Then, it follows creative modelling (2), which refers to being able to build design models more flexibly by relaxing their conformance rules; this is in contrast to many modelling desktop applications which always require correct models and hence may be too rigid for some scenarios such as early design discussion [37]. Natural user interaction, either based on sketching (2) or touch gestures (2), is also a motivation of some authors' works. Actually, even though only 2 papers explicitly mention sketching as one of the fundamental reasons to adopt modelling on mobile phones, this kind of syntax is supported by 8 (out of 25) aca-

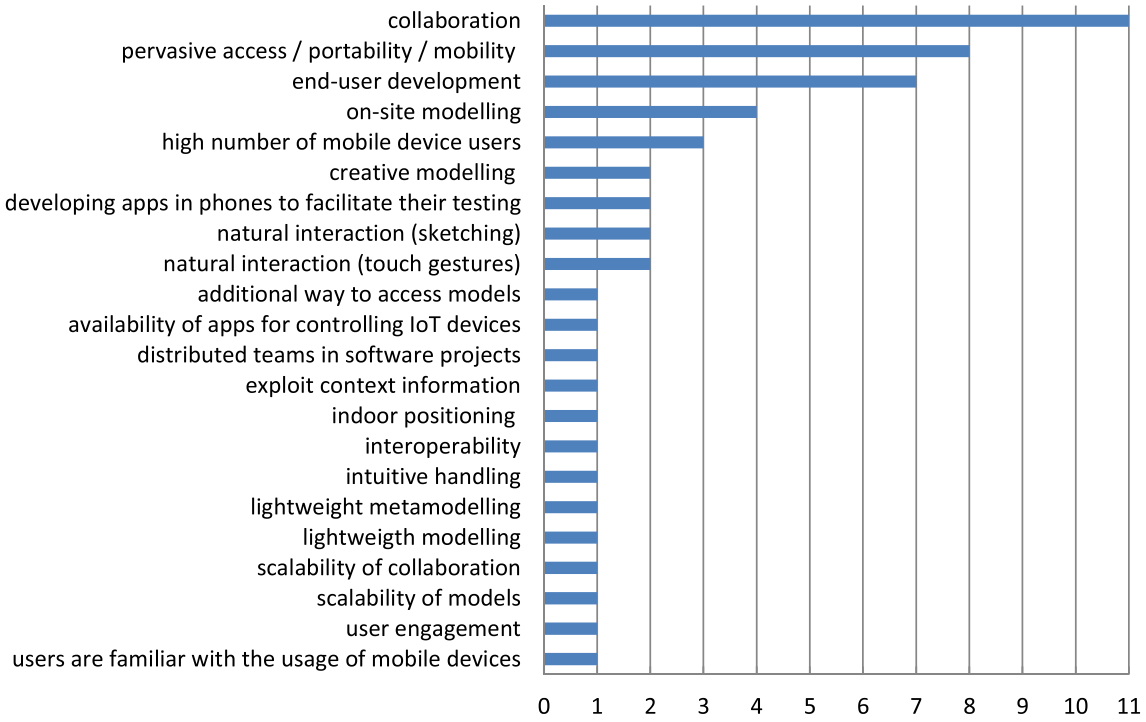


Fig. 19 Authors' motivations for proposing a mobile modelling approach

demic tools, and by 2 tools from app stores. Touch screens not only permit reproducing the pen and paper feel in the early specification process, but they eventually avoid having to make a model by hand and remake it on the machine, while enabling model sharing among several potentially distributed stakeholders. Moreover, touch screens allow manipulating models using a richer variety of gestures (like tap, drag, flick, swipe, pinch, shake, rotate, touch and hold) than standard mouse and keyboard applications.

One of the papers motivates mobile modelling by the need to obtain and profit from context information, which is possible by the sensors and components embedded in the mobile devices, and also by the possibility to interact with other external sensors like those in IoT devices. The camera, the gyroscope, the accelerometer or the GPS are some examples that are only available on mobile devices.

Finally, other reasons for mobile modelling are interoperability with other tools, intuitive handling of smart devices, lightweight (meta-)modelling, scalability of the models or the collaboration, user engagement, and familiarity of the potential users with the usage of the mobile device.

5.3 RQ3: What are the characteristic features of existing mobile modelling approaches?

In this section, we reflect on the classification of approaches presented in Sect. 4 and the coverage of the features in the feature models.

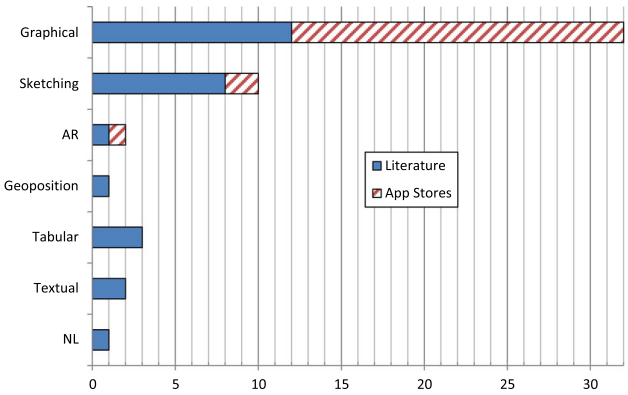


Fig. 20 Concrete syntax styles of mobile modelling approaches and tools

Regarding syntax (cf. Sect. 4.1.1), Fig. 20 shows that the most common style is graphical (88%). Of these graphical approaches, a substantial number is based on sketching (23% of them), and three use geolocation or AR. Other syntaxes—tabular, textual and based on NL—are a minority (with percentages of 6%, 4% and 2%, respectively). Two approaches combine syntaxes: graphical and based on NL (Socio [78]), and graphical/sketching and tabular (MobiDev [89]).

Only 17% of all tools and approaches support extended modelling, allowing users to draw atop the model and annotate its elements. This is always performed on languages with a graphical syntax, and in one case, based on sketching.

Regarding semantics (cf. Sect. 4.1.3), most tools and approaches (75%) do not define an explicit semantics, 6% offer a generative one, and 19% provide execution semantics. Just one tool (DSL-comet [102]) supports access control at the language level for granting different modelling privileges to its users (cf. Sect. 4.1.4).

Most approaches and tools (89%) support one or several fixed languages (cf. Sect. 4.2). Only a minority (13%) can be used to define DSLs, and in all cases, they correspond to academic works.

The architecture of the analysed approaches and tools (cf. Sect. 4.3.1) is divided roughly equally between one-tier (43%) and client-server (47%). In addition to target the deployment on mobile devices, 45% of approaches are also deployed as web or desktop applications, and 11% as both. 34% of the tools are cross-device, and interestingly, most tools in app stores have versions for iOS and Android. Finally, in two cases (HoloFlow [90] and Mind Vector [68]) the deployment is on head-set devices.

Many approaches and tools (72%) lack collaboration support (cf. Sect. 4.3.2). Collaboration is more frequent in academic works, since 44% of the approaches in the literature support collaboration. Most approaches (92%) supporting synchronous collaboration also support some form of asynchronous collaboration. The most frequent synchronous collaboration mechanism is centralized via a server (77%).

Interoperability is poor in general (cf. Sect. 4.3.3). Only 21% of the approaches and tools offer some export capabilities, and 8% have both import and export capabilities. Interoperability support is more frequent in approaches found in the literature (28% of them can export to other format) than in tools from app stores.

Finally, the vast majority of approaches use the keyboard (96%) and the touch screen (94%) for modelling (cf. Sect. 4.3.4). Instead, devices like the camera, mic, accelerometer or GPS are only used by 13% of the approaches, all of them works found in the literature.

5.4 RQ4: What are the gaps in current approaches, and the research opportunities in mobile modelling?

Our mapping study reveals some features that are scarcely represented by existing mobile modelling tools. These provide research opportunities within this field. In the following, we elaborate on such lines of potential research.

5.4.1 Syntax

First, regarding the syntax of modelling languages, most existing approaches support graphical syntaxes, and sometimes sketching, to the detriment of other syntaxes. However, geolocated syntaxes and augmented reality can bring a lot of opportunities for on-site modelling (fourth motivation argued by researchers to use a mobile modelling approach, see Fig. 19). In particular, the ability to overlay virtual objects (with information coming from a model) over physical objects can bring interesting possibilities for modelling [19] in domains like smart factories, construction, IoT—as illustrated by HoloFlow [90]—or interior design (like the IKEA room planner [42]). AR-based syntaxes for DSLs can be an enabling technology for the construction of digital twins [81].

In turn, syntaxes based on voice/text natural language may simplify the user interaction with the mobile in certain scenarios, such as modelling by non-modelling experts, by people with certain kinds of disabilities (e.g. blind people) or for end-user development [78]. These syntaxes may be combined with other natural mobile interaction modes like sketching or touch gestures.

Actually, going one step further, one may devise modelling languages with several syntaxes to give rise to *multi-experience development platforms* (MXDPs) for modelling. MXDPs[6] are a novel concept for IDEs that facilitate creating applications that run on multiple channels (desktop, web, mobile, among others). An MXDP offers an integrated set of front-end development tools and back-end services that make it easy for developers to create fit-for-purpose applications by means of different modalities (e.g. touch, voice, gestures). This way, it provides a consistent user experience from different devices including mobile devices but also the web, wearable devices, conversational agents or augmented reality. We believe that mobile modelling may play a prominent role in such modelling MXDPs.

We have seen that extended modelling features, such as annotations or drawings, are poorly supported in mobile modelling apps. Yet, they allow enhancing the informal semantics of a model and providing indications of future actions to be accomplished. This limitation is not specific to mobile modelling applications, but it has been reported by modelling languages in general. Quoting Störrle [93], when it comes to *"expressing different degrees of reality or epistemic states, such as necessity, contingency, and desirability ... conventional modelling languages are poorly equipped to capture and document this expressive richness"*. Hence, there is a need to investigate and develop effective means to convey informal information and degrees of certainty in (mobile) models.

5.4.2 Semantics

With respect to execution semantics, we have identified several approaches that provide external interaction via web

[6] https://www.gartner.com/en/documents/3987201/magic-quadrant-for-multiexperience-development-platforms.

services or IoT devices. However, exploiting the context to adapt the models, the model editing actions, or the modelling environment is largely unexplored in mobile modelling tools. However, these features may be crucial for approaches aiming at on-site modelling.

5.4.3 Fine-grained access control

Another gap in this area of research is the provision of mechanisms for access control at the model level. We argue that such mechanisms are especially important in collaborative modelling apps to protect the modelled information depending on user capacities or roles. Since collaboration is one of the top motivations argued for mobile modelling (cf. Fig. 19), access control in connection with language elements becomes relevant. In this respect, solutions to provide role-based access control for desktop or web modelling applications [14,94] may need to be transferred or adapted to work on mobile devices. The vision paper [20] presents some steps in this direction.

5.4.4 Language definition

Related to language definition capabilities, the support for meta-modelling and the creation of domain-specific languages is worth exploring. Even though not every user will be willing or able to define a new language, this does not preclude that such a functionality would be useful to more advanced users. Meta-modelling support would allow such users to create fit-for-purpose languages instead of using UML, which demands a software engineering background and might be difficult to understand by novices.

Current meta-modelling processes tend to be heavy-weight, where meta-models are built up-front, and sometimes involving code generation. However, language definition can be much more agile, as demonstrated for example by the FlexiSketch tool [106] or by *example-based* meta-modelling approaches [55,88]. In those approaches, models are built first, and then relevant concepts lifted to the meta-model level, in an automated/assisted way.

As mentioned in Sect. 4.2, truly mobile meta-modelling tools would require either from an internal meta-modelling approach (where meta-modelling occurs in the mobile), or an interpreted external approach (where meta-modelling occurs externally, but the artefacts produced are interpreted by the mobile tool). Instead, a generation-based approach cannot be fully automated, since it requires packaging the generated tool, uploading it to an app store, and waiting for a review process by the app store personnel, before the tool can be distributed and installed.

Regarding the support for fixed languages, in the software design domain, those approaches and tools targeting UML tend to support mostly class diagrams, so better coverage of other diagrams is currently lacking. This situation could be improved by better meta-modelling support.

5.4.5 Deployment

With respect to deployment, most of the tools designed to be deployed on mobile devices target smartphones, and only two are specific to smart glasses and other head-mounted devices. Although few people have access to these technologies nowadays, it remains an entire area of research within the modelling community with many potential and interesting applications, e.g. for on-site modelling and education.

5.4.6 Collaboration

Collaboration is available in about one-third of the mobile modelling tools. In many cases, this feature alone serves as the main motivation for the tool (cf. Fig. 19). However, we miss synchronous collaboration approaches among mobile devices (instead of with other fixed devices), as well as further support for remote collaboration. Moreover, collaboration is not exploited by modelling languages with a textual or jigsaw-like concrete syntax, being this an unexplored area of research.

5.4.7 Interoperability

The interoperability feature is missing in most of the tools, especially for the import. This implies that models developed with other tools, e.g. in a desktop computer, have to be redone in the mobile device.

5.4.8 User interaction

Finally, regarding user interaction, mechanisms for their customization are needed (e.g. see [92]). This could improve the flexibility and efficacy of modelling on mobile devices, which may be especially relevant given the large number of gestures supported by touch screens. Most approaches rely on the touch screen and the keyboard, but supporting several ways of interaction (e.g. voice, text, sketching) would help to realize the idea behind MXDPs. Moreover, the use of sensors like the accelerometer, the compass or the GPS would enable the modelling solution to be useful for on-site modelling.

5.5 Summary of findings

We next summarize the main findings of our study.

– **Domains.** We found that mobile modelling has been applied to mind-maps modelling, software design, process modelling, domain-specific modelling, mobile development and IoT, among others. We found mind-map

modelling exclusively in app stores; while approaches for domain-specific modelling, mobile development and IoT were found in academic papers.
- **Motivations.** The most recurrent motivation for mobile modelling is collaboration, followed by pervasive access, and end-user development. Other motivations include on-site modelling and natural interaction (via sketching and touch gestures).
- **Features.** Most tools offer graphical syntax for modelling, sometimes based on sketching. They typically support a fixed set of languages, with many of the tools being also available for the web or desktop computers. Despite the motivation, only some tools support collaboration, and most use the keyboard and the touch screen for user interaction.
- **Gaps and Opportunities.** Most approaches support graphical syntax, but there is scarce support for other more advanced syntaxes, based on augmented reality or natural language. Most approaches do not provide explicit model semantics (via execution or code generation), but there is an opportunity given the rich set of components and sensors the mobile has and can communicate with. Even though argued as the main motivation, many approaches lack collaboration, and therefore (fine-grained) access control mechanisms. There is typically poor interoperability support, and only a few tools offer language definition capabilities.

5.6 Threats to validity

The purpose of our study is to analyse the use of modelling on mobile devices. Hence, the selection of the studied approaches and tools may impact the validity of our review. A possible threat is that we might have missed some papers due to the query used. To mitigate this risk, we tried several versions of the query (as described in Sect. 3.1) and applied it on the 5 most used academic databases (cf. Table 4). Moreover, we conducted a final process of snowballing [104] to consider additional papers not retrieved in the previous phases.

Since the field under study is tool-oriented, some relevant tools might not come from academia, in which case, they may lack an associated paper in the considered databases. To reduce this risk, we also looked into app stores. However, Google Play and App Store lack a filtering mechanism or an advanced search facility to help refining the search queries. Therefore, we may have missed some relevant tool. To tackle this issue, we looked for references to tools in the selected relevant papers, and performed a search in web search engines. In addition, due to the amount of available tools, and to ensure a certain level of quality, we filtered the results based on ratings and downloads [39]. This might have left some relevant tools out. Moreover, in the search string used for the app stores, we tried to use keywords that were in-line with those used for the academic databases. These keywords might be more oriented to software design, and so, we may have missed interesting apps like room planners, minecraft- or sims-like games, fashion design, or vehicle design apps, among others. In any case, we are confident that the panorama obtained is more accurate and complete than if we did not consider tools from app stores.

Some of the tools in our study are web-based (cf. Table 11) but were designed for their use on mobile devices. Many web-based modelling tools exist and many of them can be accessed and used on mobile devices. However, as mentioned in Sect. 2, we excluded them from our study because our focus is on native mobile apps that are able to access the capabilities of the mobile. We believe that web-based modelling is a topic that deserves its own systematic mapping study.

Finally, the misunderstanding of the tool features or the impossibility to try a tool may have introduced mistakes in our classification. To alleviate this issue, we installed and tested the tools whenever possible, but some functionalities required specific context or conditions (IoT devices, other devices for initiating collaboration, payment for unlocking some functionalities). On the academia side, some tools were not available anymore, and in those cases, we looked for additional materials (videos, web pages) on the Internet.

6 Conclusions

In this paper, we have presented a systematic mapping study about modelling on mobile devices. To provide a comprehensive view of the current state of this field, we analysed both approaches from academia and tools from app stores. To classify the existing proposals, we created a feature diagram around three main dimensions: the supported modelling language, its definition and the tool support.

We conducted this study to answer four research questions. In the first two, we wanted to investigate the domains where mobile modelling has been applied, and the reasons to adopt it. We found that mobile modelling has been applied to mind-maps, software design, process modelling, domain-specific modelling, mobile development and IoT, among others; and that collaboration, pervasive access, end-user development, on-site modelling, and natural interaction via sketching and touch gestures are recurrent reasons for mobile modelling approaches. Third, we wondered about the features of existing mobile modelling tools. We found that most tools support a fixed set of graphical languages, sometimes based on sketching; with many of the tools being also available for the web or desktop computers; some supporting collaboration; and most using the keyboard and the touch screen for user interaction. Fourth, we aimed at identifying gaps in the current approaches and research opportunities in

the area. We pointed out that concrete syntaxes using AR, geolocation or voice are scarce; there are almost no tools for head-set devices; and there are research opportunities on high-level features of mobile modelling tools such as interoperability, external interaction, extended modelling, context, meta-modelling, sensor-based interaction and user interaction customizability.

This paper is an invitation to the MDE community to tackle the existing challenges relative to mobile modelling and to improve the current MDE tools available for mobile devices.

Acknowledgements We thank the reviewers for their useful comments. This work has been funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement n° 813884 (Lowcomote [96]), by the Spanish Ministry of Science (project MASSIVE, RTI2018-095255-B-I00), and by the R&D programme of Madrid (project FORTE, P2018/TCS-4314).

References

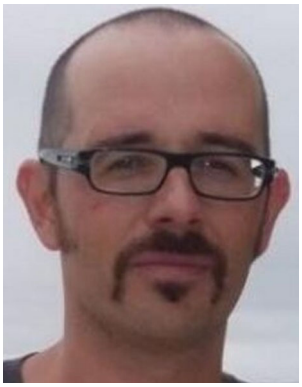
1. Aleksandr, K.: Pureflow (2020)
2. Alpers, S., Eryilmaz, E., Hellfeld, S., Oberweis, A.: Mobile modeling tool based on the horus method. In: 2014 International Workshop on Advanced Information Systems for Enterprises (IWAISE), pp. 65–71 (2014)
3. Antunes, P., Herskovic, V., Ochoa, S.F., Pino, J.A.: Reviewing the quality of awareness support in collaborative applications. J. Syst. Softw. **89**, 146–169 (2014)
4. ARCore: https://developers.google.com/ar (2020)
5. ARKit: https://developer.apple.com/augmented-reality/ (2020)
6. Astah* UML Pad: https://astah.net/products/astah-uml-pad/ (2020)
7. Avgoustinov, N.: Modelling in Mechanical Engineering and Mechatronics. Springer-Verlag, Berlin (2007)
8. Azuma, R.T.: A survey of augmented reality. Presence Teleoper. Virtual Environ. **6**(4), 355–385 (1997)
9. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
10. Baloian, N., Zurita, G., Santoro, F.M., Araujo, R.M., Wolfgan, S., Machado, D., Pino, J.A.: A collaborative mobile approach for business process elicitation. In: 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 473–480 (2011)
11. Barricelli, B.R., Cassano, F., Fogli, D., Piccinno, A.: End-user development, end-user programming and end-user software engineering: A systematic mapping study. J. Syst. Softw. **149**, 101–137 (2019)
12. Becker, J., Clever, N., Holler, J., Shitkova, M.: Icebricks - mobile application for business process modeling. In: 10th International Conference on New Horizons in Design Science: Broadening the Research Agenda (DESRIST), *LNCS*, vol. 9073, pp. 361–365. Springer, Berlin (2015)
13. Bencomo, N., Götz, S., Song, H.: Models@run.time: a guided tour of the state of the art and research challenges. Softw. Syst. Model. **18**(5), 3049–3082 (2019)
14. Bergmann, G., Debreceni, C., Ráth, I., Varró, D.: Towards efficient evaluation of rule-based permissions for fine-grained access control in collaborative modeling. In: 2nd International Workshop on Collaborative Modelling in MDE (COMMitMDE@MODELS), *CEUR Workshop Proceedings*, vol. 2019, pp. 135–144. CEUR-WS.org (2017)
15. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
16. Blair, G.S., Bencomo, N., France, R.B.: Models@run.time. IEEE Comput. **42**(10), 22–27 (2009)
17. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, Second Edition. Synthesis Lectures on Software Engineering. Morgan and Claypool Publishers (2017)
18. Brooks, F.P.: No silver bullet - essence and accidents of software engineering. Computer **20**(4), 10–19 (1987)
19. Brunschwig, L., Campos-López, R., Guerra, E., de Lara, J.: Towards domain-specific modelling environments based on augmented reality. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), pp. 56–60 (2021)
20. Brunschwig, L., Guerra, E., de Lara, J.: Towards access control for collaborative modelling apps. In: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS Companion), pp. 67:1–67:10. ACM (2020)
21. Buchmann, T.: Valkyrie: A uml-based model-driven environment for model-driven software engineering. In: 7th International Conference on Software Paradigms Trends (ICSOFT), pp. 147–157. SciTePress (2012)
22. Buchmann, T., Pezoldt, P.: A lightweight framework for graphical editors on android devices. In: 9th International Conference on Software Engineering and Applications (ICSOFT-EA), pp. 81–89 (2014)
23. Budinsky, F., Brodsky, S.A., Merks, E.: Eclipse Modeling Framework. Pearson Education (2003)
24. Corley, J., Syriani, E., Ergin, H.: Evaluating the cloud architecture of atompm. In: 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 339–346. SciTePress (2016)
25. Cribster: Json designer (2020)
26. Danado, J., Paternò, F.: Puzzle: A mobile application development environment using a jigsaw metaphor. J. Vis. Lang. Comput. **25**(4), 297–315 (2014)
27. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW), pp. 107–114. ACM (1992)
28. Döweling, S., Tahiri, T., Schmidt, B., Nolte, A., Khalilbeigi, M.: Collaborative business process modeling on interactive tabletops. In: 21st European Conference on Information Systems (ECIS), p. 29 (2013)
29. DrawExpress: https://drawexpress.com (2020)
30. Flowdia Diagrams: https://www.bezapps.com (2020)
31. Francese, R., Risi, M., Tortora, G.: Iconic languages: Towards end-user programming of mobile applications. J. Vis. Lang. Comput. **38**, 1–8 (2017)
32. Franzago, M., Ruscio, D.D., Malavolta, I., Muccini, H.: Collaborative model-driven software engineering: a classification framework and a research map. IEEE Trans. Software Eng. **44**(12), 1146–1175 (2018)
33. Gaouar, L., Benamar, A., Bendimerad, F.: Model driven approaches to cross platform mobile development. In: International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC), pp. 1–5 (2015)
34. GraphML: http://graphml.graphdrawing.org (2020)
35. Grundy, J.C., Hosking, J., Li, K.N., Ali, N.M., Huh, J., Li, R.L.: Generating domain-specific visual language tools from abstract visual specifications. IEEE Trans. Software Eng. **39**(4), 487–515 (2013)

36. Grundy, J.C., Hosking, J.G., Cao, S., Zhao, D., Zhu, N., Tempero, E.D., Stoeckle, H.: Experiences developing architectures for realizing thin-client diagram editing tools. Softw. Pract. Exp. **37**(12), 1245–1283 (2007)
37. Guerra, E., de Lara, J.: On the quest for flexible modelling. In: Proc. MODELS, pp. 23–33. ACM (2018)
38. Halna Mind: https://sites.google.com/site/halnablue/ (2020)
39. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: Msr for app stores. In: Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, MSR '12, pp. 108–111. IEEE Press (2012)
40. HoloLens: https://www.microsoft.com/en-us/hololens (2020)
41. Hutchinson, J.E., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Sci. Comput. Program. **89**, 144–161 (2014)
42. IKEA room planner. https://apps.apple.com/us/app/ikea-place/id1279244498 (2021)
43. InfoRapid: Knowledgebase builder (2020)
44. Inspiration Maps: https://www.inspiration-at.com/inspiration-maps/ (2020)
45. Jézéquel, J., Combemale, B., Barais, O., Monperrus, M., Fouquet, F.: Mashup of metalanguages and its implementation in the kermeta language workbench. Softw. Syst. Model. **14**(2), 905–920 (2015)
46. Kammerer, K., Kolb, J., Ronis, S., Reichert, M.: Collaborative process modeling with tablets and touch tables - a controlled experiment. In: 9th IEEE International Conference on Research Challenges in Information Science (RCIS), pp. 31–41 (2015)
47. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-oriented domain analysis (FODA) feasibility study. Tech. Rep. CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (1990)
48. Kelly, S., Lyytinen, K., Rossi, M.: Metaedit+ a fully configurable multi-user and multi-tool case and came environment. In: 8th International Conference on Advanced Information Systems Engineering (CAISE), pp. 1–21. Springer Berlin Heidelberg (1996)
49. Kelly, S., Tolvanen, J.: Domain-Specific Modeling - Enabling Full Code Generation. Wiley, New Jersey (2008)
50. klim: Database designer (2020)
51. Korzetz, M., Kühn, R., Kegel, K., Georgi, L., Schumann, F., Schlegel, T.: Milkyway: A toolbox for prototyping collaborative mobile-based interaction techniques. In: 13th International Conference on Universal Access in Human-Computer Interaction. Multimodality and Assistive Environments, UAHCI 2019, *Lecture Notes in Computer Science*, vol. 11573, pp. 477–490. Springer, Berlin (2019)
52. Lekh Diagram: https://www.lekhapp.com (2020)
53. Lemma, R., Lanza, M., Mocci, A.: Cel: Touching software modeling in essence. In: 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 439–448 (2015)
54. Li, T.J., Li, Y., Chen, F., Myers, B.A.: Programming IoT devices by demonstration using mobile apps. In: 6th International Symposium on End-User Development (IS-EUD), *LNCS*, vol. 10303, pp. 3–17. Springer, Berlin (2017)
55. López-Fernández, J.J., Garmendia, A., Guerra, E., de Lara, J.: An example is worth a thousand words: Creating graphical modelling environments by example. Softw. Syst. Model. **18**(2), 961–993 (2019)
56. López-Jaquero, V., Navarro, E., Simarro, F.M., González, P.: Metamodels infrastructure and heuristics for metamodel-driven multi-touch interaction. In: 13th IFIP TC13 International Conference on Human-Computer Interaction (INTERACT), *LNCS*, vol. 8118, pp. 210–227. Springer, Berlin (2013)
57. Lucidchart: https://www.lucidchart.com/ (2020)
58. Ludewig, J.: Models in software engineering. Softw. Syst. Model. **2**(1), 5–14 (2003)
59. Ma, Z., Yeh, C.Y., He, H., Chen, H.: A web based uml modeling tool with touch screens. In: 29th ACM/IEEE International Conference on Automated Software Engineering (ASE), ASE '14, pp. 835–838 (2014)
60. MagicLeap: https://www.magicleap.com (2020)
61. Malan, D., Leitner, H.: Scratch for budding computer scientists. 38th ACM Technical Symposium on Computer Science Education (SIGCSE) **39** (2007)
62. Mangano, N., van der Hoek, A.: A tool for distributed software design collaboration. In: 2012 ACM Conference on Computer Supported Cooperative Work Companion (CSCM), pp. 45–46 (2012)
63. Maróti, M., Kecskés, T., Kereskényi, R., Broll, B., Völgyesi, P., Jurácz, L., Levendovszky, T., Lédeczi, Á.: Next generation (meta)modeling: Web- and cloud-based collaborative tool infrastructure. In: 8th Workshop on Multi-Paradigm Modeling (MPM@MODELS), *CEUR Workshop Proceedings*, vol. 1237, pp. 41–60. CEUR-WS.org (2014)
64. Masson, C., Corley, J., Syriani, E.: Feature model for collaborative modeling environments. In: Proceedings of MODELS 2017 Satellite Events, *CEUR Workshop Proceedings*, vol. 2019, pp. 164–173. CEUR-WS.org (2017)
65. McDirmid, S.: Coding at the speed of touch. In: 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!), pp. 61–76. ACM (2011)
66. miMind: http://mimind.cryptobees.com (2020)
67. Mind Meister: https://www.mindmeister.com/ (2020)
68. Mind Vector: http://www.mindvectorweb.com (2020)
69. Mindly: https://www.mindlyapp.com (2020)
70. MindMapping 3D: https://www.scapehop.com (2020)
71. MindMaster: https://www.edrawsoft.com/mindmaster/ (2020)
72. MindNode: https://mindnode.com (2020)
73. Mindomo: https://www.mindomo.com/ (2020)
74. Müller, T., Müller, H.: Modelling in Natural Sciences: Design, Validation and Case Studies. Springer-Verlag, Berlin (2003)
75. Nolte, A., Brown, R., Anslow, C., Wiechers, M., Polyvyanyy, A., Herrmann, T.: Collaborative business process modeling in multi-surface environments. In: Collaboration Meets Interactive Spaces, pp. 259–286. Springer, Berlin (2016)
76. Oppl, S., Stary, C.: Effects of a tabletop interface on the co-construction of concept maps. In: 13th IFIP TC13 International Conference on Human-Computer Interaction (INTERACT), *LNCS*, vol. 6948, pp. 443–460. Springer, Berlin (2011)
77. OrgChart: https://orgchartgo.com (2020)
78. Pérez-Soler, S., Guerra, E., de Lara, J.: Collaborative modeling and group decision making using chatbots in social networks. IEEE Softw. **35**(6), 48–54 (2018)
79. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 68–77. BCS Learning and Development Ltd. (2008)
80. User Interface Description Language. http://www.uidl.net (2011). Last accessed, January 2021
81. Rabah, S., Assila, A., Khouri, E., Maier, F., Ababsa, F., bourny, V., Maier, P., Mérienne, F.: Towards improving the future of manufacturing through digital twin and augmented reality technologies. Procedia Manufacturing **17**, 460–467 (2018). 28th International Conference on Flexible Automation and Intelligent Manufacturing
82. Rachad, T., Idri, A.: Intelligent mobile applications: A systematic mapping study. Mob. Inf. Syst. **2020**, 6715363:1–6715363:17 (2020)

83. Ribeiro, A., Silva, A.: Survey on cross-platforms and languages for mobile apps. In: 8th International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 255–260 (2012)
84. Ritter, C.T., Schwaiger, J., Johannsen, F.: A prototype for supporting novices in collaborative business process modeling using a tablet device. In: 10th International Conference on New Horizons in Design Science: Broadening the Research Agenda (DESRIST), *LNCS*, vol. 9073, pp. 371–375. Springer, Berlin (2015)
85. de Sá, M., Carriço, L.: A mobile tool for in-situ prototyping. In: 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) (2009)
86. Sahay, A., Indamutsa, A., Ruscio, D.D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. In: 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 171–178. IEEE (2020)
87. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006)
88. Sebastián-Lombraña, A., Guerra, E., de Lara, J.: Positioning-based domain-specific modelling through mobile devices. In: 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 150–157. IEEE (2020)
89. Seifert, J., Pfleging, B., del Carmen Valderrama Bahamóndez, E., Hermes, M., Rukzio, E., Schmidt, A.: Mobidev: a tool for creating apps on mobile phones. In: 13th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI), pp. 109–112. ACM (2011)
90. Seiger, R., Gohlke, M., Aßmann, U.: Augmented reality-based process modelling for the internet of things with holoflows. In: 20th International Conference on Enterprise, Business-Process and Information Systems Modeling (BPMDS/EMMSAD@CAiSE), *LNBIP*, vol. 352, pp. 115–129. Springer, Berlin (2019)
91. SimpleMind: https://simplemind.eu (2020)
92. Sousa, V., Syriani, E., Fall, K.: Operationalizing the integration of user interaction specifications in the synthesis of modeling editors. In: 12th ACM SIGPLAN International Conference on Software Language Engineering (SLE), pp. 42–54. ACM (2019)
93. Störrle, H.: Modeling moods. In: 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS Companion), pp. 468–477. IEEE (2019)
94. https://www.eclipse.org/cdo/ . Last accessed, January 2021
95. Tillmann, N., Moskal, M., de Halleux, J., Fähndrich, M., Burckhardt, S.: Touchdevelop: app development on mobile devices. In: 20th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-20), p. 39. ACM (2012)
96. Tisi, M., Mottu, J., Kolovos, D.S., de Lara, J., Guerra, E., Ruscio, D.D., Pierantonio, A., Wimmer, M.: Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. In: STAF (Co-Located Events), *CEUR Workshop Proceedings*, vol. 2405, pp. 73–78. CEUR-WS.org (2019)
97. Turner, A.: How many smartphones are in the world? https://www.bankmycell.com/blog/how-many-phones-are-in-the-world (2020)
98. UML 2.5.1 OMG specification. http://www.omg.org/spec/UML/2.5.1/ (2017)
99. Urrego, J.S., Muñoz, R., Mercado, M., Correal, D.: Archinotes: A global agile architecture design approach. In: 15th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP), *LNBIP*, vol. 179, pp. 302–311. Springer, Berlin (2014)
100. Vangheluwe, H., de Lara, J., Mosterman, P.: An introduction to multi-paradigm modelling and simulation. In: Proceedings of AI, Simulation and Planning in High Autonomy Systems, pp. 9–20 (2002)
101. Vaquero-Melchor, D., Garmendia, A., Guerra, E., de Lara, J.: Domain-specific modelling using mobile devices. In: 12th International Conference on Software Technologies (ICSOFT), Revised Selected Papers, *CCIS*, vol. 743, pp. 221–238. Springer, Berl (2017)
102. Vaquero-Melchor, D., Palomares, J., Guerra, E., de Lara, J.: Active domain-specific languages: Making every mobile user a modeller. In: ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 75–82. IEEE Comp. Soc. (2017)
103. Voelter, M.: DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. dslbook.org (2013). URL http://www.dslbook.org
104. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: 18th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 38:1–38:10. ACM (2014)
105. Wu, Y.: Global smartphone user penetration forecast by 88 countries: 2007-2022. Wireless Smartphone Strategies Services (2016)
106. Wüest, D., Seyff, N., Glinz, M.: Flexisketch: a lightweight sketching and metamodeling approach for end-users. Softw. Syst. Model. **18**(2), 1513–1541 (2019)
107. OMG's XML Metadata Interchange. https://www.omg.org/spec/XMI/About-XMI/ (2015). Last accessed, January 2021
108. XMind: https://www.xmind.net/ (2020)
109. The XML Process Definition Language (XPDL) by the Workflow Management Coalition. https://www.wfmc.org/standards/xpdl (2012). Last accessed, January 2021
110. Zarwin, Z., Bjekovic, M., Favre, J., Sottet, J., Proper, H.A.: Natural modelling. J. Object Technol. **13**(3), 1–36 (2014)
111. Zein, S., Salleh, N., Grundy, J.: A systematic mapping study of mobile application testing techniques. J. Syst. Softw. **117**, 334–356 (2016)
112. Zhao, D., Grundy, J.C., Hosking, J.G.: Generating mobile device user interfaces for diagram-based modelling tools. In: 7th Australasian User Interface Conference (AUIC), *CRPIT*, vol. 50, pp. 101–108. Australian Computer Society (2006)
113. Zhu, N., Grundy, J.C., Hosking, J.G., Liu, N., Cao, S., Mehra, A.: Pounamu: a meta-tool for exploratory domain-specific visual language tool development. J. Syst. Softw. **80**(8), 1390–1407 (2007)

Léa Brunschwig is a PhD student at the Universidad Autónoma de Madrid since 2019. She is also an early stage researcher of the Lowcomote European project which aims at studying low-code development platforms and environments. Her research has a focus on model-driven engineering and mobile development.

Esther Guerra is Associate Professor at the Computer Science department of the Universidad Autónoma de Madrid. Together with J. de Lara, she leads the modelling and software engineering research group (http://miso.es). She is interested in flexible modelling, meta-modelling, domain specific languages and model transformation. Contact her at esther.guerra@uam.es, or visit http://www.eps.uam.es/~eguerra.



Juan de Lara is Full professor at the computer science department of the Universidad Autónoma de Madrid. Together with E. Guerra, he leads the modelling and software engineering research group. His research interests are in model-driven engineering and automated software development. Contact him at juan.delara@uam.es, or visit http://arantxa.ii.uam.es/~jlara/.