

# Location and Object-Based Mobile Applications

## Development Based on Use Case Templates and Visual Programming

Martin Zimmermann

Department of Economics

Offenburg University

Offenburg, Germany

email: m.zimmermann@hs-offenburg.de

**Abstract** - The main advantage of mobile context-aware applications is to provide effective and tailored services by considering the environmental context, such as location, time, nearby objects and other data, and adapting their functionality according to the changing situations in the context information without explicit user interaction. The idea behind Location-Based Services (LBS) and Object-Based Services (OBS) is to offer fully-customizable services for user needs according to the location or the objects in a mobile user's vicinity. However, developing mobile context-aware software applications is considered as one of the most challenging application domains due to the built-in sensors as part of a mobile device. Visual Programming Languages (VPL) and hybrid visual programming languages are considered to be innovative approaches to address the inherent complexity of developing programs. The key contribution of our new development approach for location and object-based mobile applications is a use case driven development approach based on use case templates and visual code templates to enable even programming beginners to create context-aware mobile applications. An example of the use of the development approach is presented and open research challenges and perspectives for further development of our approach are formulated.

**Keywords** - Location-Based Services; Object-Based Services; Mobile Applications; Visual Programming.

### I. INTRODUCTION

Sensors enable the creation of context-aware mobile applications in which applications can discover and take advantage of contextual information, such as user location, nearby people and objects. As a consequence, context-aware mobile applications can sense clues about the situational environment making mobile devices more intelligent, adaptive, and personalized.

Context has been defined as any required knowledge to identify the current situation of a person or object in order to provide tailor-made services. The situational environment of a mobile user becomes more vital in mobile applications where the context, e.g., geo position of a user can change rapidly. For example, depending on its current location, a tourist would like to see relevant tourist attractions on a map

together with distance information. Mobile applications can obtain the context information in various ways in order to provide more adaptable, flexible and user-friendly services.

A combination of context-aware applications and mobile devices provides a novel opportunity for both end users and application developers to obtain context and the consequent response to any changes in the context. Hence, the main advantage of mobile context-aware applications is to provide tailored services by considering the environmental context, such as location, time, weather conditions, nearby objects, and adapting their functionality according to the changing situations in the context data without explicit user interaction.

A general definition of context was given by Dey and Abowd [1]: "Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Categories of context information that are practically significant include [1]:

- **Environmental Context:** Includes all the surrounding environmental conditions of current location (like air quality, temperature, humidity, noise level and light condition).
- **Activity Context:** Defines the user's current activity including private and professional activities that can be sensed like talking, reading, walking, and running.
- **Temporal Context:** Consists of temporal factors, such as current time, date, and season of the year.
- **Personal (identity) Context:** Specifies user's characteristics and preferences like name, age, sex, contact number, user's hobbies and interests.
- **Spatial Context:** Involves any information regarding the position of an entity (person and object), for instance orientation, location, acceleration, speed.
- **Vital Signs Context:** Covers all information related to health state, such as heart rate, blood pressure, voice tone, and muscle activity.

The combination of spatial, temporal, activity and personal contexts makes the primary context to understand

the current situation of entities. These types of contexts can respond to basic questions about when, where, what, and who. The idea behind Location-Based Services (LBS) and Object-Based Services (OBS) as subcategories of context-aware mobile applications is to tailor services according to the location of a user or the objects in a mobile user's vicinity.

In [2], various Model Driven Development (MDD) techniques and methodologies are systematically investigated, i.e., what MDD techniques and methodologies have been used to support mobile app development and how these techniques have been employed, to identify key benefits, limitations, gaps and future research potential. Our approach based on case templates and visual programming is tailored to the needs of mobile programming beginners.

Visual Programming Languages (VPL) and hybrid visual programming languages are considered to be innovative approaches to address the inherent complexity of developing programs [3], [4]. In this work, we introduce an in-depth discussion of a new VPL based method, to enable even programming beginners to create context-aware mobile applications.

The rest of the paper is organized as follows: Section 2 introduces LBS and OBS concepts, especially various technologies to determine the object context of a user. Our proposed development process in terms of use case templates and visual code templates is described in Section 3. Visual programming concepts and the development environment which we use in our projects are introduced in Section 4. Sections 3 and 4 also describe how our approach is applied to an LBS and OBS example in the field of tourism. Finally, the limitations of the VPL approach as well as directions for future research are presented in Section 5.

## II. LOCATION AND OBJECT BASED SERVICES

LBS can be described as applications that are dependent on a certain location. Two broad categories of LBS can be defined as triggered and user-requested [5]. In a user-requested scenario, the user is retrieving the position once and uses it on subsequent requests for location-dependent information. This type of service usually involves either personal location, i.e., finding where you are or services location, e.g., where is the nearest hospital. Examples of this type of LBS are also navigation (usually involving a map) and direction (routing information). A triggered LBS by contrast relies on a condition set up in advance that, once fulfilled, retrieves the position of a given device. An example is in emergency services, where the call to the emergency center triggers an automatic location request from the mobile network.




The idea behind OBS is to tailor a service according to the objects in a mobile user's vicinity. For example, a visitor standing in front of a painting in an art gallery should be provided with additional information about the painting, such as the artist, or even the opportunity to order a print (one button pay). Popular technologies for determining the object context are Quick Response (QR) codes, Near-Field Communication (NFC) tags and beacons.

LBS are typically based on GPS. The position of a person or an object is determined in terms of latitude and longitude. To determine the object context, i.e., the object(s) at which the user is located, various technologies can be used (Table I). An object, e.g., painting in a museum can be provided with one or more of the following elements: Bluetooth Low Energy (BLE) Beacon, NFC tag, and QR code.

Beacons are small wireless, usually battery-powered devices that transmit data at regular intervals using BLE [6], [7]. This mini-radio transmission devices can be 'discovered' and seen by all BLE scanners, e.g., a smartphone within a certain radius. However, beacons do not work by themselves: they require a mobile app. So, in case of an arts gallery or museum, the visitor must have downloaded the mobile application beforehand for the beacon to work.

NFC is a short-range wireless connectivity technology that uses magnetic field induction to enable communication between devices when they're touched together or brought within a few centimeters of each other [8]. NFC builds on the work of the Radio-Frequency Identification (RFID) set of standards and specifications, such as ISO/IEC 14443 and ISO/IEC 15963. By passing a mobile device near an NFC chip, one can read the data it contains and interact with the content. Advantages and disadvantages from a user's point of view of the different technologies for implementing OBS are shown in Table I.

TABLE I. TECHNOLOGIES FOR OBJECT BASED SERVICES.

<i>Technology</i>	<i>Pros</i>	<i>Cons</i>
<b>Beacon</b> 	+ no user interactions required	- requires power supply - more expensive (compared with NFC, QR codes)
<b>NFC tag</b> 	+ can store the most data + cheap	- user must tap on the item
<b>QR Code</b> 	+ well-known technology	- requires most user interactions (open camera app, scan image)

QR codes are a type of matrix bar code that was invented by Denso Wave in 1994 to be used as labels on automotive parts [9]. It allows to store large amount of data (compared to 1D barcodes) and a high-speed decoding process using any handheld device like phones. The popularity of QR code grows rapidly with the growth of mobile users and thus the QR code concept is rapidly arriving at high levels of acceptance worldwide.

### III. DEVELOPMENT PROCESS

We propose five steps to perform requirements engineering for location and object-based mobile applications (Figure 1):

- Selection and instantiation of use case templates (input, output, steps)
- Development of the user interface for each use case, e.g., triggering a use case with a button
- Selection and instantiation of visual code templates
- Selection of additional non-visible components, e.g., location sensor, QR code scanner, etc.
- Event based programming (calling methods related to the user interface and the non-visible components).

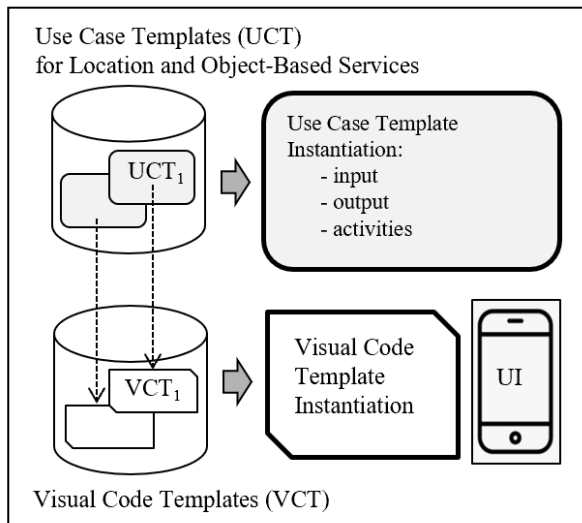


Figure 1. Use Case Templates and Visual Program Code Templates.

Use case templates in Figure 1 are use case specifications that can operate with generic objects and activities. This allows us to create a use case description whose functionality can be adapted to more than one concrete location-based scenarios without repeating the entire description for each scenario. Visual code templates are implementations of use case templates based on visual programming (Section 4). Each use case template is associated with a visual code template.

The use case template in Table II describes an LBS scenario, a search object pattern in terms of input, output and steps to be executed. For example, the template can be applied to visualize some tourist attractions and the current position of a mobile user on a map. Filters are used to display certain tourist attractions, e.g., museums.

Table III illustrates a use case template for an OBS. In this case QR codes are used to identify an object. We defined similar use case templates for OBS using Beacons and NFC tags.

TABLE II. USE CASE TEMPLATE “SEARCH AND SHOW OBJECT(S) ON A MAP”.

<i>Use Case template</i>	<i>Search and show &lt;object(s)&gt; on a map</i>
<b>Input</b>	Name/Id/Category of an <object>
<b>Steps</b>	1: Determine the current geo position of the user 2: Show a map with the user's current position as the center point 3: Search the <object(s)> according to the name/id/category (in a list of <object>) if found → Create marker(s) for the <object(s)>
<b>Output</b>	Map with markers: → marker for the current position of the user → marker(s) representing the <object(s)>

The generic part in the use case template in Table III is represented by the activities “Visualize <object property i>”. The concrete visualization is dependent on the scenario to be implemented. E.g., for exhibits, like paintings in a gallery equipped with QR codes, object visualization could mean to represent a link to a video (painter explaining interesting background information) or a link to allow a tourist to buy a print.

TABLE III. USE CASE TEMPLATE “SEARCH AND SHOW OBJECT(S)”.

<i>Use Case Template</i>	<i>Visualize &lt;object&gt; properties</i>
<b>Input</b>	QR Code
<b>Steps</b>	1: Scan QR code 2: Search the QR Code in a list of codes if found → Visualize <object property 1> Visualize <object property 2> ...
<b>Output</b>	Visualized object properties

In the following sections, we describe how our approach is applied to an LBS and OBS example in the field of tourism.

### IV. VISUAL PROGRAMMING

Visual programming environments are increasingly used in demanding problem domains, e.g., Internet of Things (IoT) applications [10] or robot applications [11]. For example, Pepper’s popular programming interface is based on visual elements for built-in sensors and actuators [12].

Basic functions of the Pepper platform are provided per sensor and actuator, e.g., open/close hand functions for the actuator hand or detect touch on hand tactile sensor. Flow elements are connected with each other to form business workflows similarly to the visual building blocks of Business Process Model and Notation (BPMN) [13]. BPMN is a graphical representation for specifying business processes in a business process model based on a flowcharting technique very similar to activity diagrams from Unified Modeling Language (UML). BPMN's basic element categories are flow objects (events, activities, gateways), connecting objects (sequence flow, message flow, association), and artifacts (data object, group, annotation).

### A. Development Environment

We use MIT App Inventor [14] and Thinkable [15], which are both cloud-based visual programming development environments for mobile applications (Android and iOS). The basic concepts are components, events and functions. App Inventor and Thinkable provide the application developer with many different components to use while building a mobile app. Components are chosen on the “Design Screen” and dragged onto the phone (Figure 2).

The properties of these components, such as color, font, speed, etc. can then be changed by the developer. Available component categories are user interface elements, media, storage, location-based services etc. Components can be clicked on and dragged onto the development screen area.

There are two main types of components: visible and non-visible. Visible components, such as buttons, text boxes, labels, etc. are part of the user interface whereas non-visible components, such as the location sensor, QR Code scanner, sound, orientation sensor are not seen and thus not a part of the user interface screen, but they provide access to built-in functions of the mobile device (Figure 2).

Components are based on an object-oriented paradigm, i.e., decomposition of a system (an app) into a number of entities called objects and then ties properties and function to these objects. An object’s properties can be accessed only by the functions associated with that object but functions of one object can access the function of other objects in the same cases using access specifiers.

Event handler blocks specify how a program should respond to certain events. After, before, or when the event happens can all call different event handlers. There are two types of events: user-initiated and automatic.

Clicking a button, touching a map, and tilting the phone are user-initiated events. Sprites colliding with each other or with canvas edges are automatic events. Timer events are another type of automatic event. Sensor events function also as user-initiated events. For example, the orientation sensor, the accelerometer, and the location sensor all have events that get called when the user moves the phone in a certain way or to a certain place.

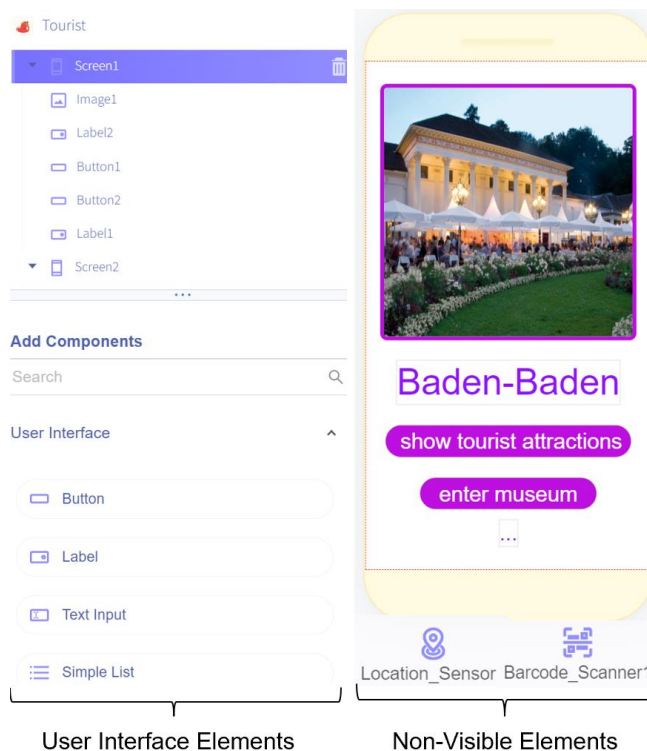


Figure 2. Development Environment.

Figure 3 shows the visual elements of the event-based programming part for a simple LBS.

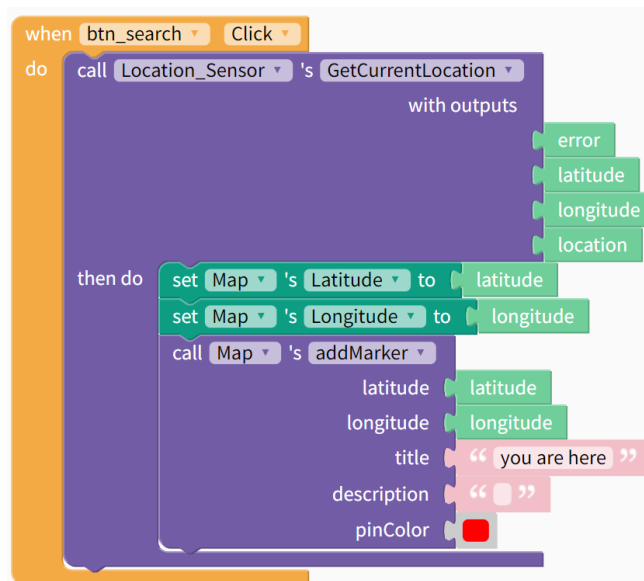


Figure 3. Visual Elements of a Simple LBS.

Objects, method calls, arguments and results of method calls are represented by visual elements with different shapes and colors. In the example in Figure 3, first the current position of a user is determined by calling the method

GetCurrentLocation(). The resulting values (latitude and longitude) are used in the next step for the specification of the map center (two set operations). By calling the method addMarker, a marker is created in a third step. The arguments for the last method call are again visual elements (previously calculated values for the current latitude and longitude of the user).

### B. Visual Code Templates

Each use case template is associated with a visual code template. Figure 4 illustrates the visual code template for the use case template “visualize <object> properties” in Table III. First, the event handler calls the scan method. The resulting id is used to search for the corresponding object in an object list. The instantiation of the template involves

- creation of an object list
- visualization of the object properties.

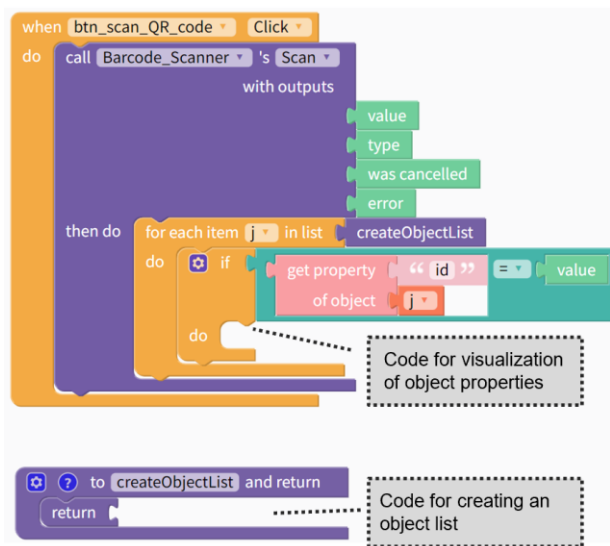


Figure 4. Example Visual Code Template.

The creation of an object list could be based on a local list (as part of a mobile app) or a cloud-based object list. The creation / access to an object list is encapsulated in the function createObjectList(). Finally, the object properties have to be visualized, which is not part of the template, because the concrete visualization are dependent on the scenario to be implemented. E.g., for exhibits, like paintings in a gallery equipped with QR codes, object visualization could mean to create a link to a video (painter explaining interesting background information) or a link to allow a tourist to buy a print.

### V. CONCLUSION

The main advantage of mobile context-aware applications is to provide an effective, usable, rapid service by considering the environmental context, such as location, time, nearby objects, and adapting their functionality

according to the changing situations in context data. LBS and OBS represent two main categories of context-aware applications. Use case templates and visual code templates are particularly well suited for programming beginners.

Meanwhile, visual programming environments are increasingly used in demanding problem domains, e.g., IoT applications [10]. The development of use cases templates (in the sense of requirements engineering) as the starting point of an app project has proven to be very advantageous. Representing programming language concepts by using visual elements with different shapes and colors fits well with the object-oriented approach.

A main drawback of the used programming environments is the identification and handling of runtime errors due to the lack of integrated debugging functions. However, our use case centered approach leads normally to manageable runtime error because each use case is developed and tested as a separate unit.

Future work will focus on the development of patterns, which are a well-known concept in the traditional software engineering. An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture. Patterns become reusable solutions for a common set of problems in software development, addressing issues like high availability, performance, and risk minimization. Additionally, we are going to implement additional components (called extensions), e.g., an NFC component offering more powerful and flexible functions and events. Extension components can be used in building projects, just like other built-in components. The difference is that extension components can be distributed on the Web and loaded into the development environment dynamically.

### REFERENCES

- [1] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Contextawareness," CHI 2000 Workshop on The What, Who, Where, When, Why and How of Context-awareness, pp. 1–6, 2000.
- [2] Md. Shamsujjoha, J. Grundy, L. Li, H. Khalajzadeh, and Q. Lu, "Developing Mobile Applications Via Model Driven Development: A Systematic Literature Review", Information and Software Technology, vol. 140, December 2021.
- [3] M. Idrees and F. Aslam, "A Comprehensive Survey and Analysis of Diverse Visual Programming Languages," VFAST Transactions on Software Engineering, vol.10, no. 2, pp. 47–60, 2022.
- [4] R. Daskalov, G. Pashev, and S. Gaftandzhieva, "Hybrid Visual Programming Language Environment for Programming Training," TEM Journal, vol. 10, issue 2, pp. 981–986, 2021.
- [5] F. F. Chamasemani and L. S. Affendey, "Impact of mobile context-aware applications on human computer interaction," Journal of Theoretical and Applied Information Technology, vol. 62, no.1, pp. 281–287, 2014.
- [6] T D’Roza and G Bilchev, "An overview of location-based services," BT Technology Journal, vol. 21, no. 1, pp. 20–27, 2003.
- [7] R. Faragher and R. Harle, "Location Fingerprinting With Bluetooth Low Energy Beacons," in IEEE Journal on Selected Areas in Communications, vol. 33, no. 11, pp. 2418–2428, 2015.

- [8] V. Coskun, B. Ozdenizci, and K. Ok, "A Survey on Near Field Communication (NFC) Technology", *Wireless Pers Commun* 71, pp. 2259–2294, 2013.
- [9] S. Tiwari, "An Introduction to QR Code Technology," 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, pp. 39–44, 2016.
- [10] M. Silva, J. P. Dias, A. Restivo, and H. S. Ferreira, "A Review on Visual Programming for Distributed Computation in IoT", Springer Nature Switzerland AG 2021, M. Paszynski et al. (Eds.): ICCS 2021, LNCS 12745, pp. 443–457, 2021.
- [11] M. Zimmermann, "Teaching Visual Programming: Humanoid Robot Programming as a Case Study", 15th International Conference On Education and New Learning Technologies, pp. 6143-6149, 2023.
- [12] A. M. Marei et al., "A SLAM-Based Localization and Navigation System for Social Robots: The Pepper Robot Case", in *Machines*, vol. 11, issue 2, 2023.
- [13] <http://www.omg.org>, Accessed July 2023.
- [14] MIT App Inventor. <https://appinventor.mit.edu>, Accessed July 2023.
- [15] Thinkable. <https://www.thinkable.com>, Accessed July 2023.