Variability Features in Building Approaches for Context-Aware Mobile Applications

Estevan Gómez-Torres[1(✉)], Cecilia Challiol[2,3], and Silvia E. Gordillo[2,4]

[1] Carrera de Ingeniería en Informática, Universidad UTE,
Av. Mariscal Sucre y Mariana de Jesús, Quito, Ecuador
estevan.gomez@ute.edu.ec
[2] LIFIA, Facultad de Informática, UNLP, La Plata, Buenos Aires, Argentina
{ceciliac,gordillo}@lifia.info.unlp.edu.ar
[3] CONICET, Buenos Aires, Argentina
[4] CICPBA, Buenos Aires, Argentina

Abstract. The growth of mobile applications has been exponential in the last couple of years and it has come with technological advances, such as embedded sensors in mobile devices. This brings out greater challenges in the development of context-aware mobile applications, according to the demands of the current market. Currently, there are building approaches for this kind of applications, but these do not have flexibility in the generated applications. Until now, there is not a unified solution for this kind of applications, so, this is an open research area. This paper presents a taxonomy of variability concepts (Relevance, Combination, Precision and Accuracy's Margins, Configuration Type, and Execution Type) to be taken into account when designing building approaches for context-aware mobile application. When these approaches are designed from scratch considering these variability concepts, this allows generating a wide variety of applications. The contribution of our taxonomy is to help the designer to identify the potential variability points in order to obtain more flexible approaches. The aim is to generate a discussion in relation to the variability concepts of the proposed taxonomy, this provides guidelines to be able to achieve variability in this kind of approaches. We hope this will enrich the discussion in relation to this kind of approaches in order to the unification of features that should be handled by these building approaches to obtain variability.

Keywords: Building approaches · Context-aware mobile applications · Variability · Mobile computing

1 Introduction

In recent years, the growth of mobile applications has been exponential and it has come with technological advances such as embedded sensors (GPS, accelerometer, etc.) in mobile devices. This kind of progress has allowed context-aware mobile applications

E. Fonseca C et al. (Eds.): TICEC 2019, AISC 1099, pp. 109–123, 2020.
https://doi.org/10.1007/978-3-030-35740-5_8

[1] to penetrate the market, becoming more and more commons. Until a couple of years ago, this was only a research topic and the applications that have been built did not go beyond the prototypical stage. This generates a new challenge when having to think about how to support the creation of this kind of applications to adapt to the demand.

The concept of context has been explored in different areas of Computer Science [2] (e.g.: Artificial Intelligence, Home Automation, etc.). However, this is carried out from different perspectives depending on each author [1], for example, modeling solutions, building approaches for this kind of applications, etc. Although, this is a topic that has been investigated in the last twenty years, there is not yet unified solution for this kind of applications as mentioned in [1] and [3]. Therefore, it remains an open area of research. Moreover, it has emerged a new issue, which consists of how to build context-aware mobile applications that are really useful for users [4], something vital in today's market.

A feature of this kind of applications is the variability, which can be managed from different levels of abstraction. For example, in [5] a way is proposed to provide variability support at the modeling level at both designing and execution time. In [6], a taxonomy is detailed understanding the differences and similarities between various ways of handling the variability in context-aware software. When should be define building approaches for context-aware mobile applications, there are not clear guidelines to take into account to ensure that these support variability, particularly in the kind of applications that can be generated. Besides, there are building approaches for both non-expert users [7] and experts [8] which require some technical knowledge as modeling features. However, these approaches are not designed to have variability features in the applications that derive; for example, they only provide GPS as a location mechanism.

The aim of this paper is to present a taxonomy that specifies variability concepts to be taken into account when designing approaches for the creation of context-aware mobile applications. When these approaches are designed from scratch considering these variability concepts, this allows generating a wide variety of applications. The contribution of our taxonomy is to help the designer to identify the potential variability points in order to obtain more flexible approaches. To do that, each concept of the taxonomy is described using a pattern-based format, describing how and what could be considered in the designing phase of this kind of approaches.

The aim is to generate a discussion in relation to the variability concepts of the proposed taxonomy, this provides guidelines to be able to achieve variability in this kind of approaches. We hope this will enrich the discussion in relation to this kind of approaches in order to the unification of features that should be handled by these building approaches to obtain variability.

The paper is structured as follows. In Sect. 2, related works are presented which are related to the variability of context-aware applications. The proposed variability taxonomy is detailed in Sect. 3. In Sect. 4 a discussion of the topic is generated. Conclusions and Future works are detailed in Sect. 5.

2 Related Works

The concept of variability in context-aware applications has been handled from different perspectives. In [9], 36 different ways of modeling context-aware are analyzed, which implies variability in the form of representation of this kind of applications. Based on these identified models, in [9] 10,498 elements of context are identified which more than half do not have a clear classification of how to categorize them. In some cases, context-aware models allow for handler variability not only at the design level, but also at execution time [5]. This allows, for example, dynamically add context while the application is running without requiring to be compiled again. Therefore, to have this support, the modeling approach should be designed to consider variability.

In [6], a taxonomy is presented understanding the differences and similarities in how variability is handled in context-aware software. This taxonomy focuses on analyzing three axes: mechanisms to support variability in binding time (post-deployment and run-time), context-feature's types and dependencies between contextual and no-contextual features. In [6] the authors are focus on variability in context-aware software not on building approaches.

There are currently several building approaches for context-aware mobile applications. For example, the *App Inventor* [7] is an "*online*" program, which allows users to create *Android* applications without having any technical knowledge. The generated applications can include only GPS as a location-sensing mechanism. That is, users can only create applications for outdoor spaces. The *App Inventor* allows configuring precision and accuracy of the location sensor (in this case, GPS). However, it is not possible to combine accelerometer sensor to orientation sensor, in that the configuration reacts to these three sensors separately. In this case, the *App Inventor* only focuses on some contexts of the device and the user's location.

On the other hand, *WebRatio Mobile* [8] allows users to create context-aware mobile web applications. *WebRatio Mobile* is oriented to expert users who should have knowledge of databases and hypermedia design. The generated applications are packaged in *PhoneGap*, facilitating their use on both *Android* and *iOS* platforms. The users could define contexts related to *Device, Network Connectivity and Location*. In this case, location setting is also limited to GPS, allowing only applications for outdoor spaces to be generated. It can define precision and accuracy related to GPS.

Considering the description mentioned above, these approaches [7] and [8] are limited. In particular, they focus only of building applications for outdoor spaces where certain contexts are also considered, most of them related to the available APIs. Therefore, these approaches are not designed to support variability in the kind of generated applications. We hope to contribute in this aspect with the proposed taxonomy in this paper.

3 The Proposed Variability Taxonomy

In [10] we present an initial version of a taxonomy for context-aware mobile applications building approaches. Only a brief description of each concept of the taxonomy is detailed in [10], without going into about how should be handled in building approaches.

We have been working in the area of context-aware mobile applications from more than ten years, this allows us to know and identify points of variability in this kind of applications. According to that, we think that the format used for design patterns in different areas of Software Engineering is a good way to describe our taxonomy. Thus, in this paper will expand each concept by describing them with a pattern-based format.

Note that each concept is described not only considered the existing literature (such as [1–5]) but also based on our expertise in the area.

Each pattern of variability has the following structure:

- *Name of the Variability Concept*
- *Scenario*: Describe situations in which the concept be handled by building approaches. This allows understanding situations that could be presented to the user of a building approach, and how this concept would provide flexibility if it should be handled by building approaches.
- *Purpose*: Define the specific objective or motivation for handler the concept from a building approach.
- *Applicability*: Every concept has challenges that should be faced when it wants to be implemented as part of a building approach. These challenges could be occur at both levels conceptual approach and tool. Therefore, how it is feasible to handle these challenges be shown.

  – *Applicability at Conceptual Approach Level*: Represented the conceptual way to handle these challenges. This is the first step that the challenges be handled by a tool.
  – *Applicability at Tool Level*: The ability of a tool to handle these challenges. In other words, how the challenges can be treated from a tool. Sometimes happen that at the conceptual level could be achieved, but at a tool level demands a lot of cost or it is not viable according to the current technology.

Using the structure described above, each variability concept of the proposed taxonomy is described below in Tables 1, 2, 3, 4, 5 and 6.

Table 1. Variability concept: relevance.

Name of the variability concept	Relevance
Scenario	Classify, in some way, the level of importance of a context would allow working with contexts at different levels. For example, to take into account only the most important contexts to generate reduced versions of applications which only provide services related to some selected contexts. Having reduced versions allows, for example, generating applications that are less heavy or that consume fewer resources
Purpose	Being able to specify the importance of each context, from a building approach, would allow making different decisions
Applicability at conceptual approach level	A way of indicating the relevance of each defined context should be provided, for example, using a numerical or descriptive ranking. This could do it by associating a certain value with each defined context. The scale of relevance should be clearly established
Applicability at tool level	Using the scale of relevance defined at the conceptual level, from a tool could be implemented using, such as, a combo-box with the possible values. In addition, some actions could be taken related to this values. For example, this could be implemented in the tool to be considered when applications are derived, indicate which contexts are considered in them Although specifying the relevance is technically feasible to implement in a tool, handling how this impacts, for example, in the derivation of applications and also involves defining heuristics of what actions to take based on the indicated values Note that the flexibility of the tool could be affected by the heuristics that are defined in relation to relevance

Table 2. Variability concept: combination.

Name of the variability concept	Combination
Scenario	Allowing indicating which contexts could be combined, for example, to provide services that are more complex. This implies defining which context they want to combine but also how they will behave together to provide, for example, services
Purpose	Being able to specify how contexts can be combined, from a building approach, would allow providing complex services that depend on several contexts

(*continued*)

Table 2. (*continued*)

Name of the variability concept	Combination
Applicability at conceptual approach level	A way of indicating how to combine contexts should be provided; for example, grouping them in some way. Moreover, each group should define how the different values of each contexts are working together to provide, for example services. This could be implemented using rules as if-them in which the conditions indicated what is the specific value of each considered context, in order to apply the rule
Applicability at tool level	In a tool could be used the specification of combination defined at the conceptual level. For example, visually represent the group of contexts that want to combine, and then it associates services to this group Specify the combination of contexts is technically feasible to implement in a tool; however, the specification of how to combine services can be complex since there should be clear rules of how to react to each value that these contexts could take It is worth mentioning that sometimes the contexts take values from sensors; this brings as a consequence the fact should be considered the margin of error of these sensors when defining how the contexts behave when they are combined

Table 3. Variability concept: precision and accuracy's margins.

Name of the variability concept	Precision and accuracy's margins
Scenario	Defining values of precision and accuracy's margin related to physical sensors allow having what is the error respect to the sensed value. This helps to interpret better the context's values that depend on these sensors. For example, the accuracy of the GPS will allow knowing how much the user's location is accurate
Purpose	Being able to specify the precision and accuracy in relation to physical sensors, from a building approach, would allow interpreting better the context's values. According to these values, different decisions could be made

(*continued*)

Table 3. (*continued*)

Name of the variability concept	Precision and accuracy's margins
Applicability at conceptual approach level	A way to indicate the precision and accuracy related to physical sensors should be provided. Different actions could be specified according to these values, for example, to adjust the sensed value according to the precision or accuracy which has. This could be represented using value ranges
Applicability at tool level	A tool could use how precision and accuracy's margin have been defined at the conceptual level. For example, entering numerical values related to physical sensors, to indicate both the precision and the accuracy of them. In addition, it should consider what services (associated with these contexts) can be affected by how these sensors behave. For example, if the GPS has a certain precision and accuracy, then in the generated application it will be possible to reflect the user's location more accurately. Specifying precision and accuracy's margin related to physical sensors is technically feasible to implement in a tool; however, the complexity is associated with how these values affect services related to the contexts. So, this is not only involved to define how to react to each sensed value but also how to consider the precision and accuracy of them

Table 4. Variability concept: categorization.

Name of the variability concept	Categorization
Scenario	Classify the types of context would allow working with them in different ways. For example, the derivation of the contexts of the user, environment or mobile objects does not have the same treatment. That is, it is not the same to take location from the user that a mobile object; this impacts on which sensors are used in each case to take the location
Purpose	Being able to specify to which category a context belongs, from a building approach, would allow being able to make different decisions or to enable different options

(*continued*)

Table 4. (*continued*)

Name of the variability concept	Categorization
Applicability at conceptual approach level	A way of indicating the context's category to which it belongs should be provided; for example, by a description. It should be clearly established what concept is defined and what is associated with each category, for example, sensors available for each category
Applicability at tool level	Using the categories defined at the conceptual level, from a tool could be implemented using, such as, combo-box with the possible values. In addition, it should specify in the tool which restrictions are associated with each category, for example, what sensors are available or how these are derived in the generated applications Although specifying the categories is technically feasible to implement in a tool; however, handling how each of these impacts the generated applications requires defining different heuristics. For example, user's contexts should be specified differently from mobile objects' context, but also be derived differently

Table 5. Variability concept: configuration type.

Name of the variability concept	Configuration type
Scenario	Specifying the configuration type associated with each context would allow providing more flexibility in the generated applications. For example, the configuration type could be passive or active, according to [1]; in which these types are defined in relation to if it requires user's intervention or not. The passive configuration requires specifying what data the user should enter, meanwhile the active configuration is automatic, for example, using automatic learning. It could also represent the default configuration
Purpose	Being able to specify the configuration type, from a building approach, would allow establishing in which way the contexts are set to take their values

(*continued*)

Table 5. (*continued*)

Name of the variability concept	Configuration type
Applicability at conceptual approach level	When the configuration type is chosen, more information should be specified in relation to what each type requires. For example, the passive configuration requires defining what the user should specify. Meanwhile, the active configuration should consider with what automatic mechanism, it will be bound in order to make the configurations dynamically in runtime. In the default configuration, the values should be set. The configuration type impacts on the generated application since it is necessary to incorporate as part of it what each one requires
Applicability at tool level	Using the types of configuration defined at the conceptual level, from a tool could be implemented using, such as, a combo-box with the possible values. In addition, what each type requires should be implemented as a part of building approach in order to generate applications with the corresponding configuration Meanwhile, specifying the configuration type is technically feasible to implement in a tool, handling what each implies is not trivial. For example, the active configuration type requires automatic learning which it would not be simple to have as part of the generated application The passive configuration type should be a little less complex to implement since it could be defined by a form with the data that the user could configure in relation to each context (or sensor associated with it). The default configuration could be set from the tool with, for example, selected options

Table 6. Variability concept: execution type.

Name of the variability concept	Execution type
Scenario	Specifying the execution type associated with each context would allow providing more flexibility in the generated applications. For example, execution type could be passive or active, according to [1]; in which these types are defined in relation to if it requires user's intervention or not. Passive execution requires user intervention (for example, QR code reading); meanwhile active execution is automatic (for example, GPS)
Purpose	Being able to specify execution type, from a building approach, would allow establishing information about how the contexts behave

(*continued*)

Table 6. (*continued*)

Name of the variability concept	Execution type
Applicability at Conceptual approach level	When the execution type is chosen, more information should be specified in relation to what each type requires. For example, passive execution requires defining how the user interact with it, meanwhile active execution is an automatic mechanism without intervention. The execution type impacts on the generated application since it is necessary to incorporate as part of it what each one requires
Applicability at tool level	Using the types of execution defined at the conceptual level, form a tool could be implemented using, such as, a combo-box with the possible values. In addition, what each type requires should be implemented as a part of building approach in order to generate applications with the corresponding mechanisms Specifying the execution type is technically feasible to implement in a tool; however, handling what each implies is not trivial. The passive execution requires detail of how the application will interact with the user; as well as, how it will react for each possible interaction. For example, in the case of having a passive execution mechanism, such as reading QR codes to take the user's location, it should be indicated how the generated application behaves when the user reads an incorrect code. Active execution is simpler to implement because it only requires the available APIs. For example, if GPS is used to take the user's location, it is enough to make the appropriate connection to the location's API as part of the generated application. This execution type is transparent to the users and their intervention is not required It could be possible that a context, such as user's location could involve a mixed execution type, for example, reading of QR codes and GPS. In this case, the two previous solutions are combined, but in addition heuristics should be defined of which mechanism has more priority given that the GPS works constantly. This add complexity in the tools

Each variability concept of the proposed taxonomy is specified above. It could observe that each concept could involve a value to be established, but the complexity arises in relation to how each value impacts or what has to be defined based on it. The biggest challenge comes from being handler from a tool and how to derive applications considering what implies each variability concept. It should be mentioned that the taxonomy is not closed, but is an initial proposal to achieve flexibility in the context-aware mobile applications building approaches.

4 Discussion

In this section, it discusses different aspects of the taxonomy presented in Sect. 3; in order to help the reader understand how each concept can affect the variability supported by the building approaches.

In [5], a model is proposed which considers the separation between the aware-objects concept of its context-feature and in addition from the sensors (which assign values to these features). When this decoupling of concepts is considered from a building approach, it allows the different layers could combine and thus support variability. For example, the location-feature could set its values from different associated sensors, and each of them could has different implementations. This decoupling allows reuse and extensibility. This is a possible way to represent these concepts in a building approach to handler a first level of variability; beyond that incorporates the variability concepts of the proposed taxonomy in this paper.

According to what is detailed in [11], the sensors can be of different types, for example, physical, virtual (using applications and services), direct user input, etc. Each of them has its own configuration and way of execution.

Considering what has been analyzed previously, Fig. 1 is shown an example of two aware-objects (*User 1* and *Package*), each one with different context-features according to its nature.

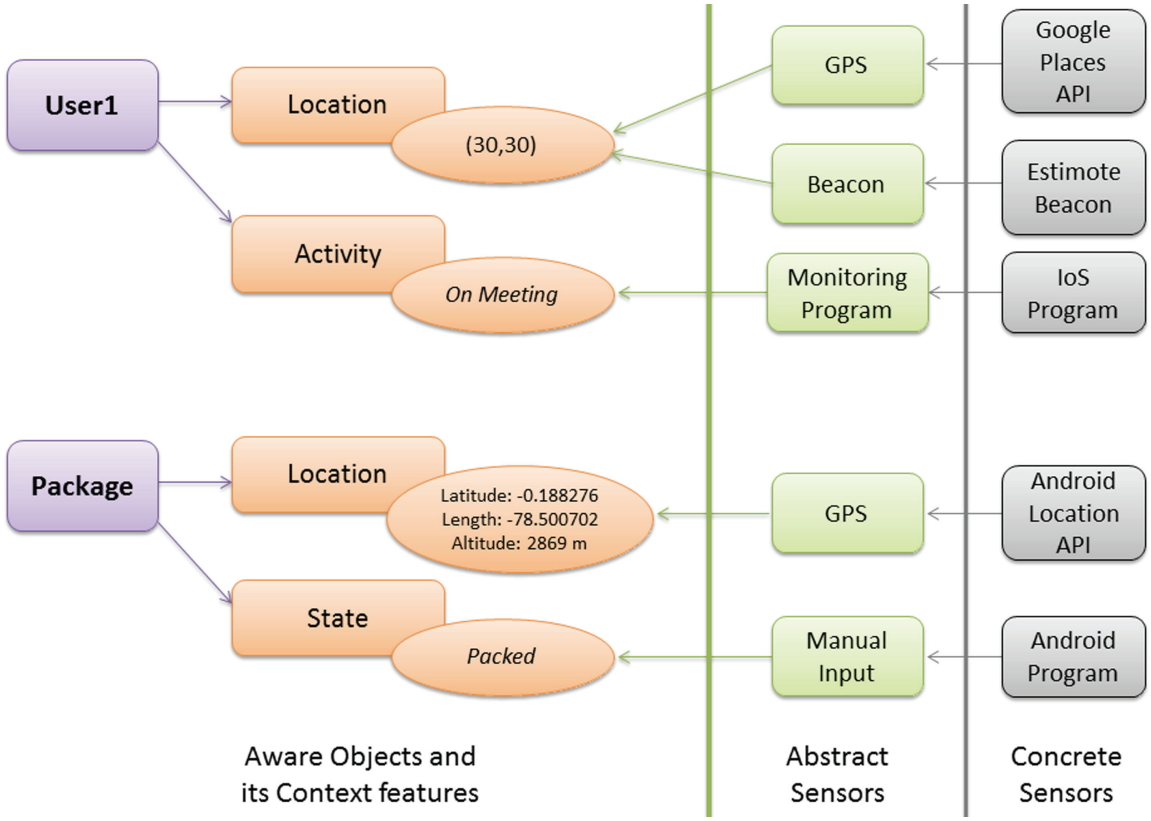

Fig. 1. Examples of aware-objects with its context-features which are related to abstract and concrete sensors.

In addition, in Fig. 1 there is a separation between the concepts of abstract and concrete sensors. This allows the flexibility that the same concept of an abstract sensor is implemented differently, such as occurs with the GPS example; that could be implemented using the *Google Places* API [12] or the *Android Location* API [13]. The separation between the concepts of abstract and concrete sensors when it is considered from a building approach allows having more level of variability in relation to the combination that can be handled.

It can also be observed in Fig. 1 that two sensors could set values to the same context-feature (in this case, the location-feature). This decoupling allows sensors to be added or removed without affecting the context concepts. This provides flexibility when considered as part of a building approach.

Based on the example in Fig. 1, it is analyzed with more details how the variability concepts of the proposed taxonomy can be considered.

The aware-objects could have its category associated. The range's values of the category are used by the building approaches when deriving context-aware mobile applications. For example, deriving code that represents the user concept is not the same as referring to a mobile object.

The relevance could be associated with the context-features. The relevance range could be used by the building approaches to generate reduced applications without all the functionality. For example, in the example presented in Fig. 1, the location could be more relevant than the activity; this depends on the services of the application that are being modeled.

The combination could occur observing the values of different context-feature. From building approaches should be able to allow the specification of rules, as well as the actions that it triggers. Each time that a context-feature change its value the rules are evaluated.

In the case of triggering services, they should be integrated into the building approach; in order to later derive these as part of the generated applications. Other triggers could update some context-features values. In the example of Fig. 1, it could happen that *User 1* is waiting for the *Package* to arrive; a rule could be that each time the object's state changes, the user is notified as long as she/he is not in a meeting. In this case, the rule observes the object's state and the user's activity, and based on its values, triggers the warning to the user as a service.

Based on the analysis carried out, in Fig. 2 is shown a possible generalization of the variability concepts: categorization, relevance and combination.

Following the analysis of the variability concepts of the proposed taxonomy, each of the specific sensors could involve different types of configuration and execution. That is, at this level these variability concepts could be handled from building approaches.

The configuration type is useful for a building approach since it allows identifying if user's intervention would be required or not. In the case of it is requiring it is important to define how this would be carried out. In the case "*Active*" configuration [1], it should be taken into account that the building approach should have, at least, one way of including the monitoring program as part of the generated application. "*Passive*" configuration [1] when is supported by building approach should have, at least, one way of indicating how the user will perform such a configuration; for example, designing the form which would

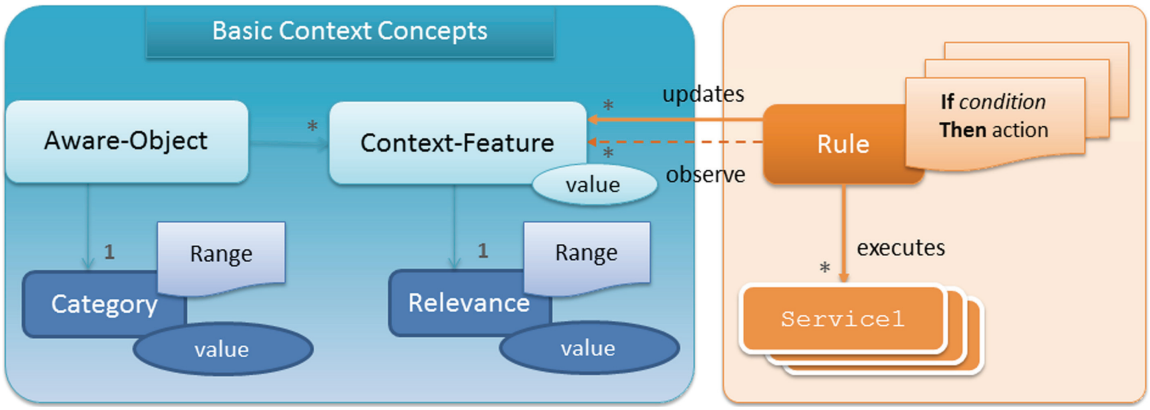

Fig. 2. Generalization of the variability concepts: relevance, combination and categorization.

then be embedded in the generated application. In the case of "*default*" configuration, the generated application defines the fixed values which have been defined using the building approach.

In the example of Fig. 1, the APIs related to GPS and *Estimote Beacons* [14], they could be set, for example, as the *default* configuration. The monitoring program could be set as "*Active*" configuration, so, it could learn to configure itself as it monitors. Whereas the manual entry could have, for example, a "*Passive*" configuration, being able to design from the building approach the form in which the user could enter new possible values.

For the execution type, some specific sensors determine automatically how they work; this facilitates the auto-completion of this value when it is handled by a building approach. For example, if the option is GPS or *Beacons,* they have an "*Active*" execution, meanwhile manual entry is "*Passive*" and it should define how the user could interact with the generated application, for example, using a form. In the case of the monitoring program could require or not user's intervention, and this defines the execution type associated with it. In this last case, the building approach could not autocompleted the execution type associated with the sensor.

The specification of precision and accuracy is usually associated with concrete physical sensors. Building approaches should consider the range's values that precision's margin can take. However, this is used at runtime in the derived applications; that is, the margin of error is specified in the building approach but allows decisions to be handled in the generated applications. In the example of the Fig. 1, the APIs related to GPS and *Beacons Estimote* are those that could define precision and accuracy's margins.

Based on the analysis carried out, in Fig. 3 is shown a possible generalization of the variability concepts: configuration type, execution type and the precision and accuracy's margins.

Figures 2 and 3 allow to observe a way of representing the variability concepts which should be associated with aware-objects, context-features or specific sensors. This is important when designing building approaches to consider these variability concepts.

In this session, it has been discussed as a possible way to represent the variability concepts of the proposed taxonomy. This representation could vary according to how

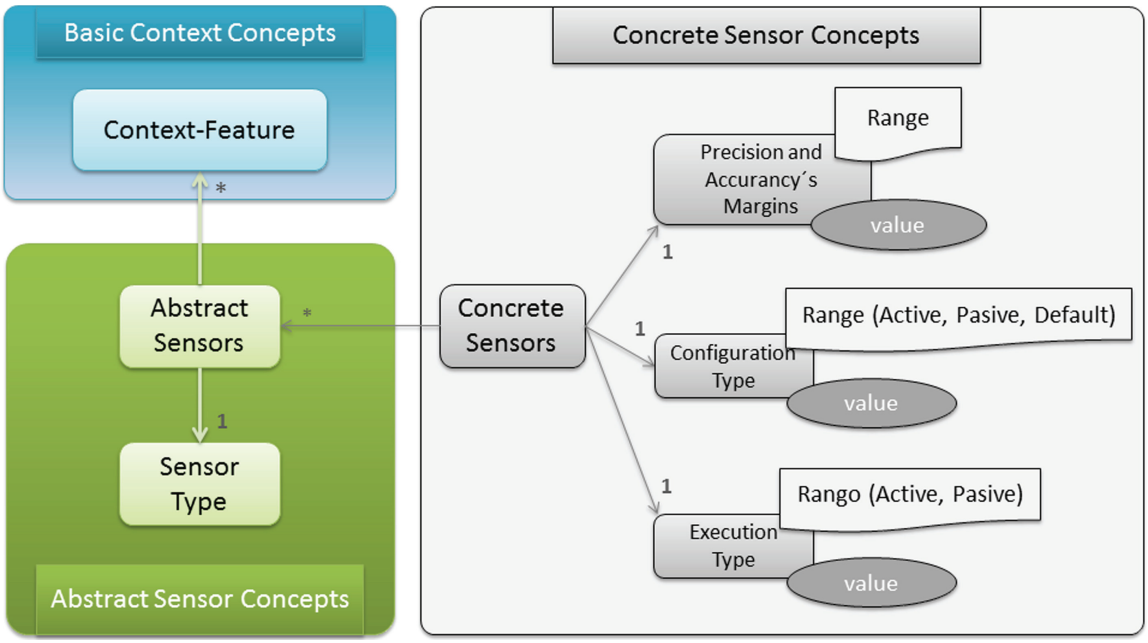

Fig. 3. Representation of the variability concepts: configuration type, execution type and precision and accuracy's margins.

the basic concepts to be represented by each building approach. The examples presented in this session are based on the concepts defined in [5].

5 Conclusions and Future Work

In this paper, a taxonomy was presented that specifies variability concepts to be taken into account when designing approaches for the creation of context-aware mobile applications. This taxonomy is focused on providing support to generate a wide range in the kind of applications from these approaches. Each variability concept of the taxonomy is described using a pattern-based format, focusing on the challenges involved in considering each of them as part of an approach to build this kind of applications. The proposed taxonomy is not complete or closed, but it is the first definition in order to have variability in the building approaches.

In addition, a discussion has been generated in relation to the variability concepts of the proposed taxonomy. It has been used the concepts defined in [5] in which separates the concepts of aware-objects, context-features and sensors; based on this, a possible way of handler each variability concept was described. Using this paper, designers of building approaches for context-aware mobile applications have a guidelines to be able to achieve variability. We hope the presented discussion will enrich how could be handled variability in these building approaches. Moreover, we wish to contribute to a unified solution to this kind of approaches.

As future work, a concrete building approach will be designed to put into practice the proposed taxonomy. It is desirable that by designing this approach, the taxonomy could be enriched by incorporating new variability concepts. In addition, we will analyze how to enrich our taxonomy with other aspects of variability such as those proposed in [6].

References

1. Alegre, U., Augusto, J.C., Clark, T.: Engineering context-aware systems and applications: a survey. J. Syst. Softw. **117**, 55–83 (2016)
2. Augusto, J., Aztiria, A., Kramer, D., Alegre, U.: A survey on the evolution of the notion of context-awareness. Appl. Artif. Intell. **31**(7–8), 613–642 (2017)
3. Bauer, C., Dey, A.K.: Considering context in the design of intelligent systems: current practices and suggestions for improvement. J. Syst. Softw. **112**, 26–47 (2016)
4. Alegre-Ibarra, U., Augusto, J.C., Evans, C.: Perspectives on engineering more usable context-aware systems. J. Ambient Intell. Humanized Comput. **9**(5), 1593–1609 (2018)
5. Fortier, A., Rossi, G., Gordillo, S.E., Challiol, C.: Dealing with variability in context-aware mobile software. J. Syst. Softw. **83**(6), 915–936 (2010)
6. Mens, K., Cardozo, N., Duhoux, B.: A context-oriented software architecture. In: 8th International Workshop on Context-Oriented Programming, pp. 7–12. ACM, New York (2016)
7. Bales, S.: Build Android Apps Without Coding: Get Started with Android Apps Using Thunkable-MIT app Inventor. Independently published. ACM, New York (2018)
8. Hamdani, M., Butt, W.H., Anwar, M.W., Azam, F.: A systematic literature review on interaction flow modeling language (IFML). In: 2nd International Conference on Management Engineering, Software Engineering and Service Sciences, pp. 134–138. ACM, New York (2018)
9. Bauer, C., Novotny, A.: A consolidated view of context for intelligent systems. J. Ambient Intell. Smart Environ. **9**(4), 377–393 (2017)
10. Gómez-Torres, E.R., Challiol, C., Gordillo, S.E.: Context-aware mobile applications: taxonomy of factors for building approaches. In: XXV International Conference on Electronics, Electrical Engineering and Computing, pp. 1–4. IEEE (2018)
11. Rivero-Rodriguez, A., Pileggi, P., Nykänen, O.A.: Mobile context-aware systems: technologies, resources and applications. Int. J. Interact. Mobile Technol. **10**(2), 25–32 (2016)
12. Google Place API. https://developers.google.com/places/web-service/intro. Accessed 28 June 2019
13. Android Location API. https://developer.android.com/training/location. Accessed 28 June 2019
14. Estimote Beacons. https://estimote.com. Accessed 28 June 2019