Desafíos de los enfoques de creación de aplicaciones móviles sensibles al contexto

Challenges in context-aware mobile applications building approaches

Estevan Gomez-Torres
UTE, Facultad de Ingeniería, Carrera de Ingeniería en Informática and also Facultad de Informática, UNLP
Quito, Ecuador
estevan.gomez@ute.edu.ec,
estevan.gomezt@info.unlp.edu.ar

Cecilia Challiol
UNLP, Facultad de Informática, LIFIA
and also CONICET
La Plata, Buenos Aires, Argentina
ceciliac@lifia.info.unlp.edu.ar

Silvia E. Gordillo
UNLP, Facultad de Informática, LIFIA
and also CICPBA
La Plata, Buenos Aires, Argentina
gordillo@lifia.info.unlp.edu.ar

***Resumen*— Las aplicaciones móviles sensibles al contexto están cada vez más presentes en nuestra vida cotidiana, permitiendo a las personas realizar diferentes tareas a través del uso de dispositivos móviles. Dada la complejidad que tiene diseñar este tipo de aplicaciones, han surgido diferentes herramientas (o entornos de desarrollo) que facilitan su creación. Sin embargo, estos entornos muchas veces estas limitados en el tipo de aplicaciones que generan; los cuales se basan generalmente en las APIs de sensores disponibles como, por ejemplo, el GPS. De todos los entornos existentes que podrían permitir diseñar y generar aplicaciones móviles sensibles al contexto es de interés para este trabajo aquellos que permiten el modelado basado en UML. Acorde a esto, este trabajo analiza distintos entornos para poder determinar la variabilidad en los tipos de aplicaciones que se pueden generar con los mismos. Para realizar este análisis se toma de base los conceptos planteados en una taxonomía de variabilidad para enfoques de creación de aplicaciones móviles sensibles al contexto. Se plantea un espacio de discusión en relación a los desafíos que conlleva abordar la variabilidad desde los entornos que brindan soporte para la creación de este tipo de aplicaciones.**

Palabras Clave*— *aplicaciones móviles sensibles al contexto; entornos de desarrollo; variabilidad; UML

***Abstract*— Context-aware mobile applications are immersed in our daily lives, allowing people to perform various tasks through the use of mobile devices. The complexity of designing these kinds of applications generate that different development tools have emerged to facilitate its creation. However, these development tools are often limited in the kind of applications that they derive which are generally based on the available APIs' sensors, such as GPS. From all the existing tools that could be allow to design and derive context-sensitive mobile applications, we are interested only in those that allow UML-based modeling. According to that, this paper analyzes different development tools to determine the variability of the kind of applications that can be generated with them. In order to carry out this analysis, we have taken into account the concepts proposed in a taxonomy of variability in building approaches for context-aware mobile applications. A discussion is presented in relation to the challenges involved to support variability in these tools.**

Keywordst*— *context-aware mobile applications; development tools; variability; UML

I. INTRODUCCIÓN

La masificación de los dispositivos móviles en la vida cotidiana y los importantes avances de la tecnología inalámbrica se han convertido en una fuerza impulsora cada vez mayor de una variedad de aplicaciones móviles sensibles a los contextos [1], las cuales utilizan datos del entorno que rodea al usuario para brindar diversos tipos de servicios. Dada la complejidad involucrada en este tipo de aplicaciones, por ejemplo, manejo de sensores, variados tipos de dispositivos, y más aún, servicios que están constantemente evolucionando; emergió la necesidad de contar con enfoques o herramientas que permitan la construcción de estas aplicaciones usando un nivel de abstracción alto, como pueden ser modelos [2]. Esto permite que los desarrolladores no tengan que conocer a bajo nivel cómo funcionan, por ejemplo, los sensores de los dispositivos móviles para poder crear una aplicación que hace uso de los mismos para brindar servicios.

Uno de los enfoques que propone un mecanismo de abstracción alto para dar solución al desarrollo de aplicaciones, es el *Desarrollo Dirigido por Modelos* (MDD) [3]; el cual permite definir la información de la aplicación mediante

modelos para luego llegar a aplicaciones funcionales. En el área de aplicaciones móviles, podemos encontrar ejemplos de entornos como son *MIT Appinventor* [4] y *Webratio Mobile* [5]. Sin embargo, este tipo de entornos tienen ciertas limitaciones, como se analizan en [6] y [7], donde se deduce que las aplicaciones generadas por estos entornos están destinadas solo a espacios abiertos; y se pueden modelar solo algunos contextos, la mayoría de ellos en relación a las APIs de desarrollo disponibles (por ejemplo, la API de GPS). Cabe mencionar que *MIT Appinventor* y *Webratio Mobile* no contemplan, por ejemplo, categorías de contextos [2]. Es decir, el tipo de aplicaciones que se pueden generar son limitadas.

Actualmente sigue siendo un área abierta de investigación los enfoques que permiten la creación de aplicaciones móviles sensibles al contexto [8], de hecho no hay un acuerdo o una guía de cómo los mismos deben ser diseñados. En [6] se plantea una taxonomía de factores que permite identificar cómo los enfoques de creación de aplicaciones móviles sensibles al contexto deberían considerar estos conceptos para lograr variabilidad en las aplicaciones que los mismos generan. En [6] estos factores son descritos y en [7] los autores amplían está taxonomía analizando cómo los mismo pueden ser abordados desde enfoques de modelados; sin embargo, no se ha analizado aún cómo las herramientas de creación de este tipo de aplicaciones deben considerar estos factores, o más aún no es claro qué tipo de variabilidad soportan los entornos de desarrollo existentes. Esta es la principal motivación de este trabajo.

En este trabajo se analizan distintos entornos de desarrollo existentes poniendo el foco en cómo los mismos abordan la variabilidad para poder generar distintos tipos de aplicaciones. En particular, los entornos que se analizan permiten el modelado basado en UML [10]. Para este análisis se toman de base los conceptos de variabilidad de la taxonomía para enfoques de creación de aplicaciones móviles sensibles al contexto propuesta en [6] y ampliada en [7]. Además, de la bibliografía existente y nuestra experticia en la temática.

El aporte de este trabajo es comprender cómo los entornos existentes abordan la variabilidad para poder generar distintos tipos de aplicaciones. También se busca detectar cuáles podrían extenderse fácilmente para poder brindar otros tipos de aplicaciones; logrando así en un futuro mayor variabilidad en los tipos de aplicaciones generadas.

A partir del análisis, se genera un espacio de discusión en relación a los desafíos que conlleva abordar la variabilidad desde los enfoques de creación de este tipo de aplicaciones. Se espera que este trabajo sirva de guía en una etapa temprana de diseño de este tipo de entornos (o herramientas), para que los mismos cuenten con la flexibilidad para luego poder generar una amplia gama de aplicaciones móviles sensibles al contexto.

Este trabajo se estructura de la siguiente manera. En la Sección 2 se presenta el estado del arte en relación a la temática del trabajo. En la Sección 3 se analizan diferentes entornos de desarrollo existentes que podrían dar abordaje a la creación de aplicaciones móviles sensibles al contexto, en particular algunos que brindan soporte al modelado basado en UML. En la Sección 4 se presenta un espacio de discusión. Las conclusiones y trabajos futuros son presentados en la Sección 5.

II. Estado del Arte

En relación al trabajo es de interés poder comprender por un lado el concepto de variabilidad pero también las características de algunos entornos de desarrollo existentes que permiten el modelado basado en UML en relación a este concepto; los cuales luego serán analizados. Además, se presentan algunos trabajos existentes que se relación con la temática.

A. Variabilidad en Aplicaciones Móviles Sensibles al Contexto

El concepto de variabilidad en aplicaciones sensibles al contexto puede ser abordado desde diferentes perspectivas. En [8], se analizan 36 formas diferentes de modelar la sensibilidad al contexto, lo que implica variabilidad en la forma de representación de este tipo de aplicaciones. En base a los modelos mencionados, en [8] se identifican 10,498 elementos de contexto que más de la mitad no tienen una manera apropiada de cómo clasificarlos. En algunos casos, los modelos sensibles al contexto permiten la variabilidad no solo a nivel de diseño, sino también en tiempo de ejecución [9].

En [11] se presenta una taxonomía que describe las diferencias y similitudes en la forma en que se maneja la variabilidad en los softwares sensibles al contexto. Esta taxonomía se centra en analizar tres ejes: mecanismos para admitir la variabilidad en el tiempo de "*enlace*" (posterior a la implementación y el tiempo de ejecución), los tipos de características contextuales y las dependencias entre características contextuales y no contextuales. En [11] los autores se centran en la variabilidad en el software sensible al contexto, no en la construcción de enfoques.

También existe variabilidad en los tipos de sensores que se pueden usar en las aplicaciones sensibles al contexto. Por ejemplo, en [12] se definen distintos tipos de sensores que pueden tener las aplicaciones móviles sensibles al contexto:

- *Sensores Físicos*: son elementos físicos capaces de capturar datos del entorno de la entidad.
- *Sensores Virtuales*: son formas de acceder a información virtual, esto se puede lograr usando aplicaciones y servicios. Un ejemplo de este tipo de sensor puede ser el calendario del dispositivo móvil del usuario, pudiendo así información de la actividad que debe realizar el usuario en cada momento.
- *Sensores Combinados*: proporcionan información obtenida al combinar información de dos o más sensores. Es decir, que el dato que se obtiene de esta información determina el valor del contexto.
- *Redes sociales*: se consumen datos de estas redes sociales para poder determinar valores de contexto. Por ejemplo, redes de contactos o gustos de un usuario.
- *Entrada directa del usuario*: es una alternativa para obtener datos pero este tipo de sensado depende del usuario, y los datos que este complete. Es decir, el usuario completa directamente la información contextual. Con este tipo de sensado se pierde lo transparente que puede llegar a ser el sensado de valores de contexto.

En [6] se plantea una taxonomía de factores que permite identificar cómo los enfoques de creación de aplicaciones móviles sensibles al contexto deberían considerar estos conceptos para lograr variabilidad en los tipos de aplicaciones que los mismos generan. A continuación se brinda un breve detalle de cada uno de estos factores para ayudar al lector a comprender el análisis realizado en el presente trabajo.

- *Relevancia:* permitir indicar la relevancia de los contextos definidos; esto facilita, por ejemplo, generar versiones reducidas con los contextos más relevantes.
- *Combinación:* permitir combinar contextos, esto implica que no sólo identificar que contextos son combinables, sino su comportamiento cuando se consideran en conjunto.
- *Margen de Precisión y exactitud:* poder especificar la precisión y exactitud para cada sensor físico [12], como por ejemplo GPS. Esto luego es útil para interpretar el valor sensado en las aplicaciones generadas.
- *Categorización*: tener una forma para categorizar contextos. Por ejemplo, los contextos del usuario, del ambiente o de los objetos móviles requieren ser manejados en forma diferente. Esto impacta luego en cómo cada uno de ellos es embebido en las aplicaciones generadas.
- *Tipo de configuración*: permitir indicar el tipo de configuración que se usará, por ejemplo activa o pasiva, como se detalla en [1]. Esto está relacionado con los diferentes tipos de sensores posibles, donde cada uno puede tener una configuración particular.
- *Tipo de ejecución*: permitir indicar el tipo de ejecución, por ejemplo, activa o pasiva, como se detalla en [1]. Esto también está relacionado con los diferentes tipos de sensores posibles, donde cada uno puede tener una configuración particular.

En [7] los autores amplían está taxonomía analizando cómo los factores pueden ser abordados desde enfoques de modelados; sin embargo, no se ha analizado aún cómo las herramientas de creación de este tipo de aplicaciones pueden abordarlos, esto es la motivación del presente trabajo.

B. Entornos de desarrollo que brindan soporte al modelado basado en UML

Algunos de los entornos existentes no fueron creados para generar aplicaciones móviles sensibles al contexto; sin embargo, al considerar el uso de sensores de los dispositivos permiten dar solución a este tipo de aplicaciones. Estos son de interés de análisis en esta subsección y se describen a continuación.

Eclipse [13] es un entorno de desarrollo que permite crear diferentes tipos de modelos y aplicaciones, el mismo se integra con una serie de editores y asistentes para dar solución a distintos aspectos del desarrollo de aplicaciones. El *Marco de Modelado de Eclipse* (EMF) [13], permite la generación de código para crear herramientas y otras aplicaciones basadas en un modelo de datos estructurados (XMI). Por ejemplo, a partir de modelos se pueden generar de clases Java. Se cuenta con la posibilidad de visualizar los modelos de forma gráfica mediante el *Marco de Modelado Gráfico* (GMF) [14].

Cabe mencionar que dentro del DDM existen transformaciones de *Modelo a Modelo* (M2M) y de *Modelo a Texto* (M2T), estas últimas son las que dan lugar a la derivación de aplicaciones concretas. Uno de los lenguajes usado para especificar las transformaciones es ATL [15] (*Atlas Transformation Language*). Para la generación de código, *Eclipse* permite configurar e instalar un plugin con el SDK de Android, para así generar aplicaciones móviles.

Sirius [16] es un proyecto *Eclipse* que permite crear herramientas basándose en *EMF*. *Obeo Sirius* [17] permite la generación de editores, representación gráfica de modelos y meta modelos; se usa para crear lenguajes específicos de dominio gráficos (*DSLs*) [18]. Cabe mencionar que si se utiliza *Obeo Sirius* se debe adicionar, por ejemplo, el plugin de ATL [19] para la generación de código.

JBoss [20] es un proyecto *Eclipse* que incluye un amplio conjunto de capacidades de herramientas y soporte para múltiples marcos y modelos de programación, incluido el desarrollo de contenedores [21] y trabaja en conjunto con *Híbrida Mobile tools* [22] y el amplio soporte multiplataforma de *Apache Cordova* [23] para la generación de aplicaciones híbridas, usando los plugins para control del dispositivo móvil y geolocalización, entre otras funcionalidades.

Visual Studio [24] es un entorno de desarrollo basado en C# que permite la creación de aplicaciones móviles multiplataforma al integrarse con *Xamarin* [25] y con *Apache Cordova* [23]; cabe mencionar en su última versión ha dejado de dar soporte al modelado basado en UML, un dato a considerar a la hora de elegirlo como entorno de desarrollo. Para la generación de código usa plantillas pre-definidas tanto en *Xamarin* como en *Android* y *iOS*, las mismas que pueden ser personalizadas, mediante el código apropiado, a fin de darles la funcionalidad que se requiere.

C. Comparación de herramientas o frameworks existentes

En [26] se presenta un estudio comparativo de varias herramientas para la creación de aplicaciones móviles, en particular, aplicaciones *Android*. En [26] se detallan exhaustivamente *jQuery Mobile, PhoneGap* y *App Inventor*, analizando cuál de ellos es recomendable utilizar dependiendo de la categoría a la cual pertenece la aplicación que se desea realizar, y los conocimientos previos con los que cuenta el desarrollador. Los autores detallan en [26] la posibilidad de contar con la información a la hora de elegir la alternativa de programación en *Android* para obtener el mejor resultado con el menor esfuerzo posible.

En [27] se realiza una evaluación de varios frameworks temiendo en cuenta la usabilidad, los cuales utilizan principalmente DSL como herramienta para la creación de aplicaciones; sin embargo no se abordan características relacionadas con las aplicaciones móviles sensibles al contexto. En particular, en [27] se realiza la evaluación para determinar la manera en la que estos frameworks se pueden usar para personalizar editores gráficos para un *DSML*. Las herramientas analizadas son: *IBM Rational Software Architect (RSA)*, *GME*

(entorno de modelado genérico), MetaEdit+, *Obeo Designer* y *GMF* (el marco de modelado gráfico).

En [28] se analizan las características de rendimiento de las aplicaciones móviles, desarrolladas con varios entornos de desarrollo, incluyendo el *SDK de Google* para *Android*, y herramientas multiplataforma de *Apache Cordova*, *Microsoft Xamarin* y *Appcelerator Titanium*. Los autores desarrollaron un grupo de aplicaciones con las mismas características, para realizar la evaluación de rendimiento bajo idénticas condiciones.

Cabe mencionar que los trabajos [26], [27] y [28] se focalizan en el desarrollo de aplicaciones móviles, sin entrar en detalle de las características sensibles al contexto.

III. Características de entornos de desarrollo que permiten el modelado basado en UML

En esta sección se analizan las características de algunos entornos de desarrollo que permiten el modelado basado en UML [10], los cuales podrían permitir la generación de aplicaciones móviles sensibles al contexto. Esto ayudará a comprender cómo los mismos abordan la variabilidad a la hora de generar distintos tipos de aplicaciones.

Para determinar qué características son de interés analizar se toma de base los conceptos de variabilidad de la taxonomía para enfoques de creación de aplicaciones móviles sensibles al contexto propuesta en [6] y ampliada en [7]. Además, se considera la bibliografía existente cómo también nuestra experticia en la temática. Acorde a esto, se definen las siguientes características de interés en relación a la variabilidad:

- *Cómo se realiza la expresividad de conceptos específicos*: para este tipo de aplicaciones es imprescindible poder detallar distintos tipos de conceptos, como por ejemplo, características de contexto o sensores. Acorde a esto, cada enfoque permitirá abordar con mayor o menor grado de flexibilidad la expresividad de conceptos. Esto permite poder expresar los factores de categorización, relevancia, combinación, tipo de configuración y tipo de ejecución mencionados en [6] y [7].
- *Cuál es el nivel de abstracción: para definir datos concretos*: es fundamental en algún momento de las etapas propuestas por los enfoques poder instanciar las clases definidas en los modelos, ya que esto es necesario para luego generar aplicaciones funcionales. Esto es útil, por ejemplo, para indicar factores de márgenes de precisión y exactitud indicados en [6] y [7].
- *Qué tipo de sensores se especifican*: por el tipo de aplicaciones que se desea generar es crítico poder usar sensores en un nivel alto de abstracción. Esto permite expresar los distintos posibles sensores, por ejemplo, los detallados en [12] y presentados en la Sección 2.a.
- *Cómo se realiza la derivación de modelos a aplicaciones móviles*: una característica fundamental es lograr generar aplicaciones funcionales, para esto, por el tipo de enfoque se debe pasar de alguna manera de modelos a códigos. Esto permite contar, por ejemplo, con variabilidad en tiempo de ejecución [12].
- *Cómo se realiza la integración de los sensores especificados en las aplicaciones generadas*: es importante que no solo se detallen los sensores a nivel de modelado, sino también que estos luego se integren en las aplicaciones derivadas. Esto guarda relación con los factores de tipo de configuración y tipo de ejecución mencionados en [6] y [7].

Las características presentadas anteriormente serán usadas para analizar los siguientes tres entornos que permiten el modelado basado en UML:

- *Eclipse combinado con JBoss,*
- *Eclipse combinado con Obeo Sirius y JBoss*
- *Visual Studio combinado con Apache Cordova*

Se busca con este análisis poder comprender cómo los mismos abordan la variabilidad a la hora de generar distintos tipos de aplicaciones. En la Tabla 1 se presenta el análisis realizado en relación a las características y los entornos descritos.

En la siguiente sección se presenta un espacio de discusión en relación a la Tabla 1 para poder así comprender la variabilidad soportada por cada uno de estos entornos en relación las aplicaciones que permiten generar.

TABLE I: Características de Entornos de Desarrollo que permiten el modelado basado en UML

	ENTORNO DE DESARROLLOS		
	ECLIPSE + JBOSS	***ECLIPSE + SIRIUS OBEO + JBOSS***	***VISUAL STUDIO + APACHE CORDOVA***
¿Cómo se realiza la expresividad de conceptos específicos?	Usando *Eclipse* se definen meta-modelos y estereotipos UML (los cuales permiten dar expresividad a las clases UML).	Usando *Eclipse* se definen meta-modelos. Desde *Sirius Obeo* se importan los meta modelos de *Eclipse*, se crean los modelos; y define el DSLsVisual (que permite mayor expresividad).	Usando *Visual Studio* se pueden definir meta-modelos pero no soporta estereotipos UML en las versiones nuevas de *Visual Studio*. Los diagramas UML en las versiones nuevas son como archivos XML.
¿Cuál es el nivel de abstracción para definir datos concretos?	La creación de instancias de clases es un proceso engorroso y limitado a pantallas donde se deben encontrar los slots de las instancias para asignarles valores. Un proceso poco práctico.	Usando *Sirius Obeo* se pueden crear DSLs Visuales, los cuales permiten fácilmente darle valores a instancias en base a estos DSLs	Usando *Visual Studio* se pueden crear DSLs Visuales, los cuales permiten fácilmente darle valores a instancias en base a estos DSLs

¿Qué tipo de sensores se especifican?	Se pueden usar los plugins de default del marketplace de *Eclipse* (Pantalla Táctil, Acelerómetro, Giroscopio, Brújula, Cámara, y GPS) que son compatibles con *Jboss* Central. Es decir son sensores físicos acorde a la definición de [12]. Pueden crearse plugins para representar los conceptos de sensores virtuales [12].	Se pueden usar los plugins de default del marketplace de *Eclipse* (Pantalla Táctil, Acelerómetro, Giroscopio, Brújula, Cámara, y GPS) que son compatibles con Jboss Central. Es decir son sensores físicos acorde a la definición de [12]. Pueden crearse plugins para representar los conceptos de sensores virtuales [12].	Se pueden usar los sensores como GPS, acelerómetro Es complejo añadir plugins de acuerdo a la necesidad; ya que es un sistema cerrado.
¿Cómo se realiza la derivación de modelos a aplicaciones móviles?	Requiere de modelos intermedios, para esto se utiliza GMF para luego generar un APK.	Requiere de modelos intermedios, para esto se utiliza ATL y OCL para generar XML y luego generar un APK. Existen plantillas que son personalizables, mediante la utilización de código de programación, HTML, JavaScript, CSS.	Requiere modelos intermedios que son autogenerados, difícilmente personalizables; ya que trabaja con plantillas predefinidas.
¿Cómo se realiza la integración de los sensores especificados en las aplicaciones generadas?	Plugins que permiten empaquetar datos cómo: indicador de red, acelerómetro, brújula, cámara y geolocalización; acorde a la definición de modelos para empaquetarlos en la aplicación derivada.	Plugins que permiten empaquetar datos cómo: indicador de red, acelerómetro, brújula, cámara y geolocalización; acorde a la definición de modelos para empaquetarlos en la aplicación derivada. Se pueden escribir programas para representar sensores virtuales, pero no está la posibilidad de empaquetarlos actualmente.	Plugins que permiten empaquetar datos cómo: indicador de red, acelerómetro, brújula, cámara y geolocalización; acorde a la definición de modelos para empaquetarlos en la aplicación derivada. Los plugins no pueden ser extendidos.

IV. DISCUSIÓN

En ésta sección se presenta un espacio de discusión tomando de base la Tabla 1 presentada en la sección anterior con el fin comprender la variabilidad soportada por cada uno de los entornos de desarrollo analizados en relación las aplicaciones que permiten generar.

Según [6] y [7] los enfoques de creación de aplicaciones móviles sensibles al contexto deben poder soportar variabilidad como un factor primordial; eso hace que estos enfoques sean flexibles, logrando así generar una amplia gama de aplicaciones. Considerando esto analizaremos los tres enfoques de desarrollo analizados en la Sección 3.

En [7] se describe que para este tipo de enfoques es fundamental poder expresar conceptos que son relevantes, y que luego permitirán generar las aplicaciones en cuestión, por ejemplo, que clases representan contextos o sensores. En relación a esta temática, en la Tabla 1 se puede apreciar que la mayor expresividad se alcanza con *ECLIPSE combinado con Sirius Obeo y Jboss*, ya que usa DSL's visuales agilizando la tarea de especificar conceptos. *ECLIPSE combinado Jboss* permite expresividad pero es engorrosa su definición ya que hay que definir estereotipos; mientras que *Visual Studio* a partir de la Versión 2017, carece de expresividad al no soportar el uso de estereotipos, ya que, los diagramas UML, en las versiones nuevas son como archivos XML. La expresividad permite poder representar información relacionada a los factores de categorización, relevancia, combinación, tipo de configuración y tipo de ejecución mencionados en [6] y [7]. Al poder indicar esto, luego se pueden generar diferentes tipos de aplicaciones.

Dado que se busca poder generar aplicaciones concretas es fundamental poder definir datos concretos para que estos estén dentro de estas aplicaciones; acode a esto, *Sirius Obeo* agiliza esta tarea ya que permite hacerlo usando los DSL`s visuales que permiten definir instancias con datos concretos. Esto también es provisto por *Visual Studio*. Esta característica permite indicar además información en relación a los márgenes de precisión y exactitud indicados en [6] y [7]; lo cual enriquece las aplicaciones generadas.

Poder tener variabilidad en los sensores que se pueden definir es fundamental. Los sensores que son compatibles con *Jboss* son sensores clasificados como físicos por [12]. También *Jboss* permite crear plugins con lo cual se podrían representar, por ejemplo, sensores virtuales u otros sensores indicados en [12] y presentados en la Sección 2.a. En cuanto a *Visual Studio* se pueden usar algunos sensores físicos pero es muy complejo agregar plugins para representar sensores virtuales.

Como se puede observar en la Tabla 1, la generación de aplicaciones móviles requiere en todos los casos de modelos intermedios para los tres entornos desarrollados. La ventaja de usar plantillas (en el caso de *ECLIPSE combinado con Sirius Obeo y Jboss*) es la rapidez para generar aplicaciones; sin embargo, no hay mucha documentación al respecto. En cuanto a *Visual Studio* es complejo personalizar las plantillas predefinidas. En este caso las plantillas permiten variabilidad en la gama de aplicaciones que se pueden generar.

En la Tabla 1 se puede apreciar que todos los entornos permiten a través del uso de plugins el empaquetado de aplicaciones junto a los sensores. Con *ECLIPSE combinado*

con Sirius Obeo y Jboss, se podrían desarrollar nuevos plugins para representar, por ejemplo, sensores virtuales y estos empaquetarse en las aplicaciones generadas.

La variabilidad en enfoques de creación está relacionado a generar una variada gama de aplicaciones móviles sensibles al contexto como se menciona en [6] y [7]. Para esto es fundamental, por ejemplo, representar una amplia gama de conceptos, definir valores concretos, soportar una variada gama de sensores y contar con la posibilidad de extensión. Este último punto permite poder pensar en agilizar algunas tareas que tienen los desarrolladores de aplicaciones móviles sensibles al contexto así poder crearlas más ágilmente. Es decir, hay tareas repetitivas o conceptos que existen en todas estas aplicaciones que se podrían brindar dentro del entorno. En [7] se describen conceptos comunes que podrían ser brindados en los DSL's visuales y así agilizar la tarea del desarrollador.

Acorde al análisis realizado *ECLIPSE combinado con Sirius Obeo y Jboss*, es la mejor opción para soportar variabilidad en las aplicaciones generadas, logrando mayor expresividad y además agiliza la instanciación mediante el uso de DSL's visuales. La posibilidad de personalizar o extender el entorno con plugins permite soportar distintos tipos de sensores, y poder lograr soportar en un futuro una más amplia gama de aplicaciones.

Cabe mencionar que *Visual Studio* no sería una buena opción si se optar por usar el estándar de UML como base del enfoque de creación, ya que actualmente el mismo no tiene soporte.

Es importante tener en cuenta que estos entornos ya existen y lo que se busca es ver cuáles de éstos podrían dar soporte a la generación de aplicaciones móviles sensibles al contexto. En el caso de diseñar desde una etapa inicial un enfoque es recomendable no solo considerar los conceptos analizados en este trabajo, sino también lo planteado en [6] y [7].

V. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se analizaron tres entornos de desarrollo para poder comprender cómo los mismos abordan la variabilidad a la hora de generar distintos tipos de aplicaciones. En particular, los entornos analizados permiten el modelado basado en UML.

Para el análisis realizado se tomaron de base los conceptos de variabilidad de la taxonomía para enfoques de creación de aplicaciones móviles sensibles al contexto propuesta en [6] y ampliada en [7]. Además, se consideró la bibliografía existente cómo también nuestra experticia en la temática. En base a esto, se definieron cinco características puntuales para analizar cada entorno.

El aporte de este trabajo es analizar la variabilidad de algunos entornos de desarrollo y cómo se podrían usar para la creación de aplicaciones móviles sensibles al contexto. En la sección de discusión se presentaron en profundidad cada ítem analizado en la Tabla 1, brindando así una mayor comprensión al lector sobre la temática que abarca cada uno de estos ítems.

Acorde a los entornos analizados, se concluye que la mejor opción de los tres entornos es *ECLIPSE combinado con Sirius Obeo y Jboss*, ya que soporta mayor variabilidad en las aplicaciones generadas con la posibilidad de extensión mediante plugins, tiene mayor expresividad y además agiliza la instanciación mediante el uso de DSL`s visuales.

Como trabajo futuro se extenderá el entorno de *ECLIPSE combinado con Sirius Obeo y Jboss* para poder proveer al desarrollador de conceptos comunes planteados en [7] para agilizar aún más la tarea; es decir, se propondrá un nuevo enfoque de creación, que cuente con características propias del DDM para agilizar la creación de este tipo de aplicaciones. Para este enfoque de creación que se creará se usara de base los conceptos de variabilidad presentados en [6] y [7], como así también los conceptos analizados en este trabajo.

REFERENCIAS

[1] U. Alegre and T. Clark, "Engineering context-aware, systems and applications: A survey", Journal of Systems and Software, pp. 55-83, 2016.

[2] C. Taconet, and Z. Kazi, "Building Context-Awareness Models for Mobile Applications", Journal of Digital Information Management, vol. 8, 2010.

[3] M. Brambilla, and Jordi. Cabot, "Model-Driven Software Engineering in Practice". Second Edition,Milano: Morgan & Claypool Publishers, 2017.

[4] S. Bales, "Build Android apps without Coding: Get started with Android apps using Thunkable-MIT app Inventor". New York: Independently published. ACM, 2018.

[5] M.Brambilla, and P. Fraternalli, "Interaction Flow Modeling Language", Morgan Kaufmann, Elsevier, 2015.

[6] E. Gómez-Torres, C. Challiol, and S.E. Gordillo, "Context-Aware Mobile Applications: Taxonomy of factors for building approaches", in IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), IEEE Press, 2018, pp. 1-4.

[7] E. Gómez-Torres, C. Challiol, and S.E. Gordillo, "Variability Features in Building Approaches for Context-aware Mobile Applications" in Tic.ec, 2019. In Press.

[8] C. Bauer, and A. Novotny, A, "A consolidated view of context for intelligent systems. Journal of Ambient Intelligence and Smart Environments", vol. 9, nº 4, pp. 377-393, 2017.

[9] A. Fortier, G. Rossi, S. Gordillo and C. Challiol, "Dealing with variability in context-aware mobile software", Journal of Systems and, pp. 915-936, 2010.

[10] J.Rumbaugh, "Lenguaje Unificado De Modelado Manual De Referencia", Second Edition. Addison-Wesley Iberoamerica, 2007

[11] K., Mens, R. Capilla, N. Cardozo, and B. Dumas, "A taxonomy of context-aware software variability approaches," in 15th International Conference on Modularity, ACM, pp. 119-124, 2016.

[12] A. Rivero-Rodriguez, P., Pileggi, and O.A. Nykänen, "Mobile context-aware systems: technologies, resources and applications". International Journal of Interactive Mobile Technologies, vol. 10, nº 2, pp 25-32, 2016.

[13] Eclipse EMF, https://www.eclipse.org/modeling/emf (last access: 10/09/2019)

[14] GMF Tooling, https://www.eclipse.org/gmf-tooling (last access: 10/09/2019)

[15] ATL https://www.eclipse.org/atl (last access: 10/09/2019)

[16] Sirius, https://www.obeodesigner.com/en/product/sirius (last access: 10/09/2019)

[17] Obeo Sirius, https://www.obeodesigner.com/en/product/sirius (last access: 10/09/2019)

[18] A.García, "Desarrollo de DSL Visuales con Sirius", https://galba.gitbooks.io/desarrollo-de-dsl-visuales-con-sirius/, 2018

[19] ATL ECLIPSE, https://www.eclipse.org/atl/documentation (last access: 10/09/2019)

[20] JBoss.org, https://docs.jboss.org/tools/4.1.x.Final/en/User_Guide/html/chap-Hybrid_Mobile_Tools_and_CordovaSim.html (last access: 10/09/2019)

[21] R. Developer, https://developers.redhat.com/products/cdk/overview (last access: 10/09/2019)

[22] JBoss.org, https://tinyurl.com/y5owlfjh (last access: 10/09/2019)

[23] A.Cordova, https://cordova.apache.org/plugins (last access: 10/09/2019)

[24] Microsoft, https://tinyurl.com/yxc9a8ak (last access: 10/09/2019)

[25] Microsoft Xamarin, https://visualstudio.microsoft.com/es/xamarin (last access: 10/09/2019)

[26] R. Iskandar, "Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android", 2013. http://hdl.handle.net/2099.1/18249 (last access: 23/10/2019)

[27] J. Xiaoping, "A Performance Evaluation of Cross-Platform Mobile Application Development Approaches", in 5th International Conference on Mobile Software Engineering and Systems, 2018.

[28] S, Xanthopoulos, "A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications", in 6th Balkan Conference in Informatics, pp. 213-220, 2013.