Programmer Experience: A systematic Mapping

J. Morales, C. Rusu, and D. Quiñones

***Abstract*—User eXperience (UX) identifies the perceptions of people over of the use (or even the anticipated use) of a product, system or service. The programmers are users of specific systems, and several types of software development artifacts, such as programming environments and design documents. We performed a systematic mapping about Programmer eXperience (PX). In this work we consider the programmers as a particular type of users of particular artifacts. We therefore consider PX as a particular type of UX. The literature usually focuses on PX from a Software Engineering point of view. We analyzed PX mainly from a Human-Computer Interaction (HCI) perspective. We reviewed articles about PX, empirical studies related to aspects of PX, and studies addressing PX on programming environments. The results show that there is an interest on the PX, but the concept is not yet clearly defined. We found 40 articles published in the last ten years and established that the studies address usability and PX aspects focusing on four topics: (i) programming languages, (ii) programmers' interaction with the integrated development environment, (iii) application programming interfaces, and (iv) articles about programmers' behavior. It is a relatively small numbers of articles, compared with other Software Engineering or HCI areas. This represents a research opportunity for this systematic review and others that can be performed.**

***Index Terms*—Programmer experience, Programming environments, Systematic mapping, Usability, User experience.**

I. Introducción

La experiencia del usuario (de sus siglas en inglés User eXperience, UX), aborda las percepciones de las personas en el contexto de uso de un producto, sistema o servicio [1]. UX está compuesta por diferentes aspectos, uno de ellos es la usabilidad, la cual es relevante para los sistemas.

Durante el proceso de desarrollo de software, el programador se convierte en usuario de distintos sistemas y documentos. En consideración a este usuario particular hemos decidido realizar un mapeo sistemático sobre la experiencia del programador (de sus siglas en inglés Programmer eXperience, PX) y estudios actuales relacionados.

Diversas investigaciones encontradas muestran interés en PX, por ejemplo, artículos sobre usabilidad tanto para entornos de programación [2] como para lenguajes de programación [3], además de otros centrados en la dificultad a la que se enfrentan los programadores al mantener códigos [4].

J. Morales trabaja en la Facultad de Ingeniería de la Universidad Autónoma de Chile, Chile, (jmoralesb@uautonoma.cl).

C. Rusu trabaja en la Escuela de Ingeniería Informática de la Pontificia Universidad Católica de Valparaíso, Chile (cristian.rusu@pucv.cl).

D. Quiñones trabaja en la Escuela de Ingeniería Informática de la Pontificia Universidad Católica de Valparaíso, Chile (daniela.quinones@pucv.cl).

El presente artículo expone un mapeo sistemático en el cual se han recopilado y analizado estudios científicos sobre PX. Los resultados obtenidos muestran que existe un interés en las actividades propias del programador, como son la usabilidad en Entornos de Desarrollo Integrados (de sus siglas en inglés IDEs), Interfaz de Programación de Aplicaciones (de sus siglas en inglés APIs), y lenguajes de programación. Además de lo anterior, los resultados muestran que existe interés en evaluar la eficiencia y productividad de un programador, estableciendo varios estudios sobre su comportamiento.

El presente documento está estructurado de la siguiente manera: la sección II presenta los antecedentes y conceptos previos relacionados al estudio, la sección III detalla la metodología de trabajo, la sección IV muestra los resultados del mapeo sistemático, y finalmente la sección V presenta las conclusiones e ideas de trabajos futuros.

II. Antecedentes y Conceptos Previos

A. Usabilidad

La usabilidad es definida por la Organización Internacional de Estandarización (de sus siglas en inglés ISO) 9241-11 como "El grado en que un sistema, producto o servicio puede ser usado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico" [5]. La usabilidad es uno de los aspectos que hemos decidido incluir en esta revisión puesto que es un aspecto evaluable para los entornos de programación.

La usabilidad tiene atributos que la componen, como son: (1) Facilidad de aprendizaje: que corresponde a la facilidad con la cual un usuario puede aprender a utilizar un sistema, elemento muy importante para usuarios novatos; (2) Eficiencia: relacionada a la rapidez y cantidad de pasos requeridos para cumplir un objetivo; (3) Facilidad de memorización: facilidad con que usuarios esporádicos o poco frecuentes pueden recordar la forma en que el sistema se usa; (4) Errores: relacionado al número de errores que un usuario comente cuando realiza una tarea (un buen indicador de este atributo es un bajo número de errores); y (5) Satisfacción Subjetiva: que corresponde a la impresión subjetiva que tiene un usuario de un sistema [6].

B. Experiencia del Usuario

La ISO 9241-210 define la UX como "la percepción y respuestas de los usuarios como resultado del uso o uso anticipado de un sistema, producto o servicio" [1]. Existen también otras definiciones para UX, en ellas se pueden encontrar aspectos comunes como son el uso de productos interactivos por parte de las personas, percepciones y sentimientos involucrados durante la interacción [7].

La UX es propia de cada persona, pues depende de las vivencias previas. A su vez, no solo aborda experiencias

individuales, sino también aquellas grupales o compartidas. Por otro lado, la experiencia también puede construirse en diferentes periodos de tiempo, siendo éstas esporádicas o como resultado de una acumulación de ellas [8].

Para explicar la UX con mayor profundidad, Peter Morville en 2004 crea el modelo panal (*honeycomb*), el cual considera un diseño detallado de los aspectos que la componen [9]. Estos son:

- Usable: El sistema producto o servicio debe ser simple y fácil de usar.
- Útil: El producto o servicio debe ser útil y satisfacer una necesidad, de otra manera no se justifica su creación.
- Deseable: La estética visual del producto, servicio o sistema debe ser atractiva y fácil de interpretar.
- Encontrable: La información debe ser fácil de encontrar y debe tener una fácil navegación.
- Accesible: El producto o servicio debe ser diseñado de tal manera que sea accesible para todos los usuarios, o usuarios con diferentes capacidades o discapacidades.
- Valioso: El producto debe agregar valor.
- Creíble: La compañía y sus productos o servicios deben ser de confianza.

C. Experiencia del Programador

El programador posee un rol importante dentro del proceso de desarrollo de software, principalmente realiza las funciones de codificación. El programador debe interactuar y hacer uso de diversos elementos, como son entornos de programación y artefactos de desarrollo. Este usuario programador, al igual que el usuario normal, tiene diversas respuestas y percepciones sobre los productos, sistemas o servicios que utiliza al realizar su trabajo. Si bien existen estudios sobre el programador y el desempeño de sus tareas [10] [11], en nuestra revisión no hemos encontrado una definición explícita de PX. Debido a esto, nosotros proponemos la siguiente definición para la experiencia del programador con base en otros estudios realizados: "PX es el resultado de la motivación intrínseca y percepción de los programadores sobre el uso de artefactos de desarrollo" [12]. Sin duda, PX es un espacio abierto para esta investigación y otras que se realicen con posterioridad.

D. Entornos de Programación Integrado e Interfaz de Programación de Aplicaciones

Un Entorno de Desarrollo Integrado (de sus siglas en inglés IDE) comúnmente contiene un editor de texto, depurador, funciones de autocompletado de código, herramientas de construcción automática, intérprete o compilador (en algunos casos), entre otros elementos. Estas herramientas facilitan la tarea del programador y lo ayudan a establecer con mayor rapidez las fallas que puedan existir, por ejemplo, al compilar un programa [13]. Un ejemplo de IDE es Eclipse, el cual es mundialmente conocido [14].

La Interfaz de Programación de Aplicaciones (de sus siglas en inglés API), es una interfaz de un componente que es agregado en un proyecto de software. Esta interfaz posee características, las cuales pueden ser llamadas o invocadas para ser utilizadas por otro software. Específicamente "la combinación de las características de la interfaz y una descripción de su protocolo de uso y semántica es lo que se denomina API" [15].

III. Metodología de Trabajo

El mapeo sistemático fue realizado considerando cinco etapas las cuales se detallan a continuación.

A. Definición de las Preguntas de Investigación

Consideramos que las preguntas de investigación debían abordar conceptos generales propiciando así la recolección de trabajos relacionados. Las preguntas de investigación que se definieron fueron:

1) *¿Qué tipos de estudios existen sobre la experiencia del programador?*
2) *Si existen estos estudios, ¿Qué aspectos de la experiencia del programador abordan estos estudios?*
3) *¿Hay estudios sobre la experiencia del programador en entornos de programación*?

B. Fuentes de Datos y Estrategias de Búsqueda

La búsqueda de los artículos se realizó utilizando cinco cadenas de búsqueda detalladas en la Tabla I.

Las fuentes de datos usadas fueron: ACM Digital Library, IEEE Xplore: digital library, Springer Link, Science Direct, Scopus, y Google Scholar. Los criterios de búsquedas en las bases de datos fueron aplicados con la opción de texto completo, de tal manera de cubrir la mayor cantidad de artículos y no dejar restringidos solo al uso de palabras clave o título y resumen (*abstract*). Además, para mejorar la asertividad de los resultados, se aplicó un filtro que consistió en quitar las palabras "robot" y "aprendizaje", pues ambas no son parte del foco de esta investigación.

C. Selección de Artículos: Criterios de Exclusión

Después de la búsqueda, y dada la gran cantidad de resultados obtenidos en Google Scholar (el volumen de ítems obtenido superaba los miles en las tres primeras cadenas de búsqueda), se decidió desestimar estos resultados y seguir trabajando con las otras bases de datos.

Para continuar, se incorporó una selección basada en la antigüedad de los artículos, considerando sólo aquellos que no superaran los 10 años. En la Tabla I se puede observar cada una de las cadenas de búsqueda y los resultados obtenidos.

TABLA I
Resultados Iniciales de la Búsqueda

Cadenas de Búsqueda	Resultados obtenidos	**<10** años
"usability" and "programming environments"	35	16
"user experience" and "programming environments"	468	165
"programmer experience"	497	188
"user experience" and "programmer experience"	28	17
"programmer experience" and "programming environments"	77	28
Total de Artículos	1105	414

Además, los artículos encontrados fueron consolidados eliminando los duplicados. Seguidamente, una selección de artículos fue realizada considerando el título y resumen (*abstract*), obteniéndose un total de 82 documentos científicos para lectura completa.

Durante la lectura de los 82 artículos, 45 de ellos fueron excluidos dado que se centraban en otros aspectos no considerados en este estudio. A continuación, se detallan las temáticas abordadas en los documentos excluidos:

1) *Inteligencia artificial*
   Artículos científicos con fuertes orientaciones en inteligencia artificial y el desarrollo de la misma.
2) *Seguridad de la información y redes*
   Estudios basados en tecnologías de seguridad de la información y redes de comunicaciones.
3) *Otros tipos de programadores*
   Estudios que abordan la programación para niños o personas jóvenes no programadores.
4) *Otros documentos científicos*
   Se descartó también a aquellos artículos que solo desarrollan herramientas sin relación con la experiencia del programador.

Finalmente, seleccionamos una cantidad de 37 artículos para analizar en la siguiente etapa. Además, agregamos 3 artículos encontrados en las referencias bibliográficas de estos 37 documentos científicos seleccionados, los cuales fueron considerados de interés para el mapeo sistemático. Finalmente, un total de 40 artículos fueron considerados para la siguiente fase de análisis. Ver Fig. 1.

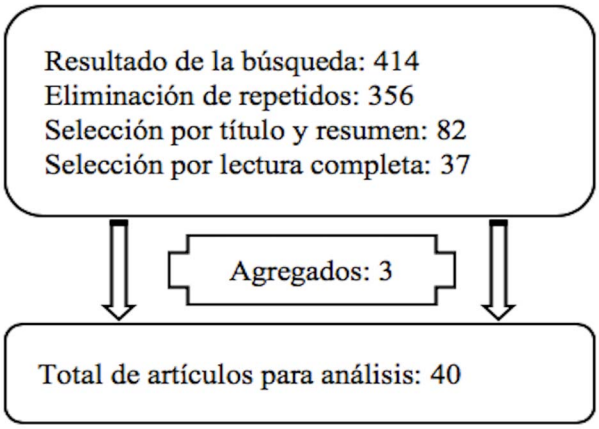

Fig. 1. Artículos para análisis.

D. Clasificación de Artículos

De entre los artículos seleccionados, 28 de ellos corresponden a conferencias entre el año 2009 y 2017, presentando una mayor frecuencia en los años 2016 y 2017, ambos con 6 artículos. La conferencia con más artículos publicados en nuestro tema de interés es la Conferencia Internacional de Ingeniería del Software (de sus siglas en inglés ICSE), con un total de 6 artículos, las cuales abordaron tópicos como complementos (*plug-ins*) para IDEs, usabilidad en las APIs y lenguajes de programación. Los artículos de revista son 9 en total, publicados entre el año 2012 y 2017, teniendo una mayor frecuencia en el año 2013 con 3 artículos. Por otro lado, solo dos pósteres y un capítulo de libro fueron considerados en el análisis. En la Fig. 2 se muestra el porcentaje de artículos por tipo de publicación (conferencias, revistas, pósteres y capítulo).

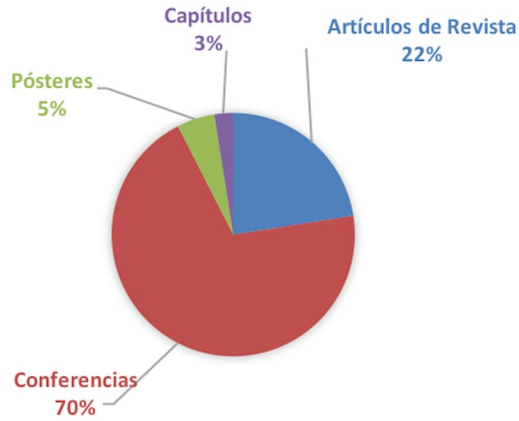

Fig. 2. Tipos de artículos.

Los artículos de revista y el capítulo analizado fueron publicados por: ELSEVIER 3 artículos y 1 capítulo, SPRINGER 2 artículos, IEEE 2 artículos, y finalmente ACM con 1 artículo. Claramente, ELSEVIER posee la mayoría de los artículos científicos analizados (ver Fig. 3).

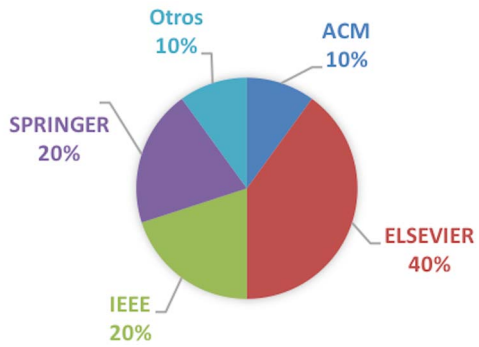

Fig. 3. Artículos según fuente de publicación.

Para clasificar los artículos según su contenido y en relación con las preguntas de investigación, se utilizaron las siguientes categorías: (1) *Lenguajes de programación:* Visuales, características y estudios de los lenguajes; (2) *IDEs:* Complementos, herramientas, IDEs específicos, técnicas de andamiaje y entornos restringidos; (3) *APIs:* Usabilidad en las APIs; y (4) *Sobre programadores:* Legibilidad de los códigos, comportamiento del programador y métricas de productividad.

El resultado del análisis de documentos encontrados se presenta en la siguiente subsección, considerando la clasificación ya indicada, así como también la particularidad del estudio en cuanto a la validación empírica que lleva a cabo.

E. Resultados de la Clasificación

En la Tabla II se pueden ver los resultados de la clasificación. A continuación, se analizará brevemente los documentos encontrados según la categoría correspondiente.

1) Lenguajes de programación

En la primera categoría de lenguajes de programación, se encontraron investigaciones sobre los beneficios de los lenguajes de programación y entornos gráficos. Se les considera como elementos positivos que pueden mejorar la legibilidad de

los códigos presentando ventajas sobre aquellos que son más bien textuales e impactando positivamente en el desarrollo de software [16][17]. Por otro lado, entornos de desarrollo que implementan aspectos gráficos pueden mejorar y facilitar variadas tareas del programador, como son la búsqueda de información en los entornos de programación [18] y el monitoreo y entendimiento del código [19].

TABLA II
RESULTADOS DE LA CLASIFICACIÓN

Categoría	Con prueba de usuarios o inspección	Sin prueba de usuarios
Lenguajes		
Visuales	[16][17] [18] [19]	[20][21] [22]
Características	[23][24][25]	-
Estudio de los lenguajes	[26]	[27]
Total	8	4
IDEs		
Complementos	[28][29][32] [33]	[30]
Herramientas	[31][34]	-
IDE específicos	[2][36]	[35]
Técnicas de andamiaje	[37]	-
Entornos restringidos	[38]	-
Total	10	2
API		
Usabilidad en las API	[40][41][3] [42]	[39]
Total	4	1
Sobre Programadores		
Legibilidad	[43][44][4]	[45]
Comportamiento	[10][46] [47] [48][49][50]	-
Métricas de productividad	-	[11]
Total	9	2

Se pudieron encontrar también, entornos que no son los más adecuados para dar soporte al desarrollo con objetos tridimensionales, generándose así la necesidad de contar con entornos que puedan responder más adecuadamente a estos requerimientos [20]. En relación con la facilidad de uso de los entornos, se encontraron estudios sobre las implicaciones que tiene un entorno muy sobrecargado y entornos muy simples o carentes, estableciendo que ambos poseen inconvenientes [21]. Otros estudios manifiestan la necesidad de un entorno multi-táctil el cual aporta a la facilidad de uso [22].

En relación a artículos relativos a las características de los lenguajes, fueron encontrados por ejemplo: (1) La prevalencia del polimorfismo en dos tipos de lenguajes orientados a objeto, estableciendo deficiencias de los IDEs en términos de la administración de esas características [23]; y (2) Estudios sobre la implementación de las funciones lambda en lenguaje C++ y la relación de esas implementaciones con la experiencia del programador, demostrando que los novatos presentan mayores problemas para la implementación de estas funciones [24].

Destaca un estudio sobre los lenguajes de programación, independientemente del IDE, el sistema operativo u otras herramientas [25]. IDE para programar UI (*User Interface)* fue considerado, en relación con lo que permite llevar a cabo tareas de carga, edición, compilación, etc. y el lenguaje a máquina UI, como la interfaz que proporciona acceso a los recursos de la máquina como memoria, pantallas de dispositivos de almacenamiento. Los entornos de programación Alice, Scratch y Small Basic fueron evaluados en esa investigación mediante la metodología Diseño Estructurado de Interfaz de Usuario SUID (*Structured User Interface Design*), obteniéndose como resultado que ambos, IDE para programar y Lenguaje a máquina, en teoría pueden evaluarse de manera separada. Sin embargo, lo más interesante es la evaluación que se realiza en esta investigación de los lenguajes con sus características como si fuera una interfaz.

Con relación a los estudios de los lenguajes de programación, fueron analizados dos artículos. Uno propicia la introducción de métodos empíricos cuantitativos para evaluar construcciones de los lenguajes de programación [26]. El otro critica fuertemente los experimentos realizados en este tipo de investigación, haciendo reparo en los estudios que existen sobre los lenguajes de programación, estableciendo que muchos de ellos no cumplen con la validez experimental necesaria, considerándose hasta las anécdotas como información valida de respaldo [27].

Sin embargo, aunque el artículo [27] nota esta falta de rigurosidad en la experimentación, es importante comentar que, en nuestra revisión de literatura (la cual no solamente aborda lenguajes de programación) existe un 77,5% de artículos analizados que presentan validez empírica en sus estudios.

2) Entornos de desarrollo integrados (IDEs)

La revisión de literatura mostró una tendencia interesante relacionada a los IDEs, identificando: (1) La incorporación de complementos para facilitar la edición y cambios en los códigos [28][29]; (2) La incorporación de aspectos sociales y los beneficios que esto trae para los programadores [30][31]; (3) El mejoramiento de aspectos específicos de depuración que los IDEs no cubren hoy en día [32]; y (4) La incorporación de usuarios que puedan definir experimentos de usabilidad integrados en los entornos [33]. En adición a lo anterior, también se analizó un artículo que implementa una herramienta específica para IDE con el fin de hacer más fácil la programación y minimizar los errores [34].

Para la subcategoría de IDE específicos, fueron encontrados artículos orientados a establecer características deseables de los IDEs [2], así como también que permitan el trabajo de programación en vivo, incorporando tanto a programadores como a diseñadores [35], junto al desarrollo en entornos en la nube y los beneficios que esto tiene [36]. Por otro lado, se analizaron estudios que realizaron ajustes a los IDE de forma tal que favorecen la programación en entornos con diversas restricciones, como se muestra en [37] [38].

3) Interfaz de programación de aplicaciones (APIs)

Las APIs en general, han sido abordadas en las investigaciones desde varios puntos de vista, lo que parece razonable dado que es uno de los elementos más cotidianos en el trabajo de los programadores.

Se seleccionaron algunos artículos dedicados a estudiar la usabilidad de las APIs de variadas formas como son: (1) Mediante métodos de inspección [39]; (2) En implementaciones de clase y métodos de programación [40]; (3) En investigaciones sobre la relevancia de la documentación en relación con la facilidad de uso [41]; y (4) En investigaciones sobre la facilidad de uso e implementación, mostrando variados errores en los manuales, lo que indica que su uso no es tan sencillo como se esperaba [3].

Un aspecto importante de las APIs se basa en proveer una comunicación fácil, lo que favorecería y minimizaría los errores que se pueden generar en el desarrollo [42].

4) Sobre programadores

Dado el contexto de la experiencia del programador, fueron encontradas investigaciones relacionadas a las tareas que él realiza y que contemplan variadas dificultades, como es el caso del mantenimiento del software, que requiere una lectura y análisis de códigos ya programados. Es así como surgen estudios relevantes relacionados con el mantenimiento de software, específicamente aquellos que facilitan la lectura de códigos [43], como pueden ser las instrucciones más predecibles por el lector [44], y cómo los códigos pueden implementarse de forma más legible [4] [45].

El comportamiento del programador puede dar claras luces sobre lo que requiere en su trabajo. Astromskis y otros en 2017 [10], realizaron un extenso seguimiento a los programadores en un estudio de campo que arrojó conclusiones interesantes sobre las actividades llevadas a cabo por los desarrolladores. Una de ellas, por ejemplo, es la comunicación constante que ellos mantienen. Además, casi la mitad de las sesiones incluyen la ejecución del sistema en desarrollo, así como el uso de consultas en línea, e ir y venir a través de los depuradores que pueden afectar el rendimiento de los programadores, sugiriendo entonces las opciones de navegación [10]. Claramente, proporcionar una mejor comunicación para el programador con el mundo exterior y sus compañeros puede favorecer y acelerar el desarrollo al mejorar la coordinación entre el equipo (como se muestra en [46]), así como la incorporación temprana de dispositivos disponibles en la red que pueden ser útil en el desarrollo [47].

Otros estudios se han referido a la forma en que trabaja el programador, como lo son el seguimiento al uso y manejo de excepciones, obteniendo como resultado que, dependiendo de los tipos de proyectos, los programadores prefieren realizar su manejo de excepciones de forma personalizada [48]. Es interesante también lo planteado en [49], en este artículo, los autores analizan el esfuerzo cognitivo que debe realizar un programador para usar un middleware (software para intercambio de información entre aplicaciones distribuidas). Por otro lado, se analizó una investigación que evalúa empíricamente si la experiencia del programador influye en la calidad de los códigos que escribe [50].

Finalmente, se consideró un estudio sobre la productividad del programador, especificando nuevas métricas para medir la productividad tanto para programadores en pareja como para aquellos solitarios, aplicable a todo el ciclo de vida del desarrollo [11]. Aunque este último trabajo no presenta experimentación práctica, deja abierta la pregunta sobre la dificultad de medir el trabajo del programador.

IV. Resultados

Los resultados obtenidos muestran variadas tendencias en estudios relacionados a PX. Una de ellas está relacionada con la usabilidad de IDEs, APIs y lenguajes de programación. A su vez, también existe preocupación sobre la forma en que los programadores se desempeñan, encontrándose investigaciones relacionadas con los esfuerzos que ellos realizan para comprender los códigos en mantenimiento, y cómo se desenvuelven dentro del entorno de programación. Surge como una idea importante de mejora en los IDEs, el integrar la comunicación entre el programador y el exterior u otras personas relacionadas, como otros programadores, diseñadores o usuarios.

A. Respuestas a las Preguntas de Investigación

Dado los artículos analizados y para dar respuesta a las preguntas de investigación que se definieron para esta revisión de literatura, se puede indicar lo siguiente.

1) *¿Qué tipos de estudios existen sobre la experiencia del programador?*

Formalmente, el concepto de PX no es evidente en los documentos analizados. Sin embargo, los 40 estudios analizados se enfocan en trabajar la PX desde distintos puntos de vista, tales como: (1) del uso que hacen los programadores de los lenguajes de programación (y sus propias características); (2) la interacción y el uso de entornos de programación y códigos de programación; y (3) estudios relacionados al comportamiento del programador. Todo lo anterior, con el fin de estudiar y analizar las posibilidades de mejorar la experiencia de los programadores.

2) *Si existen estos estudios, ¿Qué aspectos de la experiencia del programador abordan estos estudios?*

Hay varios aspectos de la PX que se consideran en los artículos revisados, como son: (1) Aspectos de *usabilidad*, tanto para APIs, IDEs y lenguajes de programación; (2) El aspecto *útil* se observa en entornos de programación y las nuevas herramientas para IDEs; (3) El aspecto *deseable* se puede observar en el interés de mejorar la escritura y la lectura de códigos de tal manera que sean más legibles y, por lo tanto, más fáciles de mantener; y (4) El aspecto *encontrable* se trabaja en aquellos artículos que proponen nuevos elementos que el programador requiere y que se pueden incorporar al entorno, como es el caso de las búsquedas y la interacción social.

Otros aspectos no han sido evidenciados en los artículos analizados, sin embargo, la *credibilidad* es algo que está en todos los artículos implícitamente, dado que los programadores en general al elegir un entorno de programación y/o una API, en su trabajo están haciendo uso de un elemento confiable.

3) *¿Hay estudios sobre la experiencia del programador en entornos de programación*?

En base a los artículos revisados, es posible concluir que sí existen estudios sobre PX en entornos de programación. Varios estudios analizan entornos de programación, como aquellos que apuntan a agregar nuevas funcionalidades y herramientas para facilitar el trabajo del programador y mejorar su experiencia en el entorno, así como aquellas

adaptaciones para trabajar en IDEs modificados o específicos dependiendo de las restricciones que existen desde el punto de vista de los elementos de interacción, las tecnologías disponibles y los usuarios potenciales.

B. Desafíos Abiertos

En adición a los resultados que dieron respuesta a las preguntas de investigación, se realizó una búsqueda manual de artículos. En esta oportunidad la investigación tiene como objeto ampliar los resultados obtenidos con relación a la usabilidad en entornos de programación. Como resultado de lo anterior, se encontraron en total 6 artículos.

Se encontraron dos artículos relacionados con evaluaciones heurísticas (las cuales permiten evaluar la usabilidad de las interfaces de usuario) proponiendo nuevos conjuntos para dominios específicos. Estos artículos no fueron encontrados en el mapeo sistemático, dado que no utilizan los términos asociados a las búsquedas definidas. El primero de ellos [51] realiza una evaluación heurística sobre las características de los lenguajes de programación. El segundo [52] aborda los sistemas de programación para principiantes, contemplando un conjunto de heurísticas para evaluar tanto entornos de programación como lenguajes de programación. Ambos artículos plantean nuevos conjuntos de heurísticas para los dominios específicos, obteniendo que ellas entregan mejores resultados que el uso de heurísticas generales. Sin embargo, los conjuntos deben cumplir con características bien definidas para favorecer su utilización.

Se encontraron también artículos que buscan establecer herramientas para mejorar los entornos de programación y así mejorar también la experiencia del programador: (i) en [53] se propone una nueva forma de exploración de objetos, mejorando así el uso de los selectores de métodos de interpretación estricta, lo que tiene efectos positivos en programadores novatos; (ii) En [54] se presentan herramientas moldeables para dar soporte a la tarea de comprensión del código según el dominio que se esté trabajando, con el objetivo de minimizar el esfuerzo que hacen los programadores por entender un código; (iii) Los autores en [55] muestran un estudio para utilizar la fijación de la vista del programador para establecer el foco del teclado en un entorno de programación, y de esta manera disminuir la cantidad de errores y facilitar el trabajo del programador que generalmente utiliza varias pantallas a la vez; finalmente, (iv) en [56] los autores identifican la variedad de hilos de trabajo que sigue el programador y propone mejoras a los IDEs para dar mejor soporte a las múltiples tareas que realiza el programador. Además, establece que el desarrollo de software es una colaboración entre un programador y un IDE, por lo que le da una visión del trabajo del programador muy ligada a la herramienta.

Los artículos anteriormente mencionados plantean diversos desafíos los que se pueden desagregar en: (i) proponer un conjunto de heurísticas que permita evaluar la usabilidad/UX de los entornos de programación; y (ii) profundizar en el concepto de Programmer eXperience, abordando la usabilidad/UX en otros artefactos de desarrollo además del IDE, como son documentos de diseño, códigos de programación, lenguajes, entre otros, mediante el desarrollo de conjuntos heurísticos de dominio específico.

V. CONCLUSIONES

Los resultados obtenidos en el mapeo sistemático muestran que existe interés científico sobre PX, y aunque en esta revisión bibliográfica no se ha encontrado una definición propiamente tal sobre PX, nosotros hemos incorporado una definición propia, construida en otros estudios relacionados.

Dentro de los artículos analizados, se pudo establecer que existen variados estudios que abordan la PX, desde distintos puntos de vista como son: (1) aquellos enfocados en el uso de lenguajes de programación (12 de 40, es decir 30%); (2) usabilidad de entornos de programación (12 de 40, es decir, 30%); usabilidad de las APIs (5 de 40, es decir, 12,5%); y artículos propios sobre los programadores (11 de 40, es decir, 27,5%). Se logró establecer elementos de la experiencia del programador en los distintos artículos analizados: (1) *Usabilidad* para las tres primeras categorías creadas en el análisis de los documentos (Lenguajes, IDEs y APIs); (2) *Útil*, asociado a las nuevas propuestas de funcionalidades para los IDEs y las API; (3) *Deseable*, en artículos relacionados a códigos de programación; (4) *Encontrable* en artículos que proponen nuevos elementos de búsqueda e interacción para los entornos de programación. Se encontró que el 30% de los artículos seleccionados aborda la PX en los entornos de programación.

Estos aspectos encontrados, podrían ser evaluados mediante heurísticas específicas, desarrolladas para dominios específicos como en [57] en donde se plantean heurísticas basadas en directrices de usabilidad. De esta manera se puede establecer la experiencia del programador asociada a entornos de programación y otros artefactos de desarrollo.

Como trabajo futuro se espera desarrollar conjuntos de heurísticas específicas bajo una metodología formal, dado la importancia que esto tiene para encontrar una mayor cantidad de problemas de usabilidad [58], que permitan evaluar la experiencia del programador (PX) en los entornos de programación y otros artefactos de desarrollo. Junto con esto, y considerando al desarrollador como concepto más amplio que el programador, es interesante ampliar el estudio hacia la experiencia del desarrollador, involucrando todo el ciclo de vida del desarrollo de software, manteniendo la orientación basada en usabilidad y UX de los artefactos que utiliza.

Además de lo anterior, nos gustaría ampliar nuestro trabajo abordando la programación para niños, personas jóvenes (novatos) y usuarios finales, puesto que es una temática emergente y relevante en diversos aspectos de la vida cotidiana, como son: la educación de la programación, los juguetes programables, las aplicaciones personalizadas, entre otros.

AGRADECIMIENTOS

Agradecemos a la Escuela de Ingeniería Informática de la Pontificia Universidad Católica de Valparaíso, Chile. Jenny Morales es beneficiaria de la beca doctoral INF-PUCV.

REFERENCIAS

[1] ISO 9241-11: Ergonomics of human-system interaction- Part 11: Related concepts and disciplines: Terms and definitions, International Standardization Organization (ISO), 2018.

[2] E. McArdle, J. Holdsworth, and I. Lee, “Assessing the usability of students object-oriented language with first-year IT students: A case study,” In *Proc. of the 25th Australian Computer-Human Interaction*

Conference: Augmentation, Application, Innovation, Collaboration, Adelaide, Australia, Nov., 2013, pp. 181-188.

[3] R. Gonçalves, M. Amaris, T. Okada, P. Bruel, and A. Goldman, "Openmp is not as easy as it appears," In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, Koloa, HI, USA, Jan., 2016, pp. 5742-5751.

[4] T. Sedano, "Code Readability Testing, an Empirical Study," In *Software Engineering Education and Training (CSEET), 2016 IEEE 29th International Conference on*, Dallas, TX, USA, Apr., 2016, pp. 111-117.

[5] ISO 9241-11: Ergonomics of human-system interaction- Part 11: Usability: Terms and definitions, International Standardization Organization (ISO), 2018.

[6] J. Nielsen, "Usability Engineering," AP Professional, 1993.

[7] Allaboutux, Definición de User Experience. [Online]. Disponible: http://www.allaboutux.org/ux-definitions. (Accedido 20 de marzo 2018).

[8] Dagstuhl seminar on Demarcating User Experience, User Experience White Paper, 2010. [Online]. Disponible: http://www.allaboutux.org/files/UX-WhitePaper.pdf. (Accedido 12 de marzo 2018).

[9] P. Morville, User experience honeycomb. [Online]. Disponible: http://semanticstudios.com/user_experience_design/. (Accedido 14 marzo 2018).

[10] S. Astromskis, G. Bavota, A. Janes, B. Russo, and M. Di Penta, "Patterns of developers behaviour: A 1000-hour industrial study," *Journal of Sys and Sw*, vol. 132, pp. 85-97, 2017.

[11] M. Solla, A. Patel, and C. Wills, "New metric for measuring programmer productivity," In *Computers & Informatics (ISCI), 2011 IEEE Symposium on*, Kuala Lumpur, Malaysia, Mar., 2011, pp. 177-182.

[12] J. Morales, C. Rusu, F. Botella, and D. Quiñones, "Programmer eXperience: A Systematic Literature Review," *IEEE Access,* vol. 7, pp. 71079-71094, 2019.

[13] M. Chicote and J. P. Galeotti, "TacoPlug: An Eclipse plug-in for TACO," *In 2012 Second Inter. Workshop on Developing Tools as Plug-Ins (TOPI)*, Zurich, Switzerland, Jun. 2012, pp. 37-42.

[14] Eclipse Fundation, Eclipse IDE, 2019. [online]. Disponible: https://www.eclipse.org/. (accedido el 8 de julio de 2019).

[15] M. Piccioni, C. A. Furia, and B. Meyer, "An empirical study of API usability,". In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Oct., 2013, pp. 5-14.

[16] N. Hollmann, and S. Hanenberg, "An Empirical Study on the Readability of Regular Expressions: Textual Versus Graphical," In *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*, Shanghai, China, Sept., 2017, pp. 74-84.

[17] Y. Zhang, S. Surisetty, and C. Scaffidi, "Assisting comprehension of animation programs through interactive code visualization," *Journal of Visual Lang. & Comp.*, vol. 24, no. 5, pp. 313-326, 2013.

[18] B. Athreya, and C. Scaffidi, "Towards aiding within-patch information foraging by end-user programmers," In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, Melbourne, VIC, Australia, Jul., 2014, pp. 13-20.

[19] T. Karrer, J. P. Krämer, J. Diehl, B. Hartmann, and J. Borchers, "Stacksplorer: call graph navigation helps increasing code maintenance efficiency," In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, Santa Barbara, CA, USA Oct., 2011, pp. 217-224.

[20] R. A. H. Acuna, C. Fidas, V. Argyriou, and S. A. Velastin, "Toward a Two-Handed Gesture-Based Visual 3D Interactive Object-Oriented Environment for Software Development," In *Intelligent Environments (IE), 2012 8th International Conference on*, Guanajuato, Mexico, Jun., 2012, pp. 359-362.

[21] E. Dillon, M. Anderson, and M. Brown, "Comparing feature sets within visual and command line environments and their effect on novice programming," In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, Raleigh, NC, USA, Feb., 2012, pp. 675-675.

[22] A. Bellucci, M. Romano, I. Aedo, and P. Díaz, "Software support for multitouch interaction: the end-user programming perspective," *IEEE Perv. Computing*, vol. 15, no. 1, pp. 78-86, 2016.

[23] N. Milojković, A. Caracciolo, M. F. Lungu, O. Nierstrasz, D. Röthlisberger, and R. Robbes, "Polymorphism in the spotlight: Studying its prevalence in Java and Smalltalk," In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, Florence, Italy, May, 2015, pp. 186-195.

[24] P. M. Uesbeck, A. Stefik, S. Hanenberg, J. Pedersen, and P. Daleiden, "An empirical study on the impact of C++ lambdas and programmer experience," In *Proceedings of the 38th International Conference on Software Engineering*, Austin, TX, USA, May, 2016, pp. 760-771.

[25] L. V. Morales Díaz, "Programming languages as user interfaces," In *Proceedings of the 3rd Mexican Workshop on Human Computer Interaction*, San Luis Potosí, SLP, México, Nov., 2010, pp. 68-76.

[26] S. Hanenberg, "Empirical, Human-Centered Evaluation of Programming and Programming Language Constructs: Controlled Experiments," In *International Summer School on Generative and Transformational Techniques in Software Engineering*, Aug., 2015, pp. 45-72.

[27] A. Stefik, and S. Hanenberg, "Methodological Irregularities in Programming-Language Research," *Computer*, vol. 50, no. 8, pp. 60-63, 2017.

[28] M. Vakilian, N. Chen, R. Z. Moghaddam, S. Negara, and R. E. Johnson, "A compositional paradigm of automating refactorings," In *European Conference on Object-Oriented Programming*, Berlin, Heidelberg, Jul., 2013, pp. 527-551.

[29] Y. Yoon, and B. A. Myers, "Supporting selective undo in a code editor," In *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, May, 2015, vol. 1, pp. 223-233.

[30] C. D. Hundhausen, and A. S. Carter, "Supporting Social Interactions and Awareness in Educational Programming Environments," In *Proceedings of the 5th Workshop on Evaluation and Usability of Programming Languages and Tools*, Portland, OR, USA, Oct., 2014, pp. 55-56.

[31] C. Bravo, R. Duque, J. Gallardo, "A groupware system to support collaborative programming: Design and experiences," *Journal of Sys. and Sw.*, vol. 86, no. 7, pp. 1759-1771, 2013.

[32] G. Salvaneschi, and M. Mezini, "Debugging for reactive programming," In *Proceedings of the 38th International Conference on Software Engineering*, Austin, TX, USA, May, 2016, pp. 796-807.

[33] S. R. Humayoun, Y. Dubinsky, and T. Catarci, "UEMan: A tool to manage user evaluation in development environments," In *Proceedings of the 31st International Conference on Software Engineering*, Vancouver, Canada, May, 2009, pp. 551-554.

[34] M. Coblenz, W. Nelson, J. Aldrich, B. Myers, and J. Sunshine, "Glacier: Transitive class immutability for Java," In *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on*, Buenos Aires, Argentina, May, 2017, pp. 496-506.

[35] S. Ghorashi, and C. Jensen, "Jimbo: a collaborative IDE with live preview," In *Cooperative and Human Aspects of Software Engineering (CHASE)*, 2016 IEEE/ACM, Austin, TX, USA, May, 2016, pp. 104-107.

[36] L. Wu, G. Liang, S. Kui, and Q. Wang, "CEclipse: An online IDE for programing in the cloud," In *Services (SERVICES), 2011 IEEE World Congress on*, Washington, DC, USA, Jul., 2011, pp. 45-52.

[37] C. Mbogo, E. Blake, and H. Suleman, "Design and use of static scaffolding techniques to support Java programming on a mobile phone," In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, Arequipa, Peru, Jul., 2016, pp. 314-319.

[38] A. Esakia, "Smartwatches For Junior/Senior Level CS Education," In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, Kansas City, MO, USA, Feb., 2015, pp. 691-691.

[39] U. Farooq, and D. Zirkler, "API peer reviews: a method for evaluating usability of application programming interfaces," In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, Savannah, GA, USA, Feb., 2010, pp. 207-210.

[40] T. Scheller, and E. Kuhn, "Influencing factors on the usability of api classes and methods," In *Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference and Workshops on*, Novi Sad, Serbia, Serbia, Apr., 2012, pp. 232-241.

[41] S. Endrikat, S. Hanenberg, R. Robbes, and A. Stefik, "How do api documentation and static typing affect api usability?," In *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India, May, 2014, pp. 632-642.

[42] J. A. Bastos, L. M. Afonso, and C. S. de Souza, "Metacommunication between programmers through an application programming interface: A semiotic analysis of date and time APIs," In *Visual Languages and Human-Centric Computing (VL/HCC), 2017 IEEE Symposium on*, Raleigh, NC, USA, Oct., 2017, pp. 213-221.

[43] T. R. Beelders, and J. P. L. du Plessis, "Syntax highlighting as an influencing factor when reading and comprehending source code," *Journal of Eye Mov*. Research, vol. 9, no. 1, 2015.
[44] A. Stefik, and S. Siebert, "An empirical investigation into programming language syntax," *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 4, pp.19, 2013.
[45] M. Kraeling, "Embedded Software Programming and Implementation Guidelines," In *Software Engineering for Embedded Systems*, 2013, pp. 183-204.
[46] J. Kubelka, "Artifact driven communication to improve program comprehension," In *Proceedings of the 39th International Conference on Software Engineering Companion*, Buenos Aires, Argentina, May, 2017, pp. 457-460.
[47] D. S. Makiyama, and P. T. Aquino, "Fundamentals of a components sharing network to accelerate JavaScript software development," In *Computer Science and Information Systems (FedCSIS)*, 2017 Federated Conference on, Sep., 2017, pp. 1303-1306.
[48] H. Osman, A. Chiş, C. Corrodi, M. Ghafari, and O. Nierstrasz, "Exception evolution in long-lived Java systems," In *Proceedings of the 14th International Conference on Mining Software Repositories*, Buenos Aires, Argentina, May, 2017, pp. 302-311.
[49] R. Maia, R. Cerqueira, C. S. de Souza, and T. Guisasola-Gorham, "A qualitative human-centric evaluation of flexibility in middleware implementations," *Empirical Sw. Eng*., vol. 17, no. 3, pp.166-199, 2012.
[50] O. Dieste et al., "Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study," *Emp. Software Eng.*, vol. 22, no. 5, pp. 2457-2542, 2017.
[51] C. Sadowski, and S. Kurniawan, "Heuristic evaluation of programming language features: two parallel programming case studies," In *Proceedings of the 3rd ACM SIGPLAN workshop on Evaluation and usability of programming languages and tools*, Portland, OR, USA, Oct., 2011, pp. 9-14.
[52] M. Kölling, and F. McKay, "Heuristic evaluation for novice programming systems," *ACM Transactions on Computing Education (TOCE)*, vol. 16, no 3, pp. 12, 2016.
[53] P. Rein, and R. Hirschfeld, "The exploration workspace: interleaving the implementation and use of plain objects in smalltalk," In *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming*, NY, USA, 2018, pp. 113-116.
[54] A. Chiş et al., "Exemplifying moldable development," In *Proceedings of the Programming Experience 2016 (PX/16) Workshop*, Rome, Italy, 2016, pp. 33-42.
[55] A. Thomschke, D. Stolpe, M. Taeumel, and R. Hirschfeld, "Towards Gaze Control in Programming Environments," In *Proceedings of the Programming Experience 2016 (PX/16) Workshop*, Rome, Italy, 2016, pp. 27-32.
[56] M. Hauswirth, and M. R. Azadmanesh, "The entangled strands of time in software development," In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Programming Experience*, Vancouver, BC, Canada, 2017, pp. 11-16.
[57] A. P. Proenca et al., "Systematic Review on Cognitive Engineering Applied to Critical Systems for Proposition of Evaluation Heuristics for Virtual Reality," *IEEE L. A. Trans.*, vol. 15, no 10, pp. 2024-2029, 2017.
[58] D. Quinones, C. Rusu, S. Roncagliolo, V. Rusu, and C. A. Collazos, "Developing usability heuristics: a formal or informal process?," *IEEE L. A. Trans.*, vol 14, no. 7, pp. 3400-3409, 2016.

Jenny Morales is a PhD candidate in Informatics Engineering at Pontificia Universidad Católica de Valparaíso (PUCV), in Chile. She has a Master in Science of Informatics Engineering from PUCV and a master's in Information Technology at Universidad Técnica Federico Santa María, in Chile.

She is professor at Universidad Autónoma de Chile in Chile. Her research interests focus on Human–Computer Interaction (HCI), User Experience and Computer Science Curricula.

She is an officer of the ACM SIGCHI Valparaiso chapter. She is also a member of the "UseCV" Research Group in HCI at PUCV, Chile.

Cristian Rusu is PhD in Applied Informatics from Technical University of Cluj-Napoca (Romania).

He is full professor at Pontificia Universidad Católica de Valparaíso (PUCV), in Chile. He worked in several software companies and research institutes in Romania and Chile. Since 1999 he has been fully dedicated to academia. He serves on editorial boards and conference program committees and disseminates Usability's importance, in Chile and Latin America. His research interests focus on Human–Computer Interaction (HCI), Usability, User eXperience, Customer eXperience, and Service Science.

Former Chair of the Chilean ACMSIGCHI Chapter, he is also the head of the "UseCV" Research Group in HCI at PUCV.

Daniela Quiñones is a PhD in Informatics Engineering at Pontificia Universidad Católica de Valparaíso (PUCV), in Chile. She has a master's in Science of Informatics Engineering from PUCV, Chile.

She is a professor at PUCV, Chile. Her research interests focus on Human–Computer Interaction (HCI), Usability Engineering, Usability, Customer eXperience, and User Experience.

She is member of the "UseCV" Research Group in HCI at PUCV, Chile. She is the chair of the ACM SIGCHI Valparaiso chapter.