Definición De Una Gramática Para El Léxico Extendido Del Lenguaje

Pablo Roldán Valdiviezo[1][0009-0003-8674-4835], Leandro Antonelli[23][0000-0003-1388-0337]

[1]Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[2]Lifia, Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[3] CAETI - Facultad de Tecnología Informática - Universidad Abierta Interamericana.
pablo.roldan@info.unlp.edu.ar
lanto@lifia.info.unlp.edu.ar

Resumen. En la ingeniería de requerimientos es importante comprender el lenguaje del dominio de la aplicación, puesto que escribir requerimientos sin los términos adecuados puede llevar a confusiones. El Léxico Extendido del Lenguaje (LEL) es un lenguaje de representación que permite capturar dicho vocabulario. Sin embargo, la elaboración de un LEL enfrenta desafíos, para quienes son nuevos en el uso de este concepto. Dado que el lenguaje natural es por sí ambiguo y, además, el LEL puede usarse de diferentes maneras, los usuarios pueden utilizar distintos estilos de redacción que pueden afectar la comprensión del lenguaje. En este contexto, este artículo propone una gramática EBNF (Gramática Extendida de Backus-Naur), para ayudar a estandarizar la redacción del LEL. Esta gramática tiene por objetivo tratar la homonimia, sinonimia, jerarquías, tiempos verbales, frases nominales, atomicidad y conexiones claras entre distintos tipos de símbolos. La gramática EBNF propuesta se ajusta a los principios de circularidad y vocabulario mínimo. Finalmente, el artículo describe los resultados de una evaluación preliminar que muestra la aplicabilidad de la propuesta. Los participantes hallaron la gramática clara y directa, lo que sugiere su potencial como guía para quienes se inician en su uso.

Palabras Claves: LEL. Requirements engineering. Grammar. EBNF.

1 Introducción

La etapa de ingeniería de requisitos desempeña un papel importante en el ciclo de vida del desarrollo de software, todo error que se presente en esta fase puede implicar altos costos para su corrección una vez que el producto final ha sido entregado [1]. Mientras que herramientas como Casos de Uso y las Historias de Usuario delinean de manera efectiva los objetivos y funcionalidades del sistema, se debe considerar que el software encapsula conocimiento “empaquetado sobre el dominio” [2]. Este conocimiento debe ser documentado en artefactos complementarios, tales como reglas de negocio [3] o escenarios específicos [4], que capturan la esencia del conocimiento

Proceedings of the 27th Workshop on Requirements Engineering (WER24), August 07-09, 2024, Buenos Aires, Argentina.
DOI: 10.29327/1407529.27-3

del dominio. A menudo, este conocimiento se dispersa entre un amplio grupo de partes interesadas, cada una aportando perspectivas únicas y complementarias al proyecto. Por lo tanto, se debe adoptar un enfoque colaborativo que permita reunir esta diversidad de conocimientos de manera efectiva [5].

El Léxico Extendido del Lenguaje (LEL) actúa como un glosario [3] que busca comprender el lenguaje de un dominio de aplicación específico sin enfocarse directamente en el software de la aplicación. Clasificando los términos en categorías como sujetos, objetos, verbos y estados, y utilizando atributos como la noción y las respuestas conductuales para describirlos, el LEL ofrece una forma estructurada de documentar el vocabulario relevante. Su facilidad de aprendizaje, uso y expresividad lo hacen adecuado para diversos dominios, incluso aquellos de complejidad considerable, con reportes de resultados positivos en ámbitos como el de la salud [6].

Los expertos en el dominio y el equipo de desarrollo hablan "idiomas" diferentes, lo que representa un desafío significativo [7]. Los primeros usan el lenguaje del dominio, mientras que los segundos emplean un lenguaje técnico informático. Para salvar esta brecha comunicativa, es fundamental el uso de artefactos en lenguaje natural accesibles para ambos grupos [5]. No obstante, la adopción del lenguaje del dominio por parte del equipo de desarrollo es compleja, dado que el dominio de aplicación suele ser más extenso que la aplicación de software propiamente dicha, lo que puede resultar abrumador y dificultar la delimitación de lo relevante para los límites de la aplicación de software.

En respuesta a los desafíos, este artículo propone una gramática en notación EBNF (Extended Backus–Naur Form) para estandarizar la creación del LEL. Basada en el análisis y diseño orientado a objetos y en prácticas reconocidas de especificación de requerimientos, la EBNF es conocida por definir la sintaxis de lenguajes de programación de manera formal y precisa. Esta gramática sistematiza las secuencias válidas de símbolos en el LEL, utilizando variables para unidades sintácticas y símbolos terminales, permitiendo una representación clara y estructurada de la sintaxis, y facilitando la generación de expresiones válidas a partir de un conjunto finito de reglas.

Al aplicar esta metodología al LEL, se organiza el vocabulario del dominio para mejorar la descripción de términos, gestionando homonimia, sinonimia y jerarquías, y alineándose con los principios de circularidad y vocabulario mínimo [9]. La gramática EBNF aplicada al LEL ofrece una especificación estructurada de software, facilitando una comunicación más eficiente entre desarrolladores. Una evaluación preliminar sugiere que la propuesta es práctica y útil como guía introductoria en ingeniería de requisitos para estandarizar el LEL, mejorando la comunicación entre expertos y desarrolladores.

2 Marco Conceptual

2.1 Léxico Extendido del Lenguaje

El Léxico Extendido del Lenguaje (LEL) es un glosario que describe el lenguaje de un dominio de aplicación sin centrarse en una aplicación de software específica. Basado en el concepto de "comprender el lenguaje de un problema sin preocuparse por el problema" [10], el LEL organiza este lenguaje mediante símbolos que representan términos o expresiones concisas del universo de discurso. Cada término en el LEL se define como un símbolo que incluye un nombre (o varios para cubrir sinónimos), una noción (la denotación del símbolo) y un impacto (la connotación del símbolo), explicados mediante oraciones que siguen dos principios importantes [9]: el principio de circularidad, que busca el uso intensivo de símbolos ya definidos en nuevas definiciones, y el principio de vocabulario mínimo, que limita el uso de términos ajenos al LEL en las descripciones de símbolos.

2.1.1 Descripción De Símbolos Del LEL [11]

El LEL organiza los elementos de un dominio de aplicación en cuatro categorías fundamentales: sujetos, objetos, verbos y estados. Este enfoque clasifica:

- Sujetos como entidades activas que realizan acciones, definidas por su noción (características o estado) e impactos (acciones realizadas).
- Objetos como entidades pasivas impactadas por sujetos, con noción refiriéndose a sus atributos e impactos a las acciones recibidas.
- Verbos representando acciones ejecutadas por sujetos hacia objetos, con una noción del objetivo y los impactos enumerando los pasos para la acción.
- Estados indicando condiciones de sujetos y objetos, con noción describiendo la situación y los impactos las acciones para cambiar dicho estado.

2.1.2 Reglas Para Describir Símbolos Del LEL [14]

Cada símbolo en el LEL debe tener un nombre del dominio, una oración en la noción que describa sus características y otra en el impacto que defina acciones asociadas. Los nombres deben ser representativos del dominio, agrupando sinónimos bajo un mismo término y excluyendo significados irrelevantes para mantener la especificidad del dominio. Es importante que cada símbolo se relacione con otros del LEL, manteniendo una estructura interconectada y coherente. La eliminación de un símbolo implica también la de sus estados asociados. Aplicar Principios de Circularidad y Vocabulario Mínimo.

2.2 Gramáticas

Las gramáticas son sistemas formales esenciales para estructurar lenguajes naturales y de programación, permitiendo la generación de expresiones válidas mediante la

clasificación de palabras y frases. Distinguen fenómenos lingüísticos como homonimia y polisemia, y abordan sinonimia, antonimia, hiponimia e hiperonimia. Noam Chomsky introdujo en 1957 las gramáticas de libre contexto, aplicables al inglés y al español, que simplifican la formación de oraciones mediante reglas específicas, aunque con limitaciones [16]. Estas gramáticas, al actuar como un metalenguaje, contribuyen al análisis del lenguaje en diversos campos, incluyendo lingüística computacional y desarrollo de software, resaltando su importancia en la comunicación y el reconocimiento de voz [15].

2.3 Notación EBNF (Extended Backus-Naur Form)

La EBNF (Extended Backus-Naur Form) [8] es una notación avanzada para describir la sintaxis de lenguajes de programación y otros sistemas formales. Es una extensión de la BNF, creada por John Backus y Peter Naur para el lenguaje ALGOL 60. La EBNF introduce mejoras significativas sobre la BNF, permitiendo una descripción más precisa y concisa de la sintaxis. La norma ISO/IEC 14977 [24] establece un estándar para la notación EBNF, asegurando la consistencia y claridad en la definición de lenguajes formales. Los símbolos utilizados en la EBNF son:

- ::= Define una regla de producción. []Indica que un símbolo es opcional. {} Denota repetición cero o más veces. | Señala una elección entre alternativas. <> Encierra símbolos no terminales. Concatenación: Combina cadenas de caracteres con símbolos no terminales. Símbolos terminales y no terminales: Elementos básicos del lenguaje y elementos definidos en términos de otros símbolos, respectivamente.

3 Contribución

Nuestra propuesta, una gramática EBNF diseñada para mejorar la documentación de términos LEL, se basa en las diez guías de Antonelli et al. [43], análisis y diseño orientado a objetos [31], y prácticas estándar de especificación de requerimientos. Este enfoque, enriquecido por el propósito detrás de cada acción inspirado en las historias de usuario, enfatiza la claridad y el objetivo de las interacciones. Incorporamos también la gestión de homonimia y sinonimia para asegurar descripciones precisas y coherentes. Fundamentada en conceptos clave derivados del análisis y diseño orientado a objetos propuesto por Wirfs-Brock [13], conformidad con el estándar IEEE 830 [29] y las adaptaciones de Kovitz [30] para describir acciones y roles, y la propuesta de Chelimsky [26] que especifica comportamientos a través de una máquina de estados alineada estrechamente con el patrón de estado [27]. Promovemos la simplicidad en impactos según IEEE 830 [29], recomendamos el uso explícito de auto-referencias para minimizar ambigüedades [30] [36], y evitamos auto-referencias y redundancias. Prohibimos frases débiles [34] para mantener objetividad,

utilizamos la relación "es un" aplicando el principio de sustitución de Liskov [33] para herencia y especialización, implementamos “desempeña el rol” mediante "Role Object Pattern" [25] permitiendo a símbolos sujetos adoptar distintos roles, y enfatizamos la relación “tiene un” para expresar composición [28], guiando hacia una estructuración coherente y modular del LEL. Describiremos cada gramática EBNF para los tipos de símbolos en secciones dedicadas: gramática para Sujeto en Sección 3.1, Objeto en 3.2, Verbo en 3.3, y Estado en 3.4, proporcionando una guía detallada para su aplicación en la ingeniería de requisitos.

3.1 Símbolo Tipo Sujeto

Para construir símbolos tipo sujeto utilizando la gramática EBNF (Figura 1), se debe seguir los siguientes pasos:

```
<DenominacionSujeto> ::= "Sujeto: " <NombreSujeto> [<Homonimos> "."] [<Sinonimos> "."] [<Jerarquia> "."]
    <NombreSujeto> ::= <SustantivoSingular> | <FraseNominalSingular>
    <Homonimos> ::= "Contexto: " <OracionSimple>
    <Sinonimos> ::= "Sinónimos: " <Sinonimo> {"," <Sinonimo>}
    <Jerarquia> ::= "Sub-tipo de: " <NombreSujetoSupertipo>

<NocionSujeto> ::= <OracionNocion> {"." <OracionNocion>} ["," <Estados> ","] ["," <Roles> ","]
    <OracionNocion> ::= {"Es" | "Está"} <OracionSimple>
    <Estados> ::= "Estados que posee: " <Estado> {"," <Estado>}
    <Roles> ::= "Roles que desempena: " <Rol> {"," <Rol>}

<ImpactoSujeto> ::= <OracionImpacto> {"." <OracionImpacto>}
    <OracionImpacto> ::= <NombreSujeto> <Verbo> [<Objeto>] {"para" <OracionSimple>}

<SustantivoSingular> ::= <Palabra>
<FraseNominalSingular> ::= <Palabra> {<Palabra>}
<OracionSimple> ::= <Palabra> {<Palabra>}
<Sinonimo> ::= <Palabra> /* Sinónimo del sujeto */
<NombreSujetoSupertipo> ::= <SustantivoSingular> | <FraseNominalSingular> /* Referencia a símbolo supertipo */
<Estado> ::= <Palabra> /* Posible referencia a símbolo tipo estado */
<Rol> ::= <Palabra> /* Posible referencia a símbolo */
<Verbo> ::= <Palabra> /* Posible referencia a símbolo verbo */
<Objeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Posible referencia a símbolo tipo sujeto */
<Palabra> ::= [a-zA-Z]+
```

Figura 1. Gramática EBNF para símbolo tipo Sujeto

1. Denominación: Iniciar seleccionando un nombre que refleje claramente el papel del sujeto en el dominio de aplicación, usando un sustantivo o frase nominal singular. Si existe homonimia, definir el contexto para aclarar el significado específico dentro del dominio.
2. Sinonimia: Identificar y listar sinónimos relevantes para el sujeto. Esto ayuda a agrupar términos con significados equivalentes bajo una única denominación.
3. Jerarquía: Si el sujeto es una especialización de otro, indicar esta relación jerárquica mencionando la denominación del símbolo general o padre. Esto refleja el principio de herencia de la programación orientada a objetos, donde el sujeto hereda características del símbolo padre y añade las propias.
4. Noción: Detallar las características o condiciones del sujeto a través de una o más oraciones simples que eviten la auto-referencia. Estas descripciones deben ser precisas, enfocándose en los aspectos relevantes del sujeto sin caer en redundancias.
5. Estados y Roles: Si el sujeto puede adoptar diferentes roles o estados, especificar estos mediante símbolos adicionales que detallen sus características y

comportamientos específicos. Esto enriquece la descripción del sujeto y su adaptabilidad dentro del sistema.

6. Impactos: Definir las acciones e interacciones del sujeto con otros elementos del LEL, siguiendo la estructura de: sujeto + (verbo en tercera persona del singular en tiempo presente indicativo) + [objeto opcional] + "para" propósito. Esto clarifica el rol del sujeto y las razones detrás de sus acciones. Si el verbo es transitivo debe estar acompañado de un objeto, sino es opcional. Los verbos que tienen significado trivial no se definen en el LEL. Se busca describir situaciones actuales y evitar caer en frases débiles que dan lugar a cierta subjetividad sobre la descripción.

Se proporciona en la Figura 2 un ejemplo de aplicación de esta gramática.

Denominación Sujeto
Sujeto: Director
Contexto: En un Liceo Militar
Sinónimos: Jefe, Líder
Tipo de: Personal Administrativo

Noción Sujeto
Es la figura de autoridad máxima dentro del Liceo Militar.
Está encargado de la dirección y gestión integral del liceo
Está encargado de la planificación y supervisión de todas las actividades académicas.
Está encargado de la planificación y supervisión de todas las actividades militares.
Estados que posee: En Actividad, Retirado
Roles que desempeña: Gestor, Supervisor

Impacto Sujeto
Director dirige las operaciones del liceo para cumplir con los objetivos educativos y militares.
Director supervisa las actividades académicas para asegurar la calidad de la educación y la formación.
Director supervisa las actividades militares para asegurar la calidad de la educación y la formación.
Director supervisa el Plan de Estudios para asegurar la calidad de la educación.

Figura 2. Ejemplo de símbolo tipo Sujeto

3.2 Símbolo Tipo Objeto

Para construir símbolos tipo objeto utilizando la gramática EBNF (Figura 3), se debe seguir los siguientes pasos:

```
<DenominacionObjeto> ::= "Objeto: " <NombreObjeto> [<Homonimos> "."] [<Sinonimos> "."] [<Jerarquia> "."]
    <NombreObjeto> ::= <SustantivoSingular> | <FraseNominalSingular>
    <Homonimos> ::= "Contexto: " <OracionSimple>
    <Jerarquia> ::= "Tipo de: " <DenominacionObjetoSupertipo>
    <Sinonimos> ::= "Sinónimos: " <Sinonimo> {"," <Sinonimo>}

<NocionObjeto> ::= <OracionNocion> { "." <OracionNocion>} ["." <Estados> "."]
    <OracionNocion> ::= ("Es" | "Está") <OracionSimple>
    <Estados> ::= "Estados que posee: " <Estado> { "," <Estado>}
    <Composicion> ::= "Tiene un/a: " <DenominacionSimboloObjetoContenido>

<ImpactoObjeto> ::= <OracionImpacto> { "." <OracionImpacto>}
    <OracionImpacto> ::= <NombreSujeto> <Verbo> <NombreObjeto> ["para" <OracionSimple>]

<SustantivoSingular> ::= <Palabra>
<FraseNominalSingular> ::= <Palabra> {<Palabra>}
<OracionSimple> ::= <Palabra> {<Palabra>}
<Sinonimo> ::= <Palabra> /* Sinónimo del objeto */
<DenominacionObjetoSupertipo> ::= <SustantivoSingular> | <FraseNominalSingular> /* Referencia a símbolo supertipo */
<DenominacionSimboloObjetoContenido> ::= <SustantivoSingular> | <FraseNominalSingular> /* Referencia a símbolo contenido */
<NombreSujeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Referencia a símbolo sujeto que ejecuta */
<Estado> ::= <Palabra> /* Posible referencia a símbolo tipo estado */
<Verbo> ::= <Palabra> /* Posible referencia a símbolo verbo */
<Palabra> ::= [a-zA-Z]+
```

Figura 3. Gramática EBNF para símbolo tipo Objeto

1. Denominación: Elegir un nombre representativo para el objeto, usando un sustantivo o frase nominal singular. Definir contexto para homónimos y listar sinónimos para agrupar términos equivalentes.
2. Jerarquía: Si el objeto es especialización de otro, indicar esta relación de herencia, mencionando el objeto general o padre. Esto refleja herencia en POO, permitiendo al objeto derivado añadir detalles específicos mientras hereda características del objeto padre.
3. Noción: Describir el objeto mediante oraciones claras y precisas, evitando auto-referencia y enfocándose en la relevancia para el dominio de aplicación. Pueden ser varias oraciones, cada una concentrándose en una única idea.
4. Estados: Detallar posibles estados del objeto, reflejando cambios o fases que puede experimentar.
5. Composición: Especificar cualquier objeto contenido como parte integral del objeto contenedor, utilizando "Tiene un/a", mostrando una relación de dependencia directa entre ambos. Esto alinea con la composición en POO, indicando que el objeto contenido no puede existir sin el contenedor.
6. Impactos: la acción definida en un impacto debe coincidir exactamente entre el símbolo sujeto que la ejecuta y el símbolo objeto que la recibe, asegurando que oraciones como 'El Cliente abre una Cuenta' se reflejan tanto en los impactos del símbolo 'Cliente' como en los del símbolo 'Cuenta'. Esta correspondencia directa entre los impactos fortalece la consistencia y facilita el seguimiento de las interacciones dentro del LEL. Utilizar la estructura de sujeto + (verbo en tercera persona del singular en tiempo presente indicativo) + [objeto actual] + "para" añadiendo opcionalmente el propósito detrás de cada acción para claridad.

Se proporciona en la Figura 4 un ejemplo de aplicación de esta gramática.

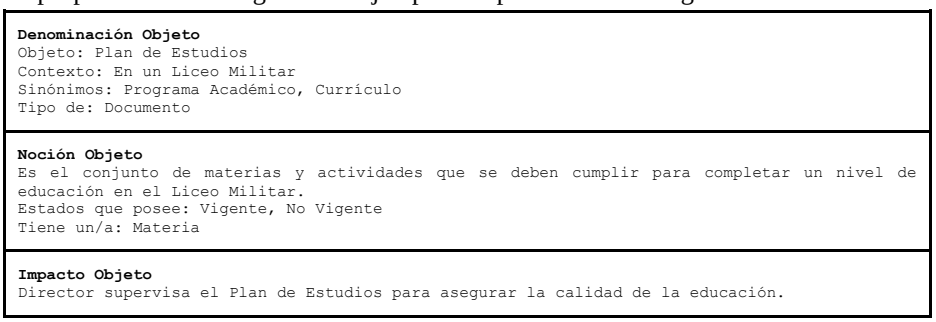

Figura 4. Ejemplo de símbolo tipo Objeto

3.3 Símbolo Tipo Verbo

Para construir símbolos tipo verbo utilizando la gramática EBNF (Figura 5), se debe seguir los siguientes pasos:

```
<DenominacionVerbo> ::= "Verbo: " <NombreVerbo> [<Homonimos> "."] [<Sinonimos> "."]
    <NombreVerbo> ::= <SustantivoSingular> | <FraseNominalSingular>
    <Homonimos> ::= "Contexto: " <OracionSimple>
    <Sinonimos> ::= "Sinónimos: " <Sinonimo> { "," <Sinonimo> }

<NocionVerbo> ::= "Objetivo/s: " <OracionSimple> { "." <OracionSimple> }

<ImpactoVerbo> ::= <OracionImpacto> { "." <OracionImpacto> }
    <OracionImpacto> ::= "Paso" <NumeroIncremental> ":" <Sujeto> <Verbo> <Objeto> ["para" <OracionSimple>]
    <NumeroIncremental> ::= [0-9]+

<SustantivoSingular> ::= <Palabra>
<FraseNominalSingular> ::= <Palabra> {<Palabra>}
<OracionSimple> ::= <Palabra> {<Palabra>}
<Sinonimo> ::= <Palabra> /* Sinónimo del verbo */
<Sujeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Posible referencia a símbolo sujeto */
<Verbo> ::= <Palabra> /* Posible referencia a símbolo verbo */
<Objeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Posible referencia a símbolo objeto */
<Palabra> ::= [a-zA-Z]+
```

Figura 5. Gramática EBNF para símbolo tipo Verbo

1. Denominación: Elegir un nombre representativo para el verbo, utilizando un verbo en infinitivo o frase nominal que refleje su acción dentro del dominio. Incluir contextos específicos (homónimos) y sinónimos para claridad.
2. Noción: Definir claramente el propósito o los objetivos del verbo con oraciones simples que destaquen la intención detrás de la acción. Permitir múltiples oraciones para explicar en detalle el propósito.
3. Impacto: Detallar las acciones necesarias, numeradas de manera incremental, para alcanzar el objetivo que símbolo verbo propone. Por ejemplo, si el verbo es "Procesar pago", los impactos podrían ser: "Paso 1: El cliente selecciona el método de pago. Paso 2: El sistema verifica la información de pago. Paso 3: El sistema confirma la transacción." Se orienta hacia la simplicidad en la estructura sujeto + verbo + [objeto opcional]. Además, se añade el propósito de cada acción, por ejemplo, “El Cliente abre una Cuenta para administrar sus finanzas”, integrando el objetivo detrás de la acción. En caso de requerirse información adicional, debe definirse alguno de los símbolos involucrados.

Se proporciona en la Figura 6 un ejemplo de aplicación de esta gramática.

```
Denominación Verbo
Verbo: Supervisar
Contexto: En un Liceo Militar
Sinónimos: Vigilar, Controlar

Noción Verbo
Objetivo/s: Asegurar que las actividades se realicen correctamente y de acuerdo a los
estándares establecidos.

Impacto Verbo
Paso 1: El Director revisa el Plan de Estudios para entender su estructura y contenido.
Paso 2: El Director evalúa el desempeño de los profesores en relación con el Plan de Estudios
para asegurar la calidad de la educación.
Paso 3: El Director evalúa el progreso de los estudiantes en relación con el Plan de Estudios
para asegurar la calidad de la educación.
```

Figura 6. Ejemplo de símbolo tipo Verbo

3.4 Símbolo Tipo Estado

Para construir símbolos tipo estado utilizando la gramática EBNF (Figura 7), se debe seguir los siguientes pasos:

```
<DenominacionEstado> ::= "Estado: " <NombreEstado> [<Homonimos> "."] [<Sinonimos> "."]
    <NombreEstado> ::= <Adjetivo> | <FraseAdjetival>
    <Homonimos> ::= "Contexto: " <OracionSimple>
    <Sinonimos> ::= "Sinónimos: " <Sinonimo> { "," <Sinonimo> }

<NocionEstado> ::= "Situación que representa: " <OracionSimple> { "." <OracionSimple> }

<ImpactoEstado> ::= <Transicion> { "." <Transicion> }
    <Transicion> ::= "Desde este estado, " <Sujeto> <Verbo> <Objeto> ", lleva a estado " <Estado>

<Adjetivo> ::= <Palabra>
<FraseAdjetival> ::= <Palabra> {<Palabra>}
<OracionSimple> ::= <Palabra> {<Palabra>}
<Sinonimo> ::= <Palabra> /* Sinónimo del estado */
<Sujeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Posible referencia a símbolo sujeto */
<Verbo> ::= <Palabra> /* Posible referencia a símbolo verbo */
<Objeto> ::= <SustantivoSingular> | <FraseNominalSingular> /* Posible referencia a símbolo objeto */
<Estado> ::= <Adjetivo> | <FraseAdjetival> /* Posible referencia a símbolo estado */
<Palabra> ::= [a-zA-Z]+
```

Figura 7. Gramática EBNF para símbolo tipo Estado

1. Denominación: Elegir un nombre representativo para el estado, utilizando un adjetivo o frase adjetival que refleje su acción dentro del dominio. Incluir contextos específicos (homónimos) y sinónimos para claridad
2. Noción: Articular qué representa este estado dentro del sistema, empleando oraciones simples para explicar la condición o fase del objeto o sujeto. Esta descripción debe ser clara y concisa, expresando ideas precisas y evitando excesos descriptivos.
3. Impactos: Detallar las transiciones posibles desde este estado a otros mediante acciones específicas (sujeto + verbo + [objeto opcional]). Estas transiciones deben describir claramente cómo las acciones afectan o modifican el estado, ofreciendo una visión de la dinámica del sistema análoga a una máquina de estados. Especifique tanto las condiciones de entrada como las de salida para cada estado, facilitando la visualización de las interacciones y transformaciones dentro del sistema.

Se proporciona en la Figura 8 un ejemplo de aplicación de esta gramática.

Denominación Estado
Estado: Presente
Contexto: En un Liceo Militar
Sinónimos: En el liceo, Activo

Noción Estado
Situación que representa: Indica que el cadete está físicamente en el liceo y participando activamente de las actividades programadas.

Impacto Estado
Desde este estado, el cadete se ausenta con causa justificada, lleva a estado Ausente con Causa.
Desde este estado, el cadete se ausenta sin causa justificada, lleva a estado Ausente sin Causa.

Figura 8. Ejemplo de símbolo tipo Estado

4 Evaluación

Para evaluar la gramática EBNF diseñada para el LEL, se empleó una herramienta colaborativa permitiendo a los participantes crear símbolos LEL individualmente en un entorno controlado, enfocándose en examinar su efectividad práctica más allá de la

usabilidad. El estudio apuntó a verificar la implementación práctica de la gramática, su aceptación entre usuarios, especialmente aquellos familiarizados con el desarrollo de software, pero nuevos en LEL y en el uso de gramática EBNF, y su capacidad para promover una interpretación uniforme de directrices teóricas, facilitando la estandarización en la descripción de símbolos del LEL para mejorar la coherencia y precisión en la documentación de requisitos.

La validación de nuestra propuesta de gramática EBNF para el Léxico Extendido del Lenguaje (LEL) fue llevada a cabo por siete profesionales con roles que incluía a programadores, testers, y jefes de proyecto con distintos niveles de experiencia, todos estaban familiarizados con el tema de dominio elegido “Régimen Disciplinario de Liceos Militares”, permitiéndoles concentrarse en la aplicación de la gramática sin distracciones de aprender un nuevo contexto.

A pesar de ser nuevos en el concepto de LEL y en la gramática EBNF, los participantes recibieron una introducción y materiales de apoyo para la descripción de símbolos del LEL utilizando la gramática propuesta. Tras una semana de trabajo y acceso a un experto para consultas, se observó una uniformidad en los estilos de redacción entre los participantes, señalando una correcta adherencia a la gramática. La evaluación final sobre la eficacia de la gramática en la validación de símbolos LEL permanece como tema de estudio futuro. Para evaluar la aplicabilidad del proceso [18][19], se utilizó el Sistema de Usabilidad Scale (SUS). Aunque él SUS se emplea principalmente para evaluar la usabilidad de sistemas de software, se ha demostrado su eficacia para evaluar productos y procesos [17]. El SUS consta de un cuestionario de 10 ítems, con una escala de cinco opciones que varía desde "1" (Totalmente en desacuerdo) hasta "5" (Totalmente de acuerdo). El cálculo del puntaje SUS se realiza sumando las puntuaciones de los ítems, ajustadas según su orden, y multiplicando el total por 2.5 para obtener un valor entre 0 y 100. El enfoque puede tener uno de los siguientes resultados: "No aceptable" 0-64, "Aceptable" 65-84 y "Excelente" 85-100 [20]. Según la puntuación obtenida, 72.01, el enfoque se considera "Aceptable" según la escala SUS.

5 Trabajos relacionados

Ruidías et al. [23] propone un marco ontológico para el LEL, integrando axiomas y heurísticas para mejorar la consistencia en las especificaciones de requerimientos y fomentar una mejor comunicación en la ingeniería de requisitos. Sebastián et al. [36] presenta una metodología para verificar el LEL usando mapas conceptuales, transformando nociones e impactos en conceptos y conexiones, lo que mejora significativamente la coherencia y completitud del LEL. Antonelli et al. [37] desarrolla un método para construir el LEL a partir de historias de usuario, documentando, normalizando y filtrando requisitos funcionales, permitiendo un crecimiento estructurado del LEL tras cada sprint. Centeno [38] propone mejorar la

precisión del LEL aplicando mediciones para optimizar la definición de requisitos, enfocándose en términos con alta carga semántica y eliminando redundancias, mejorando así la calidad y precisión de los símbolos.

Leite et al. [21] proponen un enfoque para convertir el conocimiento del mundo real en modelos conceptuales a través del LEL, destacando la creación de un léxico detallado que luego se transforma en modelos conceptuales. Este método se enfoca en estructurar el conocimiento del dominio mediante un léxico detallado. Leite et al. [39] utilizan una gramática EBNF para definir explícitamente los componentes de los escenarios, tales como título, objetivo, contexto, recursos, actores y episodios, en su trabajo sobre la mejora de la línea base de requisitos con escenarios. Este método proporciona una claridad y estructura significativas, reduciendo la ambigüedad en la documentación de requisitos. Lucassen et al. [40] exploran las historias de usuario en el desarrollo ágil, destacando la plantilla de Connextra: "Como un <rol de usuario>, quiero <objetivo/deseo>, para que <beneficio>", para comunicar necesidades de usuarios al equipo de desarrollo. La efectividad de esta plantilla, resaltada por Mike Cohn [41] y Lucassen et al., es reconocida por mejorar la productividad y alinear expectativas entre stakeholders y desarrolladores, estableciéndose como un estándar en la redacción de historias de usuario.

Litvak et al. [42] aborda una mejora significativa en las heurísticas para la creación de modelos en LEL, centrándose en la identificación y corrección de omisiones en la modelación de requisitos expresados en lenguaje natural. A través del análisis de nueve patrones de omisión detectados en modelos LEL existentes, el estudio ofrece heurísticas específicas para evitar o corregir estas omisiones, tales como la no detección de jerarquías, la incorrecta ubicación de impactos en jerarquías, y la falta de coherencia entre símbolos. Kaplan [22] mejora el Proceso de Requisitos basado en Escenarios, proponiendo ajustes para mejorar la calidad y consistencia de los modelos de requisitos, enfocándose en el LEL y Escenarios. Propone nuevas heurísticas y una metodología que aborda las diferencias entre la naturaleza declarativa del LEL y la procedural de los escenarios, mejorando la especificación mediante el manejo de jerarquías, homónimos, sinónimos, y estados de verbos, puntos de vista, así como vistas de empotramiento y clasificación.

El trabajo de Antonelli et al. [43] introduce diez guías para la especificación del LEL, las cuales han sido fundamentales para el desarrollo de una gramática EBNF en nuestra investigación. Estas guías, van desde la definición detallada de la noción y los impactos de sujetos, objetos, verbos y estados, hasta recomendaciones sobre cómo evitar ambigüedades y mantener la claridad en las descripciones, se basan en principios de la orientación a objetos y de la especificación de requerimientos. Esta gramática, en gran medida, es una implementación de la propuesta de Antonelli et al. [43].

6 Conclusiones y trabajos futuros

Este artículo presenta una adaptación de la gramática EBNF para mejorar la estructuración y redacción del LEL, buscando una especificación más precisa y coherente de términos de dominio en proyectos de software. La propuesta ofrece una metodología uniforme para describir requisitos, facilitando la comprensión y el éxito en la ingeniería de requisitos. La evaluación preliminar indica que la gramática es un paso hacia la estandarización del LEL, a pesar de los constantes desafíos en la evolución del lenguaje y los dominios de aplicación.

Futuras investigaciones se centrarán en adaptar la gramática EBNF a contextos específicos y explorar herramientas automáticas para actualizar LELs según avance el conocimiento del dominio. Una posible evolución incluiría una descripción en XML para mejorar la circularidad. Además, se propone desarrollar una herramienta de Inteligencia Artificial (IA) para asistir en la generación y corrección de símbolos del LEL, simplificando la estandarización y aumentando la eficiencia y accesibilidad en la especificación de requisitos.

Referencias

1. Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.
2. Brooks, F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition 1995.
3. Meservy, T. O., Zhang, C., Lee, E. T. and Dhaliwal, J.: "The Business Rules Approach and Its Effect on Software Testing," in IEEE Software, vol. 29, no. 4, doi: 10.1109/MS.2011.120, pp. 60-66, July-Aug, 2012.
4. Rose, S., Nagy, G.: Formulation: Document examples with Given/When/Then, Independently published, 979-8723395015, 2021.
5. Lim, S. L., Finkelstein, A.: “StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation”, IEEE transactions on software engineering, Volume 38, Issue 3, May-Jun 2012, DOI 10.1109/TSE.2011.36, pp 707-735, 2012
6. Cysneiros, L.M., Leite, J.C.S.P.: Using the Language Extended Lexicon to Sup- port Non-Functional Requirements Elicitation, in proceedings of the Workshops de Engenharia de Requisitos, Wer’01, Buenos Aires, Argentina, 2001.
7. Potts, C.: "Using schematic scenarios to understand user needs," in Proceedings of the 1st conference on Designing interactive systems: processes, practices, meth- ods, \& techniques, 1995
8. Scowen, R. S. (n.d.). Extended BNF | A generic base standard. University of Cambridge2.
9. LEITE, J.C.S.P.; DOORN, J.H.; KAPLAN, G.N.; HADAD, G.D.S; RIDAO, M.N. (2004). Defining System Context using Scenarios. En: LEITE, J.C.S.P.; DOORN, J.H. (eds.) Perspectives on Software Requirements, Kluwer Academic Publishers. Springer US.pp.169-199.

10. Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition, in Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, pp 243-246 (1993)
11. "Buenas prácticas en la especificación del dominio de una aplicación", de autores "Leandro Antonelli, Gustavo Rossi, Julio Cesar Sampaio do Prado Leite, Alejandro Oliveros"
12. Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software, Prentice Hall (1990).
13. Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software, Prentice Hall (1990).
14. Hadad, G. D. S. (2008). Uso de escenarios en la derivación de software [Tesis doctoral, Universidad Nacional de La Plata, Facultad de Ciencias Exactas].
15. González Díaz. Introducción a la construcción de compiladores. Universidad de Valladolid. (sf)
16. Hernández Yáñez, L. Fundamentos de la programación. Universidad Complutense. 2014.
17. Bangor, A., Kortum, P. T., Miller, J. T.: "An empirical evaluation of the system usability scale." Intl. Journal of Human–Computer Interaction 24.6, pp. 574-594, 2008.
18. Brooke, J.: “SUS-A quick and dirty usability scale” Usability evaluation in industry, 189(194), pp. 4-7, 1996.
19. Brooke, J: "SUS: a retrospective", Journal of usability studies 8.2, pp.29-40, 2013.
20. McLellan, S., Muddimer, A., Peres, S. C.: "The effect of experience on System Usability Scale ratings." Journal of usability studies 7.2, pp. 56-67, 2012.
21. Leite, J. C. S. D. P., & Franco, A. P. M. (1993). A strategy for conceptual model acquisition. En Requirements Engineering, 1993., Proceedings of IEEE International Symposium on (pp. 243-246). Río de Janeiro, Brasil: Departamento de Informática, Pontificia Universidade Católica do Rio de Janeiro.
22. Kaplan, G. N. (2022). Proceso de requisitos validado empíricamente [Tesis doctoral, Universidad Nacional de La Plata, Facultad de Ciencias Exactas].
23. Ruidías, H. J., Caliusco, M. L., & Galli, M. R. (2014). Construcción basada en ontologías del Léxico Extendido del Lenguaje. Universidad Gastón Dachary, Posadas, Argentina.
24. ISO/IEC. (1996). Information technology — Syntactic metalanguage — Extended BNF (ISO/IEC 14977:1996)1.
25. Bäumer, D., Riehle, D., Siberski, W., Wulf, M.: The Role Object Pattern In Proceedings of PLOP 97, Monticello, Illinois, USA (1997).
26. Chelimsky, D., Astels, D., Helmkamp, B., North, D., Dennis, Z., Hellesoy, A.: The RSpec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, Pragmatic Bookshelf, (2010).
27. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns CD: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional (1994).
28. Gunter, K. A., Mitchell, J. C.: Theoretical Aspects of Object-Oriented Programming, The MIT Press (1994).

29. IEEE Recommended Practice for Software Requirements Specifications, IEEE ComputerSociety, IEEE Std 830-1998 (1998)
30. Kovitz, B. L.: Practical software requirements: A manual of content and style, ISBN 1884777597, Manning, Greenwich (1999).
31. Leonardi, C., Leite J.C., Rossi G.: Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural, Tesis de maestría, http://www-di.inf.puc-rio.br/~julio/teses.htm, Facultad de informática, Universidad Nacional de R32 La Plata, Argentina (2001).
32. Liskov, B., Wing, J.: A behavioral notion of subtyping, ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16, Issue 6, pp. 1811 – 1841 (1994).
33. Liskov, B., Wing, J.: A behavioral notion of subtyping, ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16, Issue 6, pp. 1811 – 1841 (1994).
34. Rosenberg, L.: Methodology for Writing High Quality Requirement Specifications and for Evaluating Existing Ones, Software Assurance Technology Center, NASA Goddard Space Flight Center Greenbelt, MD, September 24 (1998)
35. Wiegers, K.: Software requirements, Miscrosoft Press (1999).
36. Sebastián, A., & Hadad, G. D. S. (2015). Mejoras a un modelo léxico mediante mapas conceptuales. CACIC 2015 – XXI Congreso Argentino de Ciencia de la Computación.
37. Urbieta, M., Antonelli, L., Rossi, G., & Sampaio do Prado Leite, J. C. (2020). The impact of using a domain language for an agile requirement management. Information and Software Technology, 127, 106375. https://doi.org/10.1016/j.infsof.2020.106375
38. Centeno, M. E. (2004). Estimación del tamaño de los artefactos producidos en la elicitación de requerimientos [Tesis de maestría, Universidad Nacional de La Plata, Facultad de Informática].
39. Leite, J. C. S. D. P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G. D. S., & Oliveros, A. (1997). Enhancing a requirements baseline with scenarios. Requirements Engineering.
40. Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2016). The Use and Effectiveness of User Stories in Practice. Requirements Engineering: Foundation for Software Quality
41. Cohn, M.: User stories applied: for agile software development. Addison Wesley (2004)
42. Litvak, C. S., Hadad, G. D. S., & Doorn, J. H. (2014). Heurísticas para el modelado de requisitos escritos en lenguaje natural. DIIT, Universidad Nacional de La Matanza, Argentina.
43. Antonelli, L., Rossi, G., Sampaio do Prado Leite, J. C., & Oliveros, A. (2013). Buenas prácticas en la especificación del dominio de una aplicación. LIFIA, Facultad de Informática, UNLP, La Plata, Buenos Aires, Argentina.