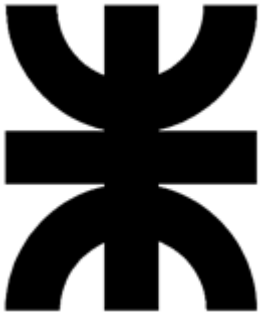

Universidad Tecnológica Nacional
Facultad Regional Santa Fe

Informe de Proyecto Final de Carrera

Ingeniería en Sistemas de Información

"Análisis de variabilidad de líneas de productos de software especificadas en Kconfig"

Alumnos

González, Rocío Pilar
LU: 21353

Sequeira, Matías Alejandro
LU: 21501

Director

Dr. Silvio Gonnet

Año 2017

Índice

1. Introducción

Una tendencia creciente en el desarrollo de software es la necesidad de desarrollar múltiples productos de software similares en conjunto en vez de varios productos individuales. Hay varias razones para esto: los productos pueden estar enfocados a distintos sectores del mercado, estar sujetos a distintas restricciones legales o culturales, o deben satisfacer necesidades específicas de diferentes stakeholders. Debido a las restricciones de costo y tiempo, no es posible desarrollar un nuevo producto desde cero para cada cliente, y el re-uso de software debe ser incrementado. Frente a este desafío, la ingeniería de líneas de productos de software (SPLE: Software Product Line Engineering) surge como un paradigma viable e importante que permite a las empresas desarrollar familias de productos, disminuyendo costos y tiempos, basándose en el re-uso de componentes [1] [2].

Una línea de productos de software (SPL: Software Product Line) es una familia de sistemas de software desarrollados a partir de un conjunto de características comunes, que apunta a satisfacer necesidades específicas de un segmento de mercado. Una SPL está constituida por un núcleo que contiene los componentes presentes en todos los productos o aplicaciones derivadas, y un conjunto de elementos variables, variabilidad, que incluye aquellas características optativas de la aplicación [1] [2]. Muchos proyectos de desarrollo de software deben administrar una variabilidad muy grande. Proyectos que adoptan SPL emplean el concepto de variabilidad para derivar productos de software individuales en nicho de mercados [3]. Los modelos de variabilidad representan las características, o “features”, comunes y variables de productos en una SPL [4] [5]. Por otro lado, existen proyectos de software altamente configurables, como es el caso del kernel de Linux, donde las opciones de configuración (características o features) son empleadas para derivar el producto cumpliendo ciertas propiedades funcionales y no-funcionales, según las necesidades del usuario. Generalmente,

se denomina configuración a un producto posible de la SPL, el cual es derivado a partir de un conjunto de características.

Sistemas altamente configurables pueden llegar a tener un gran número de características. Reportes de sistemas industriales indican poseer cientos de características [6] y en sistemas "open-source", la cantidad de características asciende a miles [7]. Particularmente, el kernel de Linux versión 4.2 para la arquitectura x86 posee más de diez mil características.

Las características en un sistema configurable interactúan de una manera no trivial, y su interacción puede introducir errores en los productos derivados [8]. El número de configuraciones (productos derivados) es exponencial en el número de características, por lo que no es posible analizar cada configuración en forma separada. En los últimos años se identificaron varias funciones que son de utilidad para extraer información desde los modelos de características (Feature Model), tales como detección de inconsistencias, detección de características muertas (las cuales no pueden incluirse en ningún modelo), número de productos derivables, entre otras [4] [9]. Debido al incremento en tamaño y complejidad de los modelos, surge el desafío de proporcionar un soporte automático para llevar adelante estas funciones. Muchas de estas funcionalidades están disponibles en el ambiente SPLOT (www.splot-research.org) [10], una herramienta académica orientada a la definición de modelos de características y su análisis mediante el empleo de lógica proposicional y problemas de satisfacción de restricciones. Sin embargo, los ejemplos disponibles son casos de estudios pequeños y no reflejan modelos del mundo real como los expresados mediante Kconfig.

Kconfig [11] fue creado para describir la variabilidad del kernel de Linux y ha sido adoptado por diversos proyectos de desarrollo de código abierto para definir su variabilidad [7] [12]. El modelo de variabilidad especificado en Kconfig puede ser interpretado como un modelo de características [12] [13].

A partir del escenario planteado, en este proyecto se plantea el análisis de la variabilidad de modelos especificado en Kconfig. Para lo cual, se propone la

construcción de una herramienta para el análisis de la variabilidad en Líneas de Producto de Software. Para esto, a grandes rasgos, se traduce de un modelo de variabilidad especificado en Kconfig a un modelo de características. El resultado es integrado a herramientas de análisis de modelos de características para poder finalmente obtener información con respecto a su variabilidad.

Para facilitar la lectura y compresión del informe en la sección 2 se exponen las definiciones de los conceptos más importantes, con el fin de establecer un vocabulario de base para desarrollar el resto de los temas. Seguidamente, en la sección 3 se describe la metodología seleccionada para el desarrollo del proyecto explicando cómo se adaptó la metodología a nuestro equipo de trabajo, los beneficios adquiridos de su utilización y todos los elementos y conceptos necesarios para poder implementarla. Luego, en el apartado 4 se exhiben las tecnologías utilizadas para construir la herramienta.

Posteriormente se expone el funcionamiento del sistema en la sección 5. Para esto, se explica en detalle cómo se lograron cada una de las funcionalidades previstas y se despliegan diagramas de la arquitectura propuesta para la lógica de la herramienta, el modelo conceptual utilizado y la arquitectura final de la aplicación.

Luego, en el apartado 6 se explica la ejecución del proyecto, planificada en el plan correspondiente a este proyecto final. Se muestra lo obtenido en la etapa de Investigación y etapa de desarrollo, la forma con la cual se lograron los objetivos propuestos, el tiempo requerido y las salidas obtenidas de cada etapa. En la etapa de desarrollo se profundiza sobre cada iteración, donde se muestra el periodo de tiempo de ejecución, una comparación entre lo estimado en el plan y lo efectivamente realizado, gráficos de avance, gestión de riesgo (de acuerdo a lo propuesto en el plan) y se incluyen comentarios y conclusiones pertinentes a cada iteración.

Finalmente en la sección 7 se explican las funcionalidades obtenidas mediante capturas de pantalla de la herramienta y en la sección 8 se presentan las conclusiones, la conclusión y una propuesta de trabajos futuros.

2. Terminología

En esta sección se explicará al lector los conceptos básicos utilizados para la realización de este proyecto. Se comenzará presentando la idea de Línea de Productos de Software, pasando a los conceptos de Modelo de Variabilidad, Lenguaje Kconfig y por ultimo, Modelo de Características.

Línea de Productos de Software

Una SPL es un conjunto de sistemas intensivos en software los cuales comparten un conjunto común de características, es gestionada para satisfacer necesidades específicas de un dominio o segmento particular del mercado, y es desarrollada a partir de un núcleo de software común de una manera prescrita [14].

Se trata de un concepto innovativo y creciente en la ingeniería del software, que permite a las empresas reducir costos y cumplir las metas de calidad, haciendo énfasis en el re-uso, de forma estratégica y planificada, con resultados predecibles.

Dentro de los beneficios que puede obtener una empresa al utilizar un enfoque basado en SPL también podemos mencionar: mejoras en la productividad y calidad, reducción en costos y tiempo al mercado, y la habilidad de expandirse a nuevos mercados en un menor tiempo.

Modelo de Variabilidad

La variabilidad se define como la habilidad de cambio o de personalización de un sistema [15]. Existen dos grandes grupos en los que se dividen los sistemas con variabilidad: los sistemas personalizables, donde la variabilidad se debe principalmente a la selección realizada por el usuario de las partes que más le interesa; y las familias de productos [16], donde una serie de productos más o menos similares se unen para permitir la reutilización de la parte común. En

estos sistemas con variabilidad, los requisitos deben incluir la definición de las cualidades del sistema variable o de la familia de sistemas. La principal diferencia con los sistemas tradicionales es que se debe prestar un especial interés al análisis de la parte común y de las partes variables, estableciendo las dependencias entre ellas. La variabilidad de un sistema puede ser representada a través de un modelo de variabilidad.

Lenguaje Kconfig

Como se mencionó anteriormente, el lenguaje de Kconfig fue creado para describir la variabilidad del kernel de Linux y ha sido adoptado por diversos proyectos de desarrollo de código abierto para definir su variabilidad.
En Linux se denomina configs a las opciones de configuración (características). Estas opciones y las dependencias entre las mismas se definen mediante el lenguaje Kconfig. En este lenguaje las configs pueden estar anidadas dentro de otras configs (menuconfig); pueden estar agrupadas en menús (menu), en grupos de opciones (choice) o bloques if. Estos agrupadores pueden anidarse entre sí de manera indistinta excepto los grupos del tipo choice, los cuáles sólo pueden contener choices. El configurador del kernel, xconfig, permite al usuario seleccionar el conjunto de opciones de configuraciones con las cuales construirá el producto final (el kernel en el caso de Linux). Esta herramienta presenta el modelo Kconfig como un árbol de opciones, donde el usuario selecciona las configuraciones que desea para construir el kernel.
Cada config posee un nombre y un tipo. Los tipos pueden ser: bool, tristate, integer (int o hex), o string [11] [12]. Una config de tipo boolean representa una opción que puede ser seleccionada como parte del producto final (y) o no (n). Una config del tipo tristate es similar a la del tipo boolean, pero posee dos alternativas de inclusión de la parte en el producto final: y indica que el código que implementa la opción es enlazado en el kernel de manera estática, mientras

que m representa que debe ser compilado como un módulo que carga de manera dinámica.

nteger configs se emplean para especificar opciones numéricas, tal como el tamaño de un buffer. String configs permiten especificar el nombre de un elemento configurable, como puede ser el nombre de una partición del disco.

She et al. [12] denominan a las configs de tipo boolean y tristate como “switch configs”, en cambio a las config de tipo integer y string la denominan “entry-field config”.

```
config X86_HT
        bool
config MMU
        def_bool y
config SBUS
        bool
menuconfig W1
        tristate "Dallas's 1-wire support"
if W1
config W1_CON
        bool "Userspace communication over connector"
        default y
menu "1-wire Bus Masters"
config W1_MASTER_DS2490
        tristate "DS2490 USB <-> W1 transport layer"
config W1_MASTER_DS2482
        tristate "Maxim DS2482 I2C to 1-Wire bridge"
        select X86_HT
config W1_MASTER_MXC
        tristate "Freescale MXC 1-wire busmaster"
        depends on MMU || SBUS
endmenu
endif
```

Figura 2.1. Fragmento de un modelo de variabilidad del kernel de Linux expresado en Kconfig.

lLos menús son empleados para agrupar elementos Kconfig y los menuconfig son similares a las config, con la diferencia que generalmente dichas sentencias están sucedidas por un bloque if.

La finalidad del bloque if es la agrupación de elementos bajo una determinada expresión, la cual en la mayoría de los casos referencia una config o un menuconfig. Cuando dicha expresión evalúa verdadera, los elementos dentro del bloque if pueden ser configurados/empleados. Por esto, se podría pensar que la especificación if agrega una dependencia entre los componentes del bloque y la/s config/s de la expresión que evalúa.
En la Figura 2.1 se define, un menuconfig (W1) seguido de la sentencia if, lo cual representa el comienzo de un bloque if. A su vez, el bloque if contiene una config (W1_CON) y un menu (1-wire Bus Masters) compuesto por tres configs de tipo Tristate. Estos elementos dependen del valor de la config W1 y podrán ser configurados si W1 es seleccionado (valores y o m).

Kconfig provee un mecanismo de visibilidad condicional para los elementos del lenguaje, colocando una especificación “depends on” debajo de la declaración de los mismos. Una especificación depends-on introduce una dependencia que debe ser satisfecha cuando se selecciona la config. Si la condición es falsa, el elemento en cuestión y sus hijos son descartados por el configurador. En la Figura 2.1, W1_MASTER_MXC depende de la selección de al menos una de las opciones: MMU o SBUS.

Inverso a lo anteriormente explicado, una especificación select obliga la selección de otra config cuando la config es seleccionada por el usuario. En el caso representado por la Figura 2.1, cuando se selecciona W1_MASTER_DS2482, se debe seleccionar X86_HT.

Los grupos de opciones (choice configs) permiten definir alternativas. Las choice configs heredan el tipo de las configs contenidas dentro del grupo. Las opciones pueden ser bool o tristate. Cuando la choice adquiere el tipo bool se debe seleccionar una única opción (XOR) [1...1], como si la misma estuviera seleccionada (y); cuando adquiere el tipo tristate, se puede seleccionar uno o más opciones (OR) [1…*], como si hubiera adquirido el valor m. La choice de la Figura 2.2 es un ejemplo de XOR. Adicionalmente, un grupo de opción marcado

como opcional (optional) puede ser configurada con el valor n, es decir, no es necesario seleccionar algunas de sus opciones ([0...1] para bool y [0…*] para tristate). Sin embargo, un grupo de opciones sin una marca es considerado obligatorio y no se le puede asignar el valor n. Es necesario destacar que la sentencia optional no fue incluida dentro del ejemplo de la Figura 2.2.

```
menuconfig THERMAL
        tristate "Generic Thermal sysfs driver"
if THERMAL
config THERMAL_HWMON
        bool
        prompt "Expose thermal sensors as hwmon
device"
        default y
choice
        prompt "Default Thermal governor"
config THERMAL_DEFAULT_GOV_STEP_WISE
        bool "step_wise"
        select THERMAL_HWMON
config THERMAL_DEFAULT_GOV_FAIR_SHARE
        bool "fair_share"
endchoice
endif
```

Figura 2.2. Fragmento de un modelo de variabilidad del kernel de Linux expresado en Kconfig.

A modo de cierre, con el fin de dar al lector una noción de las dimensiones de este lenguaje, el modelo de la versión 4.2 del kernel Linux posee más de 10000 cláusulas config, 200 menuconfig, 170 menu, 60 choice, 6900 definiciones de select, y 9000 depends on. Al momento de la redacción de este informe, la última versión del kernel es la 4.9. Es de esperarse un modelo aún más grande debido al continuo crecimiento de este sistema operativo.

Modelo de Características (Feature Model)

Un modelo de características es una representación compacta de todos los productos de una SPL en términos de características (features). Una “feature” es

definida como un aspecto prominente o distintivo visible por el usuario, calidad, o característica de un sistema de software o sistema [4].

Las características se representan mediante una estructura de árbol, en donde se vinculan características “padres” con sus “hijos”. Además pueden existir restricciones que afecten a dos o más características de cualquier lugar del modelo. Las relaciones permitidas entre características son las siguientes:

- Obligatoria: Relación que vincula una característica hija con su padre, indicando que la hija aparecerá en todos los productos en los que el padre esté incluido.
- Opcional: Relación que vincula una característica hija con su padre, indicando que la hija podrá opcionalmente estar incluidos en algunos productos en los que esté incluido su padre.
- Cardinalidad grupal: La cardinalidad grupal es un intervalo denotado por [min…máx] que limita el número de características hijas que pueden ser incluidas en un producto, cuando su padre está incluido, siendo los límites inferior y superior respectivamente.
- Requiere: Esta relación indica una implicación entre una característica requerida y una característica restringida. La característica restringida sólo puede incluirse en aquellos productos en los que se encuentre la característica requerida.
- Excluye: Esta relación indica una exclusión mutua entre dos características.

En la Figura 2.3 se ilustra un ejemplo de un modelo de característica pequeño, el cual es ilustrado empleando la herramienta SPLOT. Este modelo representa una porción de la variabilidad del kernel de Linux, en particular se ilustra la arquitectura Rapid IO. Los círculos negros de la Figura 2.3 indican una relación de obligatoriedad entre las características padre e hijas. Los círculos blancos representan una relación de tipo opcional entre padre e hijo. Las relaciones de

cardinalidad grupal se indican con sus valores mínimos y máximos debajo de la característica padre.

Según el modelo de la Figura 2.3, toda arquitectura Rapid IO (representado por Menu Padre en Figura 2.3) debe contar con las propiedades RAPID_DISC_TIMEOUT, un Enumeration method, y un RapidIO Switch drivers. Opcionalmente puede incluir las características: RAPIDIO_TSI721, RAPIDIO_ENABLE_RX_TX_PORTS, RAPIDIO_DMA_ENGINE, y RAPIDIO_DEBUG.

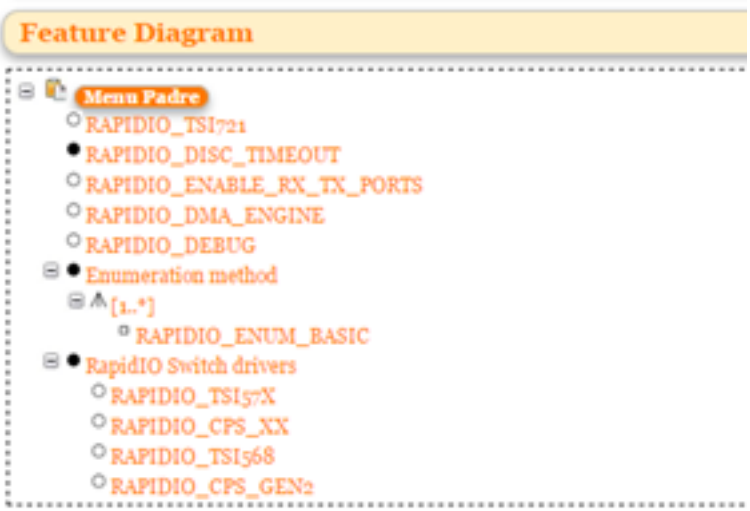

Figura 2.3. Ejemplo de diagrama de características: arquitectura Rapid IO.

3. Metodología utilizada

Durante la formulación del plan de este proyecto se seleccionó Scrum para la fase de desarrollo. Es necesario hacer la salvedad sobre la adaptación de este modelo de proceso a las circunstancias del equipo de desarrollo, el cual fue de sólo dos personas. Otro desvío de la metodología pura de Scrum fue la necesidad de una etapa de investigación, la cual suplanta la iteración 0, donde se define el modelo, arquitectura y además se recaba información útil para entender el problema en cuestión.

Se optó por este modelo teniendo en consideración diferentes aspectos:

- Al estructurar el desarrollo en ciclos de trabajo, se mitigan los riesgos acarreados por el proyecto,
- Falta de experiencia previa en gestión de proyectos de desarrollo y en la definición y ejecución de planes proyecto,
- Al contar el equipo con solo dos integrantes, la asignación de tareas y actividades es más dinámica en contraste con una metodología dirigida por un plan en particular,
- El área de estudio tratada por la herramienta era un campo desconocido, con necesidad de profundizar durante el camino y de constante validación con expertos en el dominio y con el director del proyecto.
- Se obtiene retroalimentación en cada iteración para evaluar la evolución del trabajo, y se disminuye la incertidumbre surgida de las entregas a largo plazo.

Entre las ventajas mencionadas, se destaca:

- Adaptabilidad. Propone un marco de trabajo, haciendo la metodología independiente.
- Orientado a las personas a través de la definición de roles.

- Propone un enfoque iterativo e incremental a través de un ciclo de pasos, permitiendo dar momentos de valoración al proyecto y planteo de ajustes y cambios.

Previo al inicio del proyecto, durante la etapa de planificación, se definieron los siguientes conceptos/elementos de trabajo:

- Historias de usuario a implementar con sus descripciones y estimaciones en Story Points (SP).
- Una serie de bocetos (spikes).
- El Product Backlog ordenado por prioridad con las historias discriminadas por Sprint.
- La velocidad de desarrollo estimada. Se definió por cada integrante una velocidad de 1 SP por semana, quedando 3 SP por iteración. Considerando un equipo de 2 personas, se dispuso de 6 SP por iteración. Siendo en total 5 iteraciones de desarrollo, se obtuvo un total de 30 SP para implementar todas las funcionalidades. De acuerdo al product backlog, fueron necesarias 11 historias de usuario. En la sección “Etapa de desarrollo” (6.B.) se muestra la distribución de SP según las historias de usuario. Dichas historias de usuario se encuentran en la sección B “Historias de usuario” del Anexo, ubicado al final de este informe.

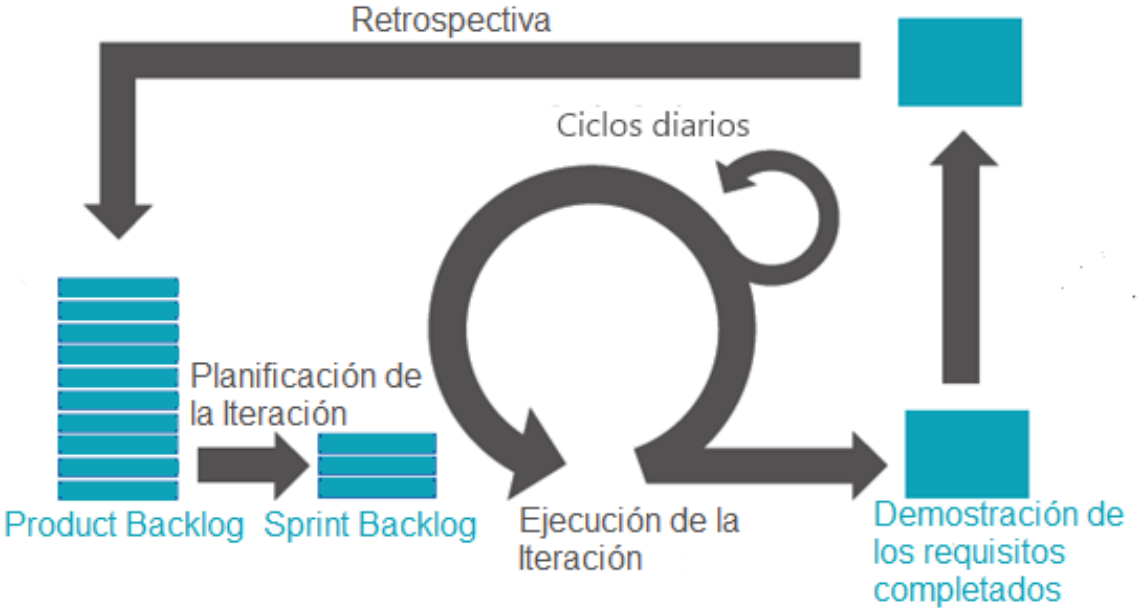

Figura 3.1. Ilustración de la metodología aplicada.

Para la planificación se emplearon diversos elementos y técnicas, propuestos por Scrum, estos son ilustrados en la Figura 3.1:

- Actividades Scrum:
  - Planificación de la iteración: Instancia en cada comienzo de una iteración para revisar el product backlog y decidir cuáles son las funcionalidades con prioridad para ser desarrolladas en el sprint en base a la velocidad de desarrollo que se estaba teniendo. La salida es un sprint backlog ordenado de las funcionalidades a desarrollar.
  - Ejecución de la iteración: Durante esta actividad se desarrollaron las tareas establecidas durante la planificación de la iteración.
  - Demostración de los requerimientos completados y retrospectiva: Realizadas al finalizar cada sprint y al tener un incremento funcional del producto. En estas reuniones participó el director del proyecto, además del equipo de desarrollo. En las misma se presentó el valor agregado que el sprint introdujo al proyecto, y además se discutió la planificación del próximo sprint.

- Artefactos o herramientas:
  - Product Backlog (Lista de requisitos priorizada): Contiene los objetivos de alto nivel expresados en forma de historias de usuario. Cada uno de los objetivos se encuentra asociado al valor que aporta al proyecto y por ende, al cliente. Cada entrada de esta lista se conoce por el nombre de historia (Story).
  - Sprint Backlog (Lista de tareas para la iteración): Contiene la lista de tareas a ejecutar durante la iteración. Cada entrada de esta lista se conoce por el nombre de tarea (Task).
  - Product Burndown Chart (Diagrama Burndown del Producto), el cual proporciona una visión global del estado del proyecto en términos de trabajo por realizar y tiempo restante para la finalización del

proyecto. La herramienta propuesta para la gestión del proyecto (Kunagi) no posee la funcionalidad para generar esta gráfica. Como se consideró importante su inclusión dentro de este informe, tuvo que ser creada manualmente mediante una hoja de cálculo.

- Sprint Burndown Chart (Diagrama Burndown de la iteración), cuya utilización permite conocer el grado de avance en una iteración. Ambos Burndown Charts permiten comparar el nivel de avance real del proyecto contra el avance planificado para un determinado momento.

- Roles involucrados en la metodología Scrum:
  - Product Owner (Dueño del Producto): El director del proyecto fue quien definió los requisitos a incluir en el Product Backlog, y proporcionó una valoración de cada uno de estos. Además participó en la planificación de la iteración y en la demostración de los requerimientos completados.
  - Scrum Master (Facilitador): Esta figura particularmente no fue adoptada debido al tamaño limitado del equipo de desarrollo. Además, como el cliente/director de este proyecto también es del área de informática y posee conocimientos de Scrum, no fue necesario guiarlo dentro del proceso.
  - Desarrollador: Participa en todas las etapas de la iteración y es responsable del desarrollo del producto.

Herramienta para la gestión del proyecto orientada a Scrum

La herramienta seleccionada para la gestión fue Kunagi. Esta herramienta permite definir un Product Backlog ordenado por prioridad. Al estar las historias ordenadas, para la planificación de las iteraciones (Sprint Backlog) sólo fue necesario hacer un “Pull” de la próxima historia. Asimismo, la herramienta permite definir las tareas que componen una historia con el objetivo de poder

trabajar de una manera más granular. Sumado a esto, también permite definir las horas estimadas para cada tarea como así también el tiempo destinado a las mismas a medida que se iba avanzando en su desarrollo. Inicialmente, el estado de una tarea era “Free”, cuando se seleccionaba alguna para su progreso se pasaba al estado “Claimed” y, cuando se finalizaba con la misma o se cumplía la estimación establecida era pasada al estado “Completed”. Para ilustrar lo mencionado anteriormente, en la Figura 3.2 se muestra una captura de pantalla del Scrum Whiteboard de Kunagi.

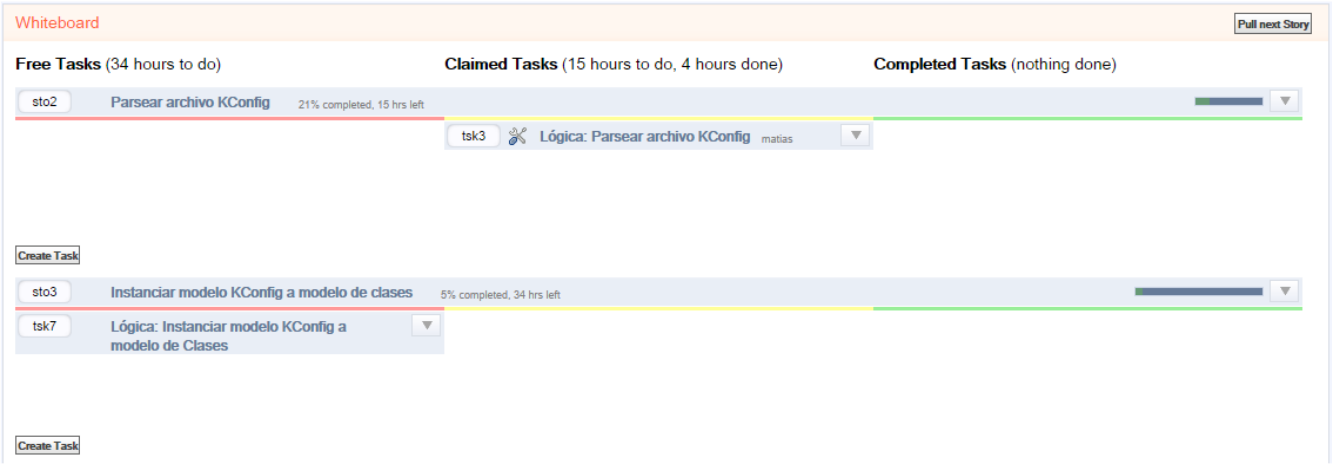

Figura 3.2. Captura de pantalla de Kunagi

Con respecto al desarrollo del proyecto, cada integrante trabajó en su propio ambiente de desarrollo y, al ser la comunicación fluida y constante, no hubo necesidades de resolver conflictos de desarrollo. Aunque no se realizaban reuniones diarias de 15 minutos como estipula SCRUM, se habían fijado reuniones semanales en las que se comentaban avances, se planteaban dudas, se discutían problemas que se presentaban y se resolvían cuestiones de implementación si alguno de los miembros se encontraba detenido con algún problema particular.

Al finalizar con las tareas del sprint backlog, se planificaba la revisión de la iteración y luego, se procedía a la planificación de una nueva, comenzando una vez más.

4. Tecnologías utilizadas

A continuación se presentarán las tecnologías empleadas las cuales permitieron el desarrollo de la herramienta en cuestión. Primero se listan y explican los lenguajes utilizados para la programación y diseño+desarrollo de interfaces, luego las herramientas que dieron soporte al proceso de desarrollo y por último los entornos utilizados.

Lenguajes

Los lenguajes se pueden clasificar según su utilización: Java para la lógica de la aplicación y JavaFX para las interfaces de usuarios.

- Java: La herramienta está desarrollada en mayor parte en este lenguaje, ya que era previamente conocido por el equipo de desarrollo y además tiene las ventajas de disponer de mucha documentación en la web. Para el desarrollo en este lenguaje se utiliza el jdk 1.8.

- Para el diseño y desarrollo de las interfaces de usuario se utilizó JavaFX, una familia de productos y tecnologías de Oracle para aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio. Para esta solución en particular, la cual no es web, se decidió adoptar esta tecnología con el objetivo de evitar la utilización de Java Swing y de poder darle un mejor aspecto visual. JavaFX hace uso de código FXML y CSS. FXML es un lenguaje basado en XML el cual provee la estructura para crear una interfaz de usuario separada de la lógica de la aplicación. CSS es un lenguaje usado para establecer el diseño visual de las interfaces de usuario.

Soporte para el desarrollo

A continuación se listan las herramientas utilizadas dentro del proceso de desarrollo de la herramienta:

- ANTLR4 es una poderosa herramienta para la generación de parser para leer, procesar, ejecutar o traducir texto estructurado. Se decidió utilizar esta herramienta ya que al estar escrita en Java genera código en el mismo lenguaje. Además posee un plugin para el entorno de desarrollo Eclipse, evitando la necesidad de ejecutar comandos por consola y haciendo posible un entorno de desarrollo integrado. Para la utilización de esta herramienta se generó un archivo con extensión .g4, el cual describe mediante expresiones regulares al lenguaje Kconfig.

- Git+GitHub: Para el control de versiones se utilizó Git, un sistema distribuido de control de versiones. Al ser distribuido, cada desarrollador cuenta con un "clon" completo del repositorio. Cada uno puede realizar cambios sobre el proyecto, lo que permite ser autónomo y trabajar en cualquier situación. Como repositorio central, se decidió utilizar los servicios de GitHub, una plataforma de desarrollo colaborativo de software para el alojamiento de repositorios de proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública bajo licencias de código Open Source.

- Google Drive: Ante las necesidades de un repositorio compartido para poder subir y compartir archivos del proyecto, se eligió la opción de utilizar Google Drive. Junto a Drive, se utilizó Docs y para la redacción de documentos y Sheets para

la elaboración de planillas de cálculo, ambos de forma colaborativa y en tiempo real.

- Kunagi: Entre las necesidades para encarar una metodología ágil inspirada en SCRUM, era necesario una herramienta de soporte para la gestión del desarrollo ágil sin implicar mayores complejidades técnicas. Como se explicó anteriormente, Kunagi ofrece la administración integrada de proyectos complementando la metodología Scrum a través de otras mejores prácticas para cubrir todas las necesidades para la administración de proyectos. Esta herramienta corre localmente en la máquina de un integrante del equipo, sobre un servidor Apache Tomcat.
- Balsamiq Mockups: Con el objetivo de establecer un acuerdo común en cuanto al diseño de las interfaces de usuario dentro del equipo de desarrollo se necesitó una herramienta para crear maquetas de interfaces de usuario. Balsamiq Mockups permite la creación de maquetas mediante un sistema Drag&Drop de elementos visuales, pudiendo generar pantallas en un corto plazo de tiempo.

Entornos de desarrollo

Los entornos de programación utilizados fueron:

- Eclipse: Se hizo uso de la conocida plataforma de software de código abierto multiplataforma. Una de las principales razones de la selección de esta herramienta es la capacidad de integrar un conjunto de herramientas en un sólo entorno de desarrollo: posee plugins para ANTLR, Git y JavaFX.
- SceneBuilder: Esta herramienta visual permite al usuario rápidamente diseñar interfaces de usuario JavaFx sin necesidad de codificar demasiado, empleando la metodología

Drag&Drop de componentes. El usuario puede modificar las propiedades de los componentes, aplicar hojas de estilo y el código FXML es generado automáticamente. El resultado es un archivo FXML que puede ser enlazado con un proyecto Java.

5. Funcionamiento del sistema

La herramienta propuesta para este trabajo tiene como fin la obtención de información de interés con respecto a la variabilidad de SPL especificadas en lenguaje Kconfig. En esta sección se explican algunos de los aspectos más importantes del funcionamiento de la misma.

Para lograr las funcionalidades previstas, en primer instancia se debió automatizar el proceso de construcción de un modelo de características (feature model) a partir de un modelo de variabilidad expresado en Kconfig, como los presentados en las Figuras 1.1 y 1.2 de la sección “Lenguaje Kconfig”. Luego, tomando la salida del paso anterior se pudo elaborar la lógica encargada de entregar el correspondiente análisis de variabilidad.

A través de la Figura 5.1 se puede observar la arquitectura de tipo Pipe & Filter propuesta para la lógica de procesamiento interna de la herramienta. El primer componente es un Parser cuya entrada es el modelo Kconfig y es el encargado de generar las instancias del modelo conceptual de Kconfig. Luego, el Transformador aplica un conjunto de reglas para traducir las instancias del modelo de Kconfig en instancias del modelo de características. Este modelo de características es especificado empleando la notación de la herramienta SPLOT, el cual es un archivo de tipo SXFM (Simple XML Feature Model). Por último, el analizador aplica las funcionalidades obtenidas a partir de un conjunto de herramientas para realizar el análisis de variabilidad y posteriormente entregar la información requerida.

Figura 5.1. Arquitectura propuesta para la lógica de la herramienta.

De acuerdo a los objetivos especificados en el plan, las funcionalidades para la herramienta de análisis de variabilidad de líneas de productos de software implementadas en la herramienta son las siguientes:

- Gestionar (Alta/Baja) modelo Kconfig: Dentro de la herramienta se mantiene y lista un conjunto de modelos/archivos Kconfig, los cuales se ponen a disponibilidad del usuario para ser convertidos a modelos de características. Los modelos Kconfig son persistidos en el FileSystem, donde está instalada la herramienta.
- Parsear archivo Kconfig: Los componentes de un archivo Kconfig son reconocidos sintácticamente. En la próxima sección, “Reconocimiento del lenguaje Kconfig”, se explica la implementación de esta funcionalidad.
- Instanciar modelo Kconfig según un modelo de clases: Una vez realizado el análisis sintáctico al archivo Kconfig, se generan las instancias de los componentes reconocidos, empleando un modelo de clases de los elementos de Kconfig. En la próxima sección, “Reconocimiento del lenguaje Kconfig”, se explica la implementación de esta funcionalidad.
- Traducir modelo de clases a modelo SPLOT: Un modelo Kconfig previamente instanciado es traducido a un modelo de características en un lenguaje admitido por la herramienta SPLOT. Este modelo de características obtenido también es guardado en el FileSystem, tal como lo hecho con los archivos Kconfig. En la sección “Reglas para traducción de Kconfig a Modelo de Características” se profundiza la implementación de esta funcionalidad.

Las funcionalidades listadas a continuación se detallan en el apartado "Análisis de Variabilidad":

- Reconocer inconsistencias en SPL: Las inconsistencias ocurren cuando existen contradicciones en las reglas y restricciones de dependencias.

Particularmente en un modelo de características, se dan cuando existe una relación entre características que no pueden ser verdaderas al mismo tiempo. Las inconsistencias se pueden dar también de manera transitiva, por ejemplo: una configuración A podría requerir B y B requerir C, se puede dar una inconsistencia transitiva si C excluye A.

- Determinar características muertas en SPL: Las características muertas son las configs que no aparecen en ningún producto correcto de una SPL debido a un conjunto de restricciones.
- Determinar características opcionales en SPL: Estas características son aquellas que derivan un producto distinto por cada variante. Una opción define una alternativa para el producto base que puede ser seleccionada como parte de un producto final o no.
- Validar producto ingresado: La herramienta permite determinar si un conjunto de configuraciones seleccionadas de una SPL definen un producto válido. Se considera válido un producto cuando no contiene inconsistencias ni configuraciones mandatorias faltantes.
- Obtener cantidad de los productos derivables en SPL: La herramienta permite contar la cantidad de productos que se puede obtener de una SPL.
- Validar configuración parcial de un producto: Se permite al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo Kconfig las cuales describen una SPL. Dada dicha selección y el modelo Kconfig, se comprueba si las configuraciones seleccionadas son compatibles, es decir, si pueden convivir en un producto válido.
- Obtener cantidad de productos derivables de una configuración parcial: Se permite al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo Kconfig el cual describe una SPL. Dada dicha selección y el modelo Kconfig, se entrega la cantidad de productos pertenecientes a la SPL que admiten las configuraciones seleccionadas.

Reconocimiento del lenguaje Kconfig

Con el objetivo de obtener esta funcionalidad, la cual representa la primer etapa del procesamiento interno de la herramienta, se elaboró un Parser (Figura 5.1), el cual reconoce la sintaxis y semántica del lenguaje Kconfig y a su vez convierte los elementos Kconfig del modelo de variabilidad a instancias del modelo conceptual propuesto en la Figura 5.2.

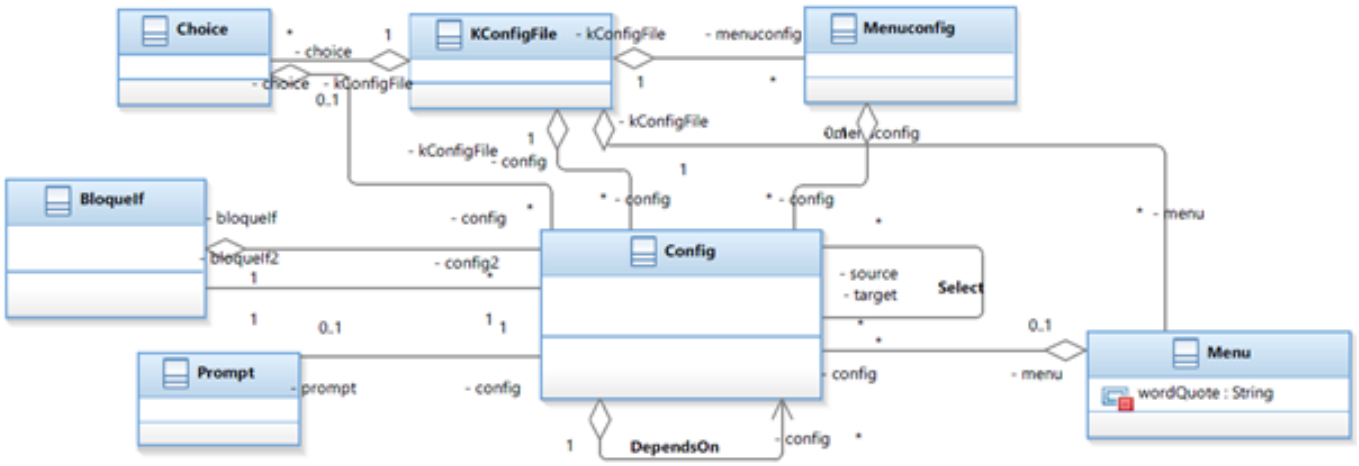

Figura 5.2. Modelo Conceptual de Kconfig.

Para la obtención de este último modelo se investigó la documentación del lenguaje Kconfig disponible en la web, siendo la principal fuente [11]. Además, se debió realizar un análisis minucioso de la sintaxis de estos archivos para salvar diferencias con la información obtenida, la cual no es del todo clara ni abundante.

Para facilitar el reconocimiento de la sintaxis del lenguaje, se hizo uso de la herramienta llamada ANTLR4. Como entrada para esta herramienta fue necesaria la elaboración de un archivo el cual describe la sintaxis del lenguaje Kconfig mediante expresiones regulares, este archivo puede ser visualizado en la sección D del Anexo “Parser del archivo KConfig”. Luego de generado el archivo mencionado, se procedió a ejecutar la herramienta y como resultado se

obtuvo un conjunto de clases y métodos escritos en java, los cuales posteriormente fueron modificados para adaptarse a las necesidades del equipo.

Reglas para traducción de Kconfig a Modelo de Características

Para obtener un modelo de características desde un modelo de variabilidad expresado en Kconfig, se analiza el lenguaje Kconfig, definiendo su semántica. La Figura 5.2 representa los conceptos de Kconfig considerados en este trabajo (Config, Menu, Menuconfig, Choice, If) .

Se definieron reglas las cuales permiten mapear conceptos de Kconfig a conceptos de modelos de características. Con el objetivo de facilitar la comprensión de estas reglas se utilizarán ejemplo cortos basado en la configuración de una bicicleta.

- Como raíz (_r) del árbol de características, se crea un menú ficticio (Menu en Figura 5.2) que contiene a los elementos raíz del archivo Kconfig.
- Aquellas configs (Config en Figura 5.2) que son del tipo “bool” o “tristate” son traducidas como configs opcionales (:o). El resto, las del tipo “string” e “int”, son tomadas como obligatorias (:m). En la Figura 5.3 se provee un breve ejemplo donde se muestra que los cambios (SHIFT) de la bicicleta tienen carácter de opcional y es obligatorio que la misma posea una marca (BRAND). El símbolo r_N_N sirve para identificar de manera unívoca las características dentro del árbol de características. Particularmente, para este ejemplo se utilizaron símbolos genéricos. Esto es explicado con más detalle en el próximo párrafo.

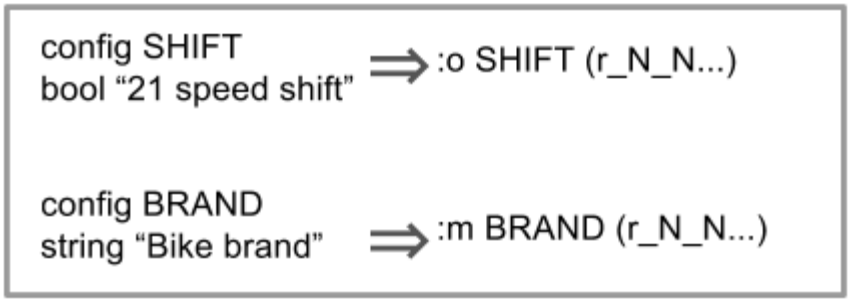

Figura 5.3. Traducción de configs.

- Se considera que todo lo que está dentro de un menú, pasa a ser elemento hijo del mismo (configs, menuconfigs, if, choices, y otros menús). En la Figura 5.4 se muestra un menú, el cual agrupa las configs opcionales para la inclusión de frenos a discos en cada rueda.

  Se puede observar en la Figura 5.4 la identificación en formato r_N_N… asignado a cada elemento. Por cada nivel dentro del árbol de características se incorpora un elemento _N al identificador y su prefijo es igual al identificador del padre. Para el ejemplo de la Figura 5.4, suponiendo que la raíz (el menú) fue identificado como _r_5, las configs hijas del menú serán entonces "_r_5_6" y "r_5_7". La selección del próximo número a utilizar sigue un criterio ascendente.

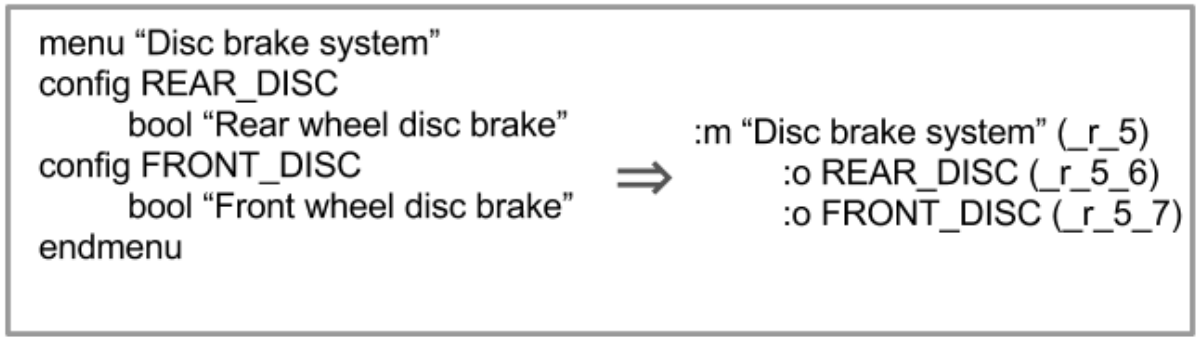

Figura 5.4. Traducción de menu.

- Una choice (Choice en Figura 5.2) es traducida según el "type" de las configs que la componen: si son del tipo "bool", sólo puede seleccionarse una de las opciones [1..1]; en caso de ser del tipo "tristate", se pueden seleccionar uno o más componentes con el valor "m" [1..*]. Si una choice posee "OPTIONAL" dentro de sus atributos, significa que las restricciones anteriores permiten la no selección de componentes (que tomen valor n), quedando [0..1] y [0..*], respectivamente. En la Figura 5.5 se muestra un ejemplo de configuración del cuadro (frame) de una bicicleta, con una choice que exige seleccionar un único ([1..1]) tipo de cuadro: de aluminio o de carbón.

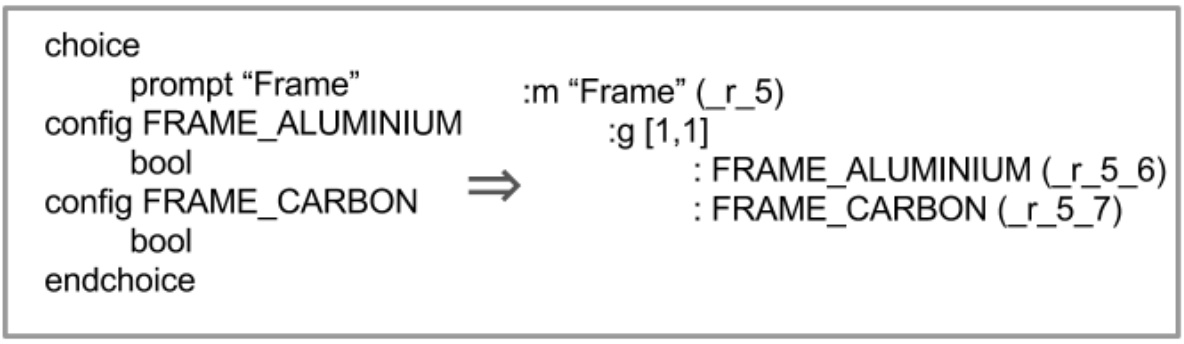

Figura 5.5. Traducción de choices.

- Los menuconfigs (Menuconfig en Figura 5.2) son traducidos de forma similar a las configs. Generalmente, cuando se encuentra un menuconfig, debajo del mismo aparece un bloque “if” cuya condición a evaluar es la config del menuconfig. Esto se puede pensar como que si todo lo que está dentro del bloque “if” es hijo del menuconfig. En caso de no estar precedido por un bloque “if”, el menuconfig es tratado como una config normal. En la Figura 5.6 se brinda un ejemplo de accesorios cuya inclusión es opcional al momento de configurar un producto correspondiente a una bicicleta.

Figura 5.6. Traducción de menuconfigs.

Adicionalmente, cuando la expresión a evaluar para un bloque if se encuentra compuesta por un símbolo negado o por más de un símbolo unidos por conectores lógicos (por ej. !A o A || B respectivamente), el contenido del bloque if se representa mediante restricciones del tipo “depends on” donde el contenido del bloque if depende de la expresión del

mismo, ya que no puede ser representado mediante la estructura de árbol. Las restricciones del tipo “depends on” son desarrolladas más adelante.

- Cuando una config posee la opción “select” (Select en Figura 5.2) sucedida por una expresión, se traduce como una restricción donde la config implica la expresión, tal como se muestra en el ejemplo de los pedales de aluminio (pedal_aluminium) y el cuadro de aluminio (frame_aluminum) en la Figura 5.7. En este caso se expresa en el archivo Kconfig que si se selecciona pedal_aluminium se deberá seleccionar el cuadro frame_aluminium. Observar que la restricción resultante es una implicancia expresada con una disyunción que especifica que no se selecciona pedal_aluminium o se elije frame_aluminium (equivalente a pedal_aluminium ⇒ frame_aluminium).

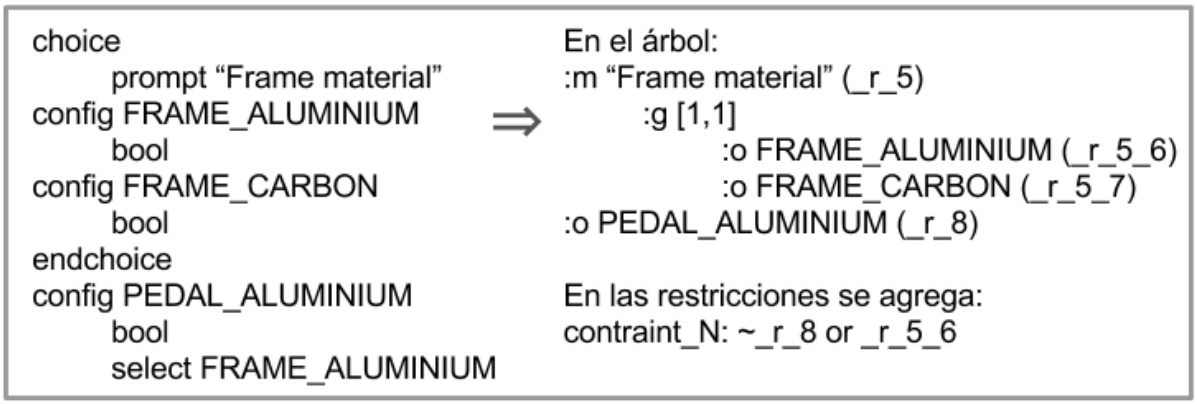

Figura 5.7. Traducción de la cláusula select.

Adicionalmente, si al final de esta opción se encuentra un “if”, la expresión que sucede al “if” es una condición que se debe cumplir para que se incluya el select. En la Figura 5.8 se muestra de manera gráfica lo explicado anteriormente.

```
config SIMBOLO
        bool
        select SIMBOLO_SELECCIONADO if CONDICION_IF

Proposicionalmente, la restricción generada por el "select" es:
CONDICION_IF ⇒ (SIMBOLO ⇒ SIMBOLO_SELECCIONADO)

Traducido a la forma disyuntiva:
¬CONDICION_IF ∨ ¬SIMBOLO ∨ SIMBOLO_SELECCIONADO
```

Figura 5.8. Generación de restricciones por cláusula "if" en el select.

En la Figura 5.9 un ejemplo similar al de la Figura 5.7, con la adición de la cláusula if. Observar la negación de la condición del "if" de la Figura 5.9 (mediante el símbolo !), la cual impacta en la restricción.

```
choice
        prompt "Frame material"
config FRAME ALUMINIUM
        bool
config FRAME_CARBON
        bool
endchoice
config PEDAL_ALUMINIUM
        bool
        select FRAME_ALUMINIUM if !FRAME_CARBON

⇒

En el árbol:
:m "Frame material" (_r_5)
        :g [1,1]
                :o FRAME_ALUMINIUM (_r_5_6)
                :o FRAME_CARBON (_r_5_7)
:o PEDAL_ALUMINIUM (_r_8)

En las restricciones se agrega:
constraint_N: _r_5_7 or ~_r_8 or _r_5_6
```

Figura 5.9. Traducción de la cláusula select seguida de "if".

- Cuando una entrada posee la opción "depends on" (DependsOn en Figura 5.2) y la expresión que le sigue a dicha opción hace referencia a sólo una entrada (el símbolo de un menú, una config, etc.), se traduce como una restricción donde la entrada que posee el "depends on" implica la entrada referenciada por el "depends on". En la Figura 5.10 se provee un ejemplo con el objetivo de dejar esto en claro.

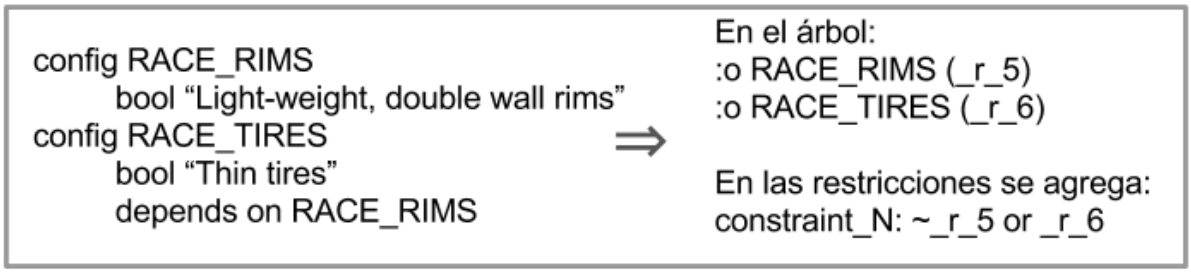

Figura 5.10. Traducción de la cláusula “depends on”.

Por lo contrario, si la expresión de la opción “depends on” hace referencia a más de una entrada, se agrega una restricción la cual representa una equivalencia *si-sólo-si* con la entrada referenciada (en lugar de la implicancia).

Con el objetivo de brindar al lector una mejor explicación sobre la generación de restricciones, en el siguiente subapartado se desarrolla esto con un mayor grado de detalle.

Restricciones

Las restricciones son traducidas por la herramienta en fórmulas proposicionales en forma normal conjuntiva (FNC), cada cláusula representa una restricción.

Aquellas restricciones las cuales poseen una implicancia bidireccional o de doble sentido, internamente son tomadas como dos restricciones distintas de un único sentido (⟵ y ⟶).

En la Figura 5.11 se muestra cómo se traduce una restricción del tipo “select”. Se consideran las posibles variantes del lenguaje Kconfig (representado por “B”). Luego se muestran la fórmulas proposicionales que representa cada expresión y las fórmulas equivalentes en FNC. Esta transformación se basa en reglas de equivalencias lógicas (leyes de De Morgan, leyes de distribución, leyes de asociación, etc.).

Adicionalmente, en la Figura 5.12 se muestra cómo es traducida una restricción del tipo “depends on” la cual hace referencia a más de una entrada.

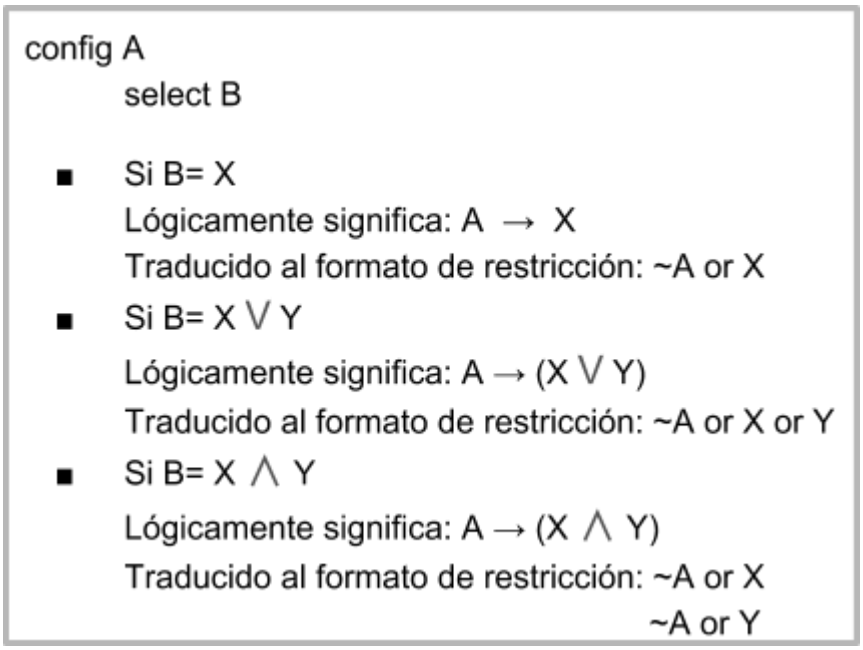

Figura 5.11. Traducción del atributo “select” al formato de restricción.

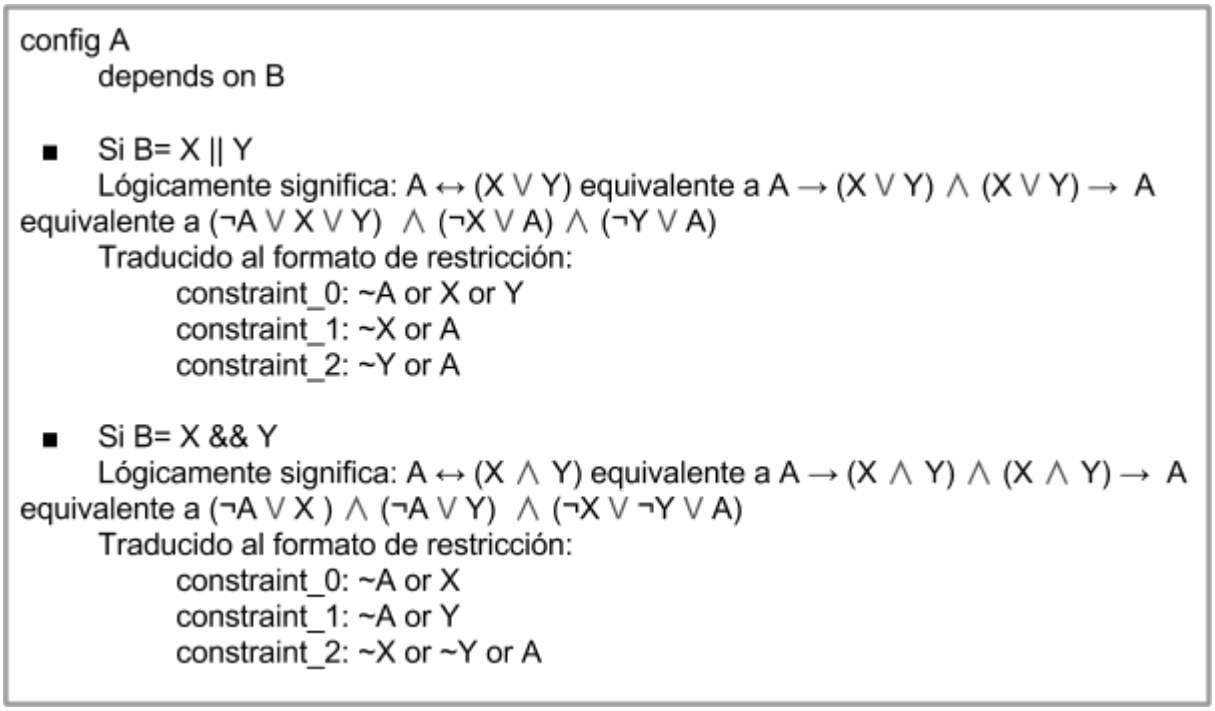

Figura 5.12. Traducción del atributo “depends on” al formato de restricción.

A continuación en las Figuras 5.13, 5.14 y 5.15 se brinda ejemplos de una porción de código Kconfig el cual posee relaciones de dependencia (depends on) y selección (select), la interpretación lógica de dichas relaciones y cómo deberían quedar las restricciones respectivamente. Con el objetivo de dar

seguimiento a cada cláusula y su correspondiente restricción, se utilizan las llamadas (1), (2) y (3).

```
config INT340X_THERMAL
        tristate "ACPI INT340X thermal drivers"
        depends on X86 && ACPI (1)
        select THERMAL_GOV_USER_SPACE (2)
        select ACPI_THERMAL_REL (3)
```

Figura 5.13. Porción de código Kconfig con cláusulas depends on y select.

```
INT340X_THERMAL ⇔ X86 ∧ ACPI (1)
INT340X_THERMAL ➡ THERMAL_GOV_USER_SPACE (2)
INT340X_THERMAL ➡ ACPI_THERMAL_REL (3)
```

Figura 5.14. Interpretación lógica para la cláusula depends on y ambas select.

```
contraint_0: ~INT340X_THERMAL or X86 (1)
contraint_1: ~INT340X_THERMAL or ACPI (1)
contraint_2: ~X86 or ~ACPI or X86INT340X_THERMAL (1)
contraint_3: ~INT340X_THERMAL or THERMAL_GOV_USER_SPACE (2)
contraint_4: ~INT340X_THERMAL or ACPI_THERMAL_REL (3)
```

Figura 5.15. Restricciones para modelo de características.

Análisis de Variabilidad

Como se dijo anteriormente, las funcionalidades soportadas para el análisis de la variabilidad de líneas de producto de software son: Reconocer inconsistencias en SPL, Determinar características muertas en SPL, Determinar características opcionales en SPL, Validar producto ingresado, Obtener cantidad de los productos derivables en SPL, Validar configuración parcial de un producto y Obtener cantidad de productos derivables de una configuración parcial.

Para esto se utilizó la biblioteca de código abierto SPLAR, la cual brinda soporte para el razonamiento a la herramienta SPLOT (http://www.splot-research.org/) [10] mencionada en la Introducción (sección 1.). Esta biblioteca está basada en un solver para el razonamiento de satisfacibilidad lógica (SAT) y permitió obtener de manera directa una porción de la funcionalidad requerida (reconocimiento de características muertas, características opcionales y obtención de cantidad de productos derivables), algo común en el ambiente de análisis de SPLs. Para obtener el resto de las funcionalidades, se tuvo que modificar esta biblioteca y hacer un uso combinado de las funciones ofrecidas, con el fin de producir la funcionalidad deseada mediante la interacción de las mismas. De todos modos, la reusabilidad del código permitió disminuir la cantidad de desarrollo planificado, por lo que más adelante, en el product backlog se puede constatar la diferencia entre el tiempo estimado versus el tiempo realmente requerido para la obtención de este conjunto de funcionalidades.

Arquitectura final del sistema

La herramienta implementada tiene un formato “Desktop” y, al haber sido desarrollada en Java, puede ser desplegada en diversos Sistemas Operativos.
A partir del uso de la tecnología JavaFX, se pudo separar la implementación de la Interfaz de usuario de la lógica de la aplicación. En la Figura 5.16 se ilustra lo dicho anteriormente. Es importante remarcar que dentro de la lógica (caja derecha de la Figura 5.16) se puede encontrar otro estilo arquitectónico, el ilustrado en la Figura 5.1.

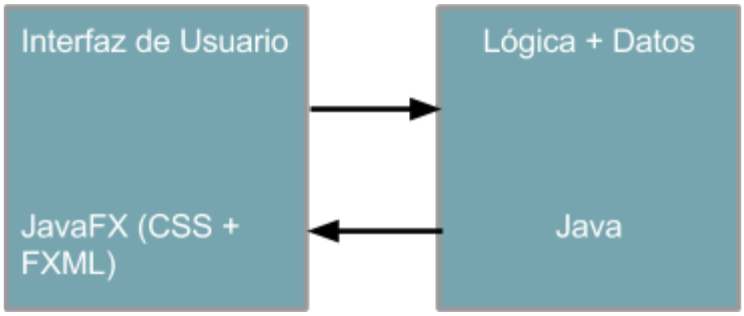

Figura 5.16. Arquitectura de la aplicación.

6. Ejecución del proyecto

6.A. Etapa de investigación

Para la ejecución de esta etapa se dispuso de 4 semanas, representando cada semana un total de 12 horas repartidas en 3 días, por integrante. El objetivo de la misma fue de disponer suficiente conocimiento a la hora de encarar la etapa de desarrollo. Teniendo en cuenta el carácter de investigación de este proyecto final, la salida de esta etapa representa un buen porcentaje del éxito o fracaso del mismo.

La información recabada puede ser dividida en dos amplias áreas:

- Temática del proyecto.
- Tecnologías de desarrollo y soporte para el mismo.

Dentro de la temática del proyecto, se realizó una profunda investigación sobre:

- Líneas de productos de software, su significado, utilización y formas de representación.
- El lenguaje Kconfig: Modo de utilización, propósitos, sintaxis, semántica y componentes del mismo. A partir de esto se obtuvo un modelo conceptual de Kconfig. Además, con el objetivo de recopilar una gran cantidad de archivos Kconfig en un único archivo Kconfig, se implementó un simple programa en Java el cual accede a los repositorios de Kconfig y concatena una serie de archivos siguiendo los links contenidos en los mismos. Por ejemplo, para el kernel de Linux se obtuvo un archivo final compuesto por alrededor de 50 mil líneas de código Kconfig.
- Concepto de variabilidad e información adicional de interés para analizar y obtener como salida de la herramienta propuesta.
- Modelos de características, junto con su sintaxis y semántica.
- Métodos para la resolución y obtención de la información del análisis de variabilidad (SAT solvers, programación matemática).

- Traducción de modelos definidos en KConfig a modelos de características.

Las salidas de esta etapa están representadas en su mayoría en la sección de Terminología de este informe (Sección 2.).

Respecto a las Tecnologías para el desarrollo, fue necesaria la búsqueda de herramientas/metodologías y posteriormente la lectura de documentación para la cobertura de las siguientes necesidades:

- Análisis sintáctico y semántico del lenguaje KConfig: desde el comienzo del proyecto, para la realización de estas funcionalidades, la herramienta ANTLR4 fue propuesta por el director/cliente debido a su flexibilidad para la integración con un amplio conjunto de lenguajes y su facilidad para generar un parser. De todos modos, fue mandatorio investigar su documentación para un óptimo uso y un mayor aprovechamiento de la funcionalidad disponible. Además de haber tenido que aprender sobre la descripción de las expresiones regulares, se dedicó tiempo a entender el código generado por dicha herramienta y la posibilidad de adaptarlo a las necesidades del equipo.
- Diseño y desarrollo de interfaces de usuario: de manera inicial, los conocimientos en el desarrollo de interfaces de usuario en Java del equipo estaban limitados a la utilización de las bibliotecas Swing y AWT. Se detectaron ciertos problemas con el empleo de estas bibliotecas, como ser una pobre performance, una complicada estructura y su inherente complejidad, impactando en un mayor tiempo requerido para el desarrollo y/o mantenimiento de la herramienta. Como solución a esto, se detectó la existencia de la tecnología JavaFX, la cual permite elaborar interfaces de usuario empleando un enfoque orientado a la web, mediante hojas de estilo. El empleo de JavaFX permitió la confección de interfaces más vistosas, más veloces y con una estructura menos compleja.
- Solver para realizar el análisis de los modelos de características: Dentro de las metodologías candidatas a utilizar para la obtención de la

información de variabilidad se encontraban los solvers para resolver programas matemáticos, programación por restricciones y satisfacibilidad proposicional (SAT). Analizando los métodos existentes para el análisis de modelos de características y las herramientas disponibles para el análisis de variabilidad de SPL, se decidió por la utilización de la biblioteca SPLAR, la cual es el solver empleado por la herramienta SPLOT (http://www.splot-research.org/) [10]. Es importante mencionar que la biblioteca está basada en un solver SAT (SAT4j), y es de código abierto al igual que SAT4j. Ambas se encuentran disponible para su descarga en la web.

Además, es necesario destacar que en esta etapa se decidió realizar el maquetado de las interfaces de usuario con el objetivo de poner en común las ideas dentro del equipo de desarrollo en lo que respecta a las interfaces de usuario. En la sección A del Anexo de este informe el lector podrá encontrar las maquetas obtenidas.

6.B. Etapa de desarrollo

Antes de proceder con la evolución de las iteraciones de la etapa de desarrollo de este proyecto, se deben definir los valores correspondientes a Story Points y Task Points, tal como lo expresado en el plan del proyecto. Particularmente los Story Points serán de una semana de trabajo de tres días y, un Task Point contempla un día de trabajo de 4 horas. En síntesis, un Story Point equivale a tres Task Points (1 Story Point=3 Task Points).

El Product Backlog inicial se detalla en la Tabla 6.B.1, con las historias de usuario a implementar priorizadas según las necesidades detectadas por el equipo de desarrollo y el Director del proyecto. En la sección B del Anexo de este informe se detallan las historias de usuario con su posterior refinamiento.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo

1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	3	Media
5	Reconocer inconsistencias en SPL	3	Media
6	Determinar características muertas en SPL	3	Media
7	Determinar características opcionales en SPL	3	Media
8	Validar producto ingresado	3	Media
9	Obtener cantidad de productos derivables en SPL	3	Media
10	Validar configuración parcial de un producto	1	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	Baja

Tabla 6.B.1. Product Backlog inicial.

A continuación en la Tabla 6.B.2 se muestra la planificación de las iteraciones con sus respectivas historias y objetivos, la cual fue incluída en el plan correspondiente a este proyecto final.

Iteración	Inicio	Fin	Historias	Story Points	Objetivos de la iteración
Iteración 1	03/10/2016	23/10/16	2,3	6	Análisis sintáctico y semántico de un archivo KConfig
Iteración 2	24/10/2016	13/11/16	1,4	6	Gestión de modelos y traducción a modelo de características
Iteración 3	14/11/2016	04/12/16	5,6	6	Validar SPL (a partir de la detección de inconsistencias) y determinar características muertas
Iteración 4	05/12/16	25/12/16	7,8	6	Determinar características opcionales en un modelo KConfig y validar la configuración de un producto ingresado
Iteración 5	26/12/16	15/01/17	9,10,11	6	Poder deducir la cantidad de productos derivables de una SPL,validar un producto a partir de la configuración parcial ingresada y deducir cantidad de productos derivables de una configuración parcial de una SPL

Tabla 6.B.2. Planificación de las iteraciones.

Para dar una perspectiva resumida de la planificación de las iteraciones se muestra un diagrama Gantt en la Figura 6.B.3, donde se muestran las iteraciones con sus respectivas fechas de inicio y fin.

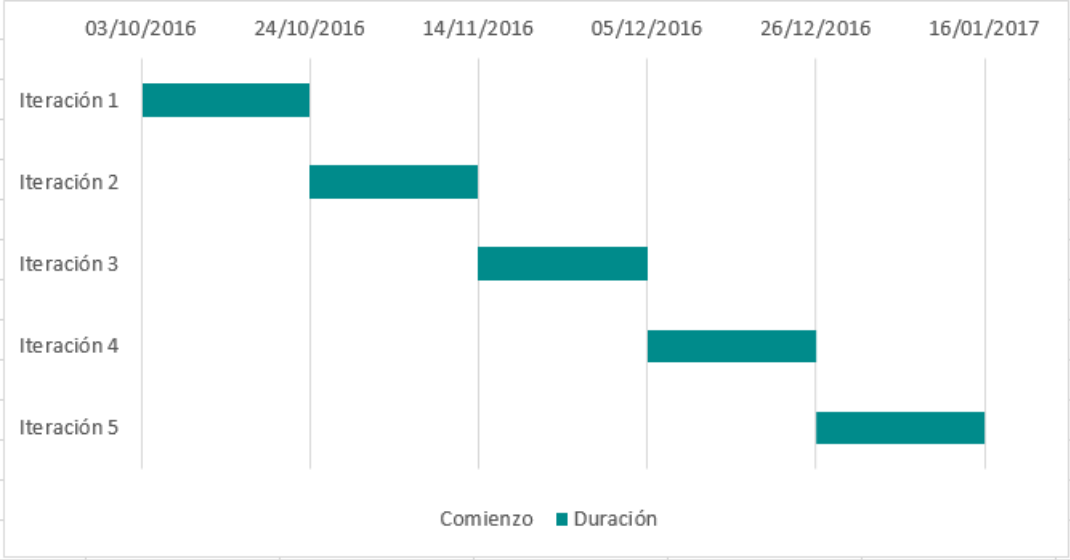

Figura 6.B.3. Distribución temporal de las iteraciones.

En las siguientes secciones se presenta el desarrollo de las iteraciones. En cada sección perteneciente a una iteración se podrá encontrar un resumen de la misma, el cual muestra el período de ejecución y la cantidad de historias y Story Points estimados vs. realizados sumado a las tareas realizadas e información relevante sobre las mismas. En sub apartados posteriores se mostrarán los Burndown Chart correspondientes a la iteración (Sprint Burndown Chart) y al Producto (Product Burndown Chart), seguidos por el Product Backlog actualizado a la finalización de la iteración, comentarios pertinentes a la gestión de riesgos y finalmente se incluirán los comentarios y conclusiones.

6.B.1 Desarrollo de la iteración 1

En la siguiente Tabla 6.B.1.1 se muestra un resumen de la iteración:

Resumen de iteración 1	Estimado	Realizado
Período de ejecución	03/10/2016 - 23/10/2016	03/10/2016 - 23/10/2016

Cantidad de historias	2	1 completa + 1 parcial
Story Points	6	2
Revisión de la iteración: Lunes, 24 de octubre de 2016		

Tabla 6.B.1.1. Resumen de la primera iteración.

Antes de proceder con la explicación de esta primera iteración, se muestra en la Tabla 6.B.1.2 un listado de las tareas correspondientes a las historias desarrolladas dentro de los plazos de la iteración, junto con los task points asignados a las mismas. Las tareas junto con sus id se corresponden con los presentados en el plan de este proyecto.

Id historia	Id tarea	Task points	Nombre
2	4	6	Lógica: Parsear archivo KConfig
3	5	12	Lógica: Instanciar modelo KConfig a modelo de Clases

Tabla 6.B.1.2. Tareas planificadas para la primer iteración.

El lector a simple vista se podría preguntar por qué no se decidió desdoblar éstas tareas en un subconjunto de tareas de modo de obtener una mayor granularidad, con el objetivo de un mayor control. Se decidió continuar así debido a la imposibilidad de dividir las misma. Esta imposibilidad fue debido a la naturaleza de las historias, las cuales eran constituidas por funcionalidades puramente lógicas, sin interfaces de usuario ni persistencia.

Al estar constituida por una gran cantidad de Tasks Points la tarea 5 (Lógica: Instanciar modelo KConfig a modelo de Clases), no se pudo reflejar el avance de la misma logrado dentro de ésta iteración. Aunque no pudo ser finalizada, sí se completó un buen porcentaje de la misma.

Gráficos de avance

A continuación se presentan el Sprint Burndown Chart en la Figura 6.B.1.3 y el Product Burndown Chart en la Figura 6.B.1.4. Los mismos reflejan lo

anteriormente comentado, la realización completa de la tarea 4 (Lógica: Parsear archivo KConfig) y la postergación de la finalización de la tarea 5 (Lógica: Instanciar modelo KConfig a modelo de Clases).

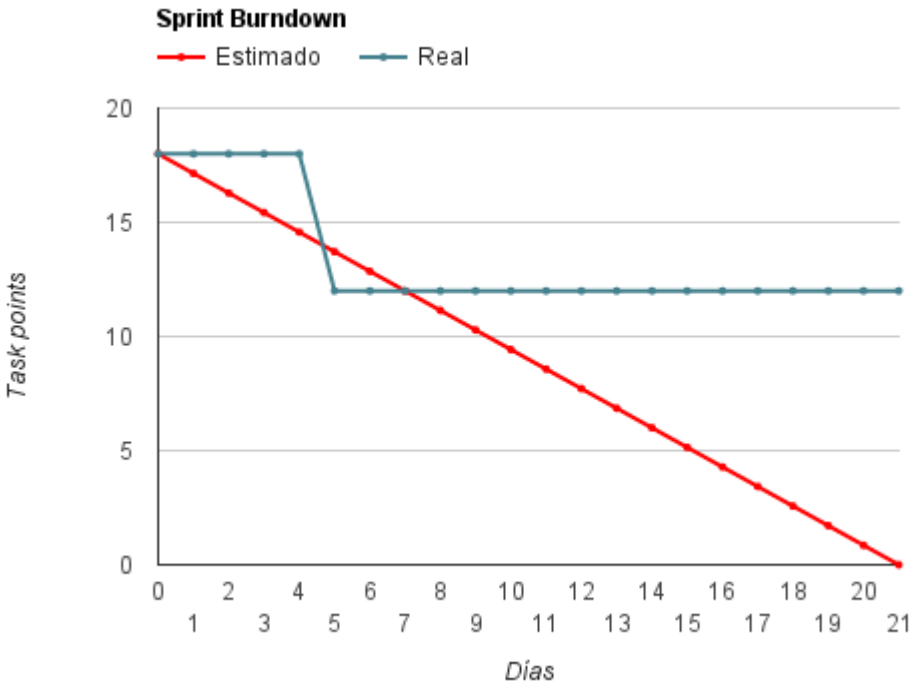

Figura 6.B.1.3. Sprint Burndown Chart correspondiente a la primera iteración.

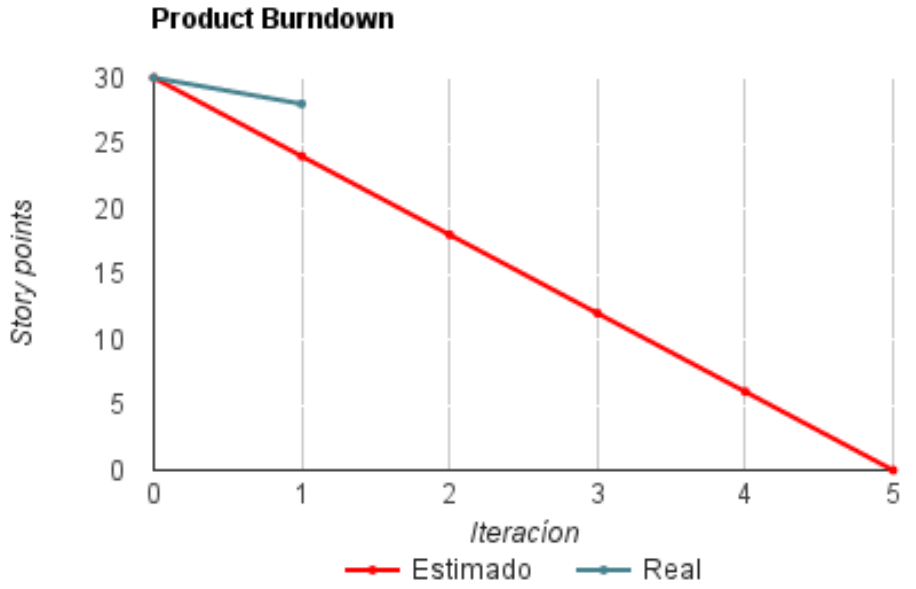

Figura 6.B.1.4. Product Burndown Chart correspondiente a la primera iteración.

Product backlog actualizado

A continuación se muestra en la Tabla 6.B.1.4 el Product Backlog correspondiente a la finalización de la iteración, donde se detallan las historias finalizadas, las desarrolladas parcialmente y las pendientes.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	3	Media
5	Reconocer inconsistencias en SPL	3	Media
6	Determinar características muertas en SPL	3	Media
7	Determinar características opcionales en SPL	3	Media
8	Validar producto ingresado	3	Media
9	Obtener cantidad de productos derivables en SPL	3	Media
10	Validar configuración parcial de un producto	1	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	Baja

Historias de usuario desarrolladas.

Historias de usuario desarrolladas parcialmente.

Historias de usuario pendientes.

Tabla 6.B.1.4. Product Backlog actualizado.

Gestión de riesgos

Al realizar la revisión del sprint, como indica el plan de gestión de riesgo, se advirtió que la historia 3 (Instanciar modelo KConfig a modelo de clases) fue subestimada. Esto se definió en el riesgo R18− "Estimaciones de tiempo incorrectas en la etapa de planificación", el cual fue ubicado en la calificación de Poco Riesgoso debido a su leve/mediano impacto. Como este riesgo no está

dentro de las dos categorías más riesgosas definidas en el plan, no se dispone de una acción correctiva de antemano. Entonces, como acción correctiva para tal riesgo se propone re calcular las estimaciones, pero, por ser el primer Sprint se considera una acción precipitada debido a la naturaleza de las historias de usuario, las cuales no requieren de un esfuerzo lineal para su culminación, por lo que se esperará para re estimar las historias. Además, se detectó la activación del riesgo R5− "La velocidad de desarrollo es menor a la planificada". Este riesgo está estrechamente relacionado con el riesgo anterior, es decir, el R18 da origen al R5, por lo que no se define una acción correctiva por el momento.

Comentarios y conclusiones

Como comentarios sobre la estimación de las historias de usuario, se puede decir que la historia 2 (Parsear archivo KConfig) pudo ser finalizada dentro del plazo preestablecido debido a que en la etapa de investigación se encontró una expresión regular dentro de la documentación sobre Kconfig la cual describe el mismo. De todos modos, fue necesario realizar retoques sobre la expresión regular debido a que no estaba completa ni contemplaba todas las opciones. Con respecto a la historia 3 (Instanciar modelo KConfig a modelo de clases), la misma no pudo ser finalizada a tiempo debido al desconocimiento de la herramienta ANTLR4 por parte del equipo. De todos modos, no resta mucho trabajo para su finalización. Se estima un story point para su finalización.

Dentro de los puntos a tener en cuenta y eventualmente reforzar para las próximas iteraciones se encuentra la precisión de las estimaciones de las historias y tareas. Esta iteración en particular no puede ser considerada como parámetro debido a la naturaleza de las historias desarrolladas.

6.B.2 Desarrollo de la iteración 2

En la siguiente Tabla 6.B.2.1 se muestra un resumen de la iteración:

Resumen de iteración 2	Estimado	Realizado

Período de ejecución	24/10/2016 - 13/11/16	24/10/2016 - 13/11/16
Cantidad de historias	2	1 completa + 1 parcial
Story Points	6	7
Revisión de la iteración: Lunes, 14 de noviembre de 2016		

Tabla 6.B.2.1. Resumen de la segunda iteración.

Antes de proceder con la explicación de esta segunda iteración, se muestra en la Tabla 6.B.2.2 un listado de las tareas correspondientes a las historias desarrolladas dentro del plazo, junto con los Task Points asignados a las mismas. Las tareas junto con sus id se corresponden con los presentados en el plan de este proyecto.

Id historia	Id tarea	Task points	Nombre
3	5	12	Lógica: Instanciar modelo KConfig a modelo de Clases
1	1	2	Interfaz: Gestionar modelo KConfig
1	2	4	Lógica: Gestionar modelo KConfig
1	3	3	Persistencia: Gestionar modelo KConfig

Tabla 6.B.2.2. Tareas planificadas para la segunda iteración.

Previo al comienzo del desarrollo de la segunda iteración se decidió hacer una re planificación considerando la demora surgida en cuanto a la tarea 5 (Lógica: Instanciar modelo KConfig a modelo de Clases) correspondiente a la historia 3 (Instanciar modelo KConfig a modelo de clases) y la complejidad esperada para la historia 4 (Traducir modelo de clases a modelo SPLOT), la cual según la planificación inicial debía ser ejecutada en esta iteración. Contemplando estos argumentos, se resolvió dedicar la segunda iteración por completo a la finalización de la historia pendiente y al desarrollo de la historia 1 (Gestionar (Alta/Baja) modelo Kconfig).

Gráficos de avance

A continuación se presentan el Sprint Burndown Chart en la Figura 6.B.2.3 y el Product Burndown Chart en la Figura 6.B.2.4. Los mismos reflejan la finalización de la tarea 5 (Lógica: Instanciar modelo KConfig a modelo de Clases), la cual había quedado pendiente y la ejecución por completo de las tareas 1 (Interfaz: Gestionar modelo KConfig), 2 (Lógica: Gestionar modelo KConfig) y 3 (Persistencia: Gestionar modelo KConfig). En el Sprint Burndown Chart se puede observar cómo se han completado más Story Points de lo planificado para una iteración, esto se debe a la inclusión de los Story Points no contemplados para la iteración anterior.

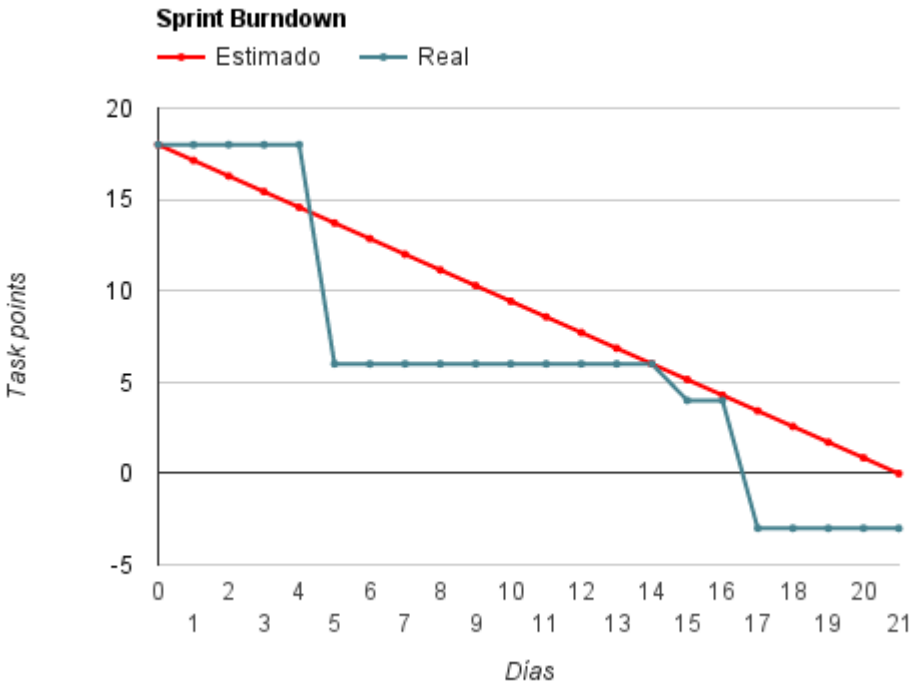

Figura 6.B.2.3. Sprint Burndown Chart correspondiente a la segunda iteración.

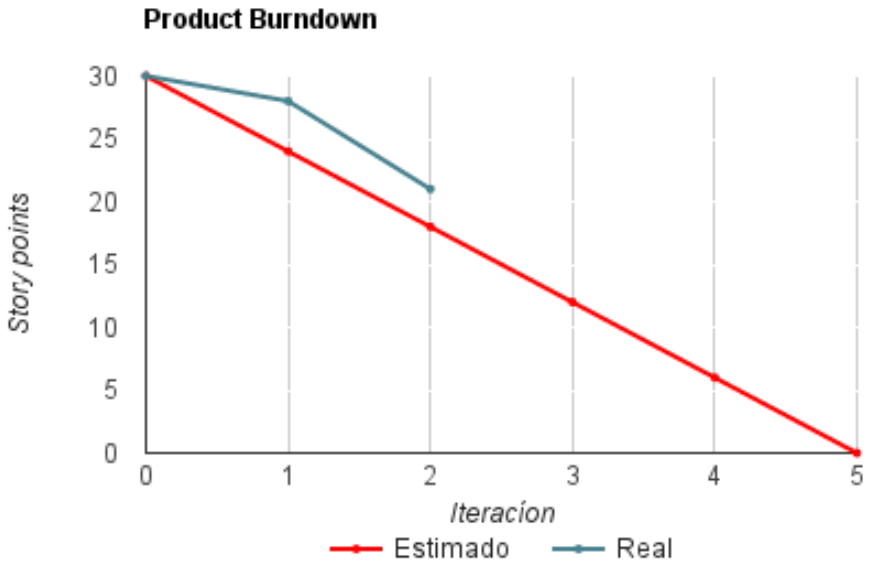

Figura 6.B.2.4. Product Burndown Chart correspondiente a la segunda iteración.

Product backlog actualizado

A continuación se muestra en la Tabla 6.B.2.5 el Product Backlog correspondiente a la finalización de la iteración, donde se detallan las historias finalizadas y las pendientes.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	3	Media
5	Reconocer inconsistencias en SPL	3	Media
6	Determinar características muertas en SPL	3	Media
7	Determinar características opcionales en SPL	3	Media
8	Validar producto ingresado	3	Media
9	Obtener cantidad de productos derivables en SPL	3	Media
10	Validar configuración parcial de un producto	1	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	Baja

Historias de usuario desarrolladas.

Historias de usuario pendientes.

Tabla 6.B.2.5. Product Backlog actualizado.

Gestión de riesgos

En esta iteración no se han activado riesgos, aunque se sufrieron las consecuencias del impacto del R18 detectado en la iteración anterior, llevando a un desplazamiento de las historias en cuanto a las fechas inicialmente pactadas.

Comentarios y conclusiones

Como comentarios sobre la estimación de las historias de usuario se puede decir que finalmente, la historia 3 (Instanciar modelo KConfig a modelo de clases) demandó un tiempo adicional levemente mayor a una semana la planificación inicial, lo cual representa una subestimación de 2 Story Points.

Con respecto a la historia 1 (Gestionar (Alta/Baja) modelo KConfig), la misma pudo ser realizada en el plazo preestablecido, aunque la estimación de las tareas fue equívoca debido a que la tarea 1 (Interfaz: Gestionar modelo KConfig) demandó más de lo esperado y, las tareas 2 y 3 (Lógica+Persistencia: Gestionar modelo KConfig) resultaron ser más simples de lo planificado, por lo que las mismas fueron compensadas entre sí. El motivo de la demora de la tarea 1 se justifica por la utilización de la tecnología JavaFX, algo nuevo para el equipo de desarrollo.

Gracias a que se pudo finalizar con estas dos historias antes de las tres semanas destinadas a la iteración, sobraron unos días para comenzar a esbozar una idea para la realización de la tarea 7 (Lógica: Traducir modelo de clases a modelo SPLOT) correspondiente a la próxima historia a realizar.

Concluyendo, para la realización de esta iteración se hizo énfasis en una mejor planificación de los tiempos y, con el objetivo de no volver a dejar historias marcadas como parcialmente implementadas, se decidió por una estimación más conservadora, lo cual permitió obtener un incremento de una mayor calidad,

algo fundamental para un proyecto de este tipo, el cual puede que requiera ser modificado por becarios del Director/Dueño del producto. Desde esta iteración en adelante se preferirán las estimaciones más conservadoras, aunque a mediano plazo pueda requerir una iteración más para la finalización del proyecto.

Como comentario final, se desea agregar que la realización de la primer interfaz de usuario seguramente afectará la estimación de las próximas tareas relacionadas, ya que servirá como base tanto estructural como de conocimiento para el desarrollo de las próximas interfaces de usuario. Por ende, puede ser que la estimación inicial se vea modificada nuevamente.

6.B.3 Desarrollo de la iteración 3

En la siguiente Tabla 6.B.3.1 se muestra un resumen de la iteración:

Resumen de iteración 1	Estimado	Realizado
Período de ejecución	14/11/2016 - 04/12/2016	14/11/2016 - 04/12/2016
Cantidad de historias	2 en el plan inicial, 1 luego de la iteración anterior	1 completa
Story Points	6	6
Revisión de la iteración: Lunes, 5 de enero de 2016		

Tabla 6.B.3.1. Resumen de la tercera iteración.

En la Tabla 6.B.3.3 se muestran la tareas correspondientes a la única historia desarrollada dentro de los plazos de la iteración, junto con los Task Points asignados a las mismas. Las tareas junto con sus id se corresponden con los presentados en el plan de este proyecto.

En base a lo decidido en la iteración anterior sobre realizar estimaciones más conservadoras al momento de ejecutar historias sobre las cuales se tenga una mayor incertidumbre sobre su desarrollo y, juzgando la complejidad de la historia 4 (Traducir modelo de clases a modelo SPLOT) en base a lo sucedido con la historia 3 (Instanciar modelo KConfig a modelo de clases), se decidió dedicar

esta iteración por completo para la ejecución de la historia 4, lo que implica una re estimación de la misma, quedando con 6 Story Points y 18 Task Points. A su vez, esta re estimación también impacta sobre los Tasks Points asignados a las tareas. En la siguientes Tablas 6.B.3.2 y 6.B.3.3 se refleja lo anteriormente mencionado: el Product Backlog luego de la re estimación de la historia 4 y el impacto en las tareas de la misma.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
4	Traducir modelo de clases a modelo SPLOT	6	Media
5	Reconocer inconsistencias en SPL	3	Media
6	Determinar características muertas en SPL	3	Media
7	Determinar características opcionales en SPL	3	Media
8	Validar producto ingresado	3	Media
9	Obtener cantidad de productos derivables en SPL	3	Media
10	Validar configuración parcial de un producto	1	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	Baja

Tabla 6.B.3.2. Product Backlog con historia 4 re estimada al principio de la iteración.

Id historia	Id tarea	Task points	Nombre
4	6	1	Interfaz: Traducir modelo de clases a modelo SPLOT
4	7	16	Lógica: Traducir modelo de clases a modelo SPLOT
4	8	1	Persistencia: Traducir modelo de clases a modelo SPLOT

Tabla 6.B.3.3. Tareas planificadas para la tercera iteración.

Recordando lo sucedido en la primera iteración sobre la imposibilidad de separar una tarea compuesta por muchos Tasks Points en más subtareas, se puede detectar la misma situación respecto a la tarea 7 (Lógica: Traducir modelo de clases a modelo SPLOT). De antemano no se pudo estimar cuántos Tasks

Points hubiesen sido necesario, pero sí se pudo estimar los Tasks Points necesarios para la ejecución de las otras tareas debido a la experiencia ganada con las iteraciones pasadas, por lo que se decidió destinar la cantidad de Tasks Points restantes a la tarea 7. Además, se acordó que de haber finalizado antes de lo estimado, se utilizaría el tiempo restante para la refactorización del código.

Gráficos de avance

A continuación se presentan el Sprint Burndown Chart en la Figura 6.B.3.4 y el Product Burndown Chart en la Figura 6.B.3.5. Los mismos reflejan la ejecución de la historia 4 con sus respectivas tareas. Se desea agregar que aunque la historia 4 (Traducir modelo de clases a modelo SPLOT) haya sido re estimada, los puntos asignados inicialmente se mantienen con el objetivo de mostrar un Product Burndown Chart consistente para todas las iteraciones.

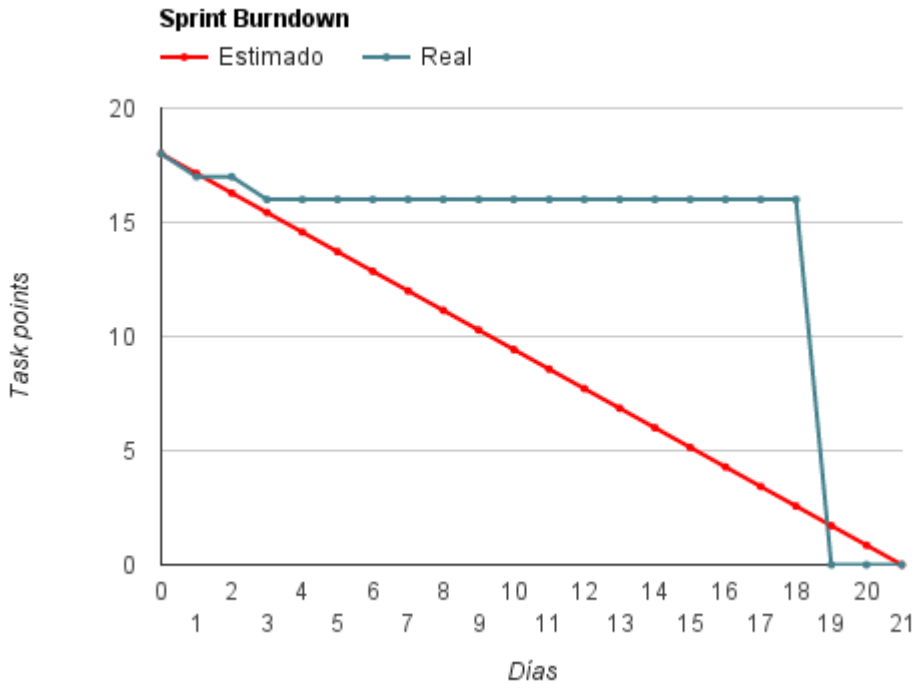

Figura 6.B.3.4. Sprint Burndown Chart correspondiente a la tercera iteración.

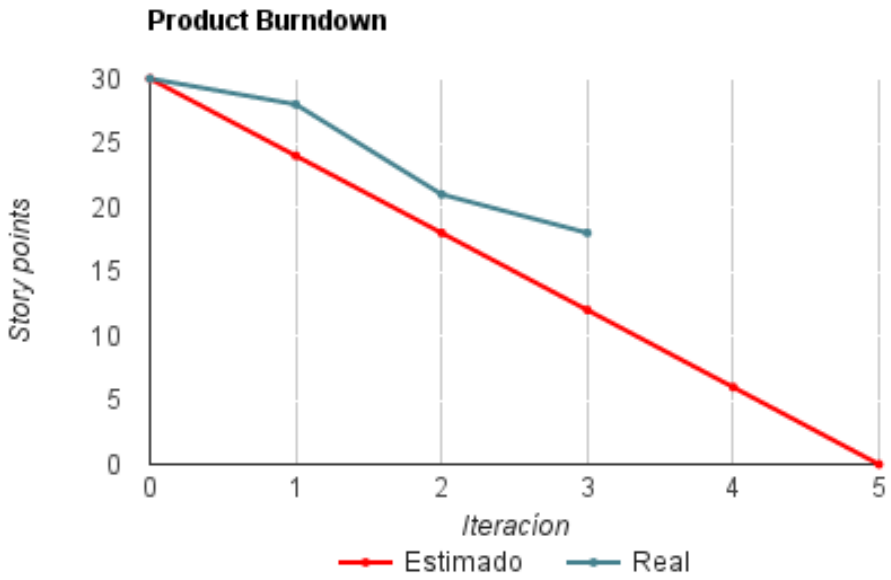

Figura 6.B.3.5. Product Burndown Chart correspondiente a la tercera iteración.

Product backlog actualizado

A continuación se muestra en la Tabla 6.B.3.6 el Product Backlog correspondiente a la finalización de la iteración, donde se detallan las historias finalizadas y las pendientes.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	6	Media
5	Reconocer inconsistencias en SPL	3	Media
6	Determinar características muertas en SPL	3	Media
7	Determinar características opcionales en SPL	3	Media
8	Validar producto ingresado	3	Media
9	Obtener cantidad de productos derivables en SPL	3	Media
10	Validar configuración parcial de un producto	1	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	Baja

Historias de usuario desarrolladas.

Historias de usuario pendientes.

Tabla 6.B.3.6. Product Backlog actualizado.

Gestión de riesgos

Con el motivo de tener estimaciones más ajustadas a la realidad del proyecto y como se mencionó anteriormente, se ha reestimado la historia 4. Esta acción ha llevado a activar el R18− "Estimaciones de tiempo incorrectas en la etapa de planificación" y por ende, el R5− "La velocidad de desarrollo es menor a la planificada" tal como lo sucedido en la primera iteración. La principal diferencia con la primera iteración es que para esta iteración el equipo se anticipó al R18 y pudo realizar una re estimación en base a lo sucedido, aunque fue inevitable la activación del mismo ya que trata sobre estimaciones incorrectas en la etapa de planificación. Además, en este punto de ejecución del proyecto el equipo se encuentra atrasado el tiempo equivalente a una iteración, por lo que el R5 se encuentra latente y la acción correctiva para el mismo sería una iteración adicional.

Comentarios y conclusiones de la iteración

Como conclusión para esta iteración, se desea destacar la re estimación hecha al comienzo de la misma. Mediante la misma se ajustaron los tiempos en base a la experiencia recolectada en las iteraciones anteriores, permitiendo al equipo continuar con una planificación más realista, aceptando la posibilidad de tener que realizar una iteración adicional.

Además de los beneficios mencionados anteriormente se desea agregar que a través de esta re estimación se logró un incremento de funcionalidad más organizado, puesto que aunque la realización de la historia 4 (Traducir modelo de clases a modelo SPLOT) demandó menos tiempo del estimado, fue necesario hacer modificaciones a la funcionalidad de la tarea 3 (Persistencia: Gestionar

modelo KConfig) para adecuar sus salidas de acuerdo a lo requerido por la tarea 7 (Lógica: Traducir modelo de clases a modelo SPLOT).

Con respecto al desarrollo de la interfaz de usuario, tarea 6, demandó menos tiempo debido a la previa realización de una interfaz similar, brindando una base para la estructura y el estilo, los cuales pueden ser fácilmente reutilizados. Una gran ventaja detectada a partir de la utilización de la tecnología JavaFX.

6.B.4 Desarrollo de la iteración 4

En la siguiente Tabla 6.B.4.1 se muestra un resumen de la iteración:

Resumen de iteración 4	Estimado	Realizado
Período de ejecución	05/12/2016 - 25/12/2016	05/12/2016 - 25/12/2016
Cantidad de historias	2	4
Story Points	6	12 (según planificación inicial) y 6 luego de re estimación
Revisión de la iteración: Lunes, 26 de diciembre de 2016		

Tabla 6.B.4.1. Resumen de la cuarta iteración.

Antes de comenzar con el desarrollo de esta iteración se decidió analizar el modo de obtención de las funcionalidades restantes del Product Backlog, las cuales corresponden al analizador de la herramienta propuesta.

En la etapa de Investigación se acordó la utilización de la biblioteca SPLAR, aunque no se profundizó en el cómo obtener las diversas funcionalidades requeridas para la realización del análisis de una SPL. Tras haber descargado y analizado el código fuente de esta biblioteca se detectó un exceso en las estimaciones realizadas ya que la adaptación de SPLAR para la obtención de las funcionalidades correspondientes al analizador de la herramienta iba a ser más sencillo de lo esperado. Además de esto, a partir del diseño de las interfaces de usuario, el cual fue plasmado en las maquetas, se tuvo en cuenta que muchas de las historias restantes compartirán la misma interfaz.

Luego de esto se decidió realizar una re estimación de las historias restantes para tener una visión más real de la porción de trabajo restante del proyecto. En la siguiente Tabla 6.B.4.2 se muestra el Product Backlog con la estimación inicial y posterior a su modificación, hecha al comienzo de esta iteración.

Product Backlog				
ID historia	Nombre historia	Estimación anterior (SP)	Estimación actual (SP)	Prioridad de desarrollo
5	Reconocer inconsistencias en SPL	3	2	Media
6	Determinar características muertas en SPL	3	1	Media
7	Determinar características opcionales en SPL	3	1	Media
8	Validar producto ingresado	3	2	Media
9	Obtener cantidad de productos derivables en SPL	3	1	Media
10	Validar configuración parcial de un producto	1	2	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	1	Baja

Tabla 6.B.4.2. Product Backlog re estimado al principio de la iteración.

Luego, para la ejecución de esta iteración se seleccionaron las primeras 4 historias, sumando un total de 6 Story Points, velocidad definida para cada iteración. En la Tabla 6.B.4.3 se muestra un listado de las tareas correspondientes a las historias anteriormente mencionadas, junto con los task points asignados a las mismas. Las tareas junto con sus id se corresponden con los presentados en el plan de este proyecto.

Id historia	Id tarea	Task points	Nombre
4	9	2	Interfaz: Reconocer inconsistencias en SPL
4	10	4	Lógica: Reconocer inconsistencias en SPL
5	11	2	Interfaz: Determinar características muertas en SPL

5	12	1	Lógica: Determinar características muertas en SPL
6	13	2	Interfaz: Determinar características opcionales
6	14	1	Lógica: Determinar características opcionales
7	15	4	Interfaz: Validar producto ingresado
7	16	2	Lógica: Validar producto ingresado

Tabla 6.B.4.3. Tareas planificadas para la primer iteración.

Gráficos de avance

A continuación se presentan el Sprint Burndown Chart en la Figura 6.B.4.4 y el Product Burndown Chart en la Figura 6.B.4.5. Los mismos reflejan la completa realización de las tareas mencionadas anteriormente. Al igual que lo sucedido en la iteración 3, para mantener la consistencia de los Product Burndown Chart, se conservan los Story Points asignados en la etapa de planificación.

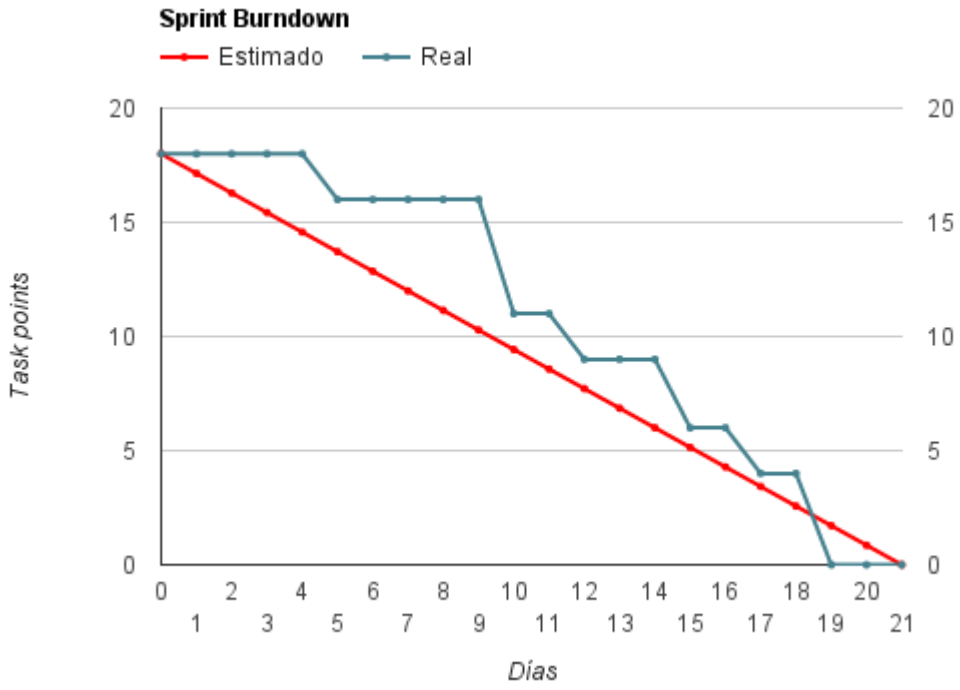

Figura 6.B.4.4. Sprint Burndown Chart correspondiente a la cuarta iteración.

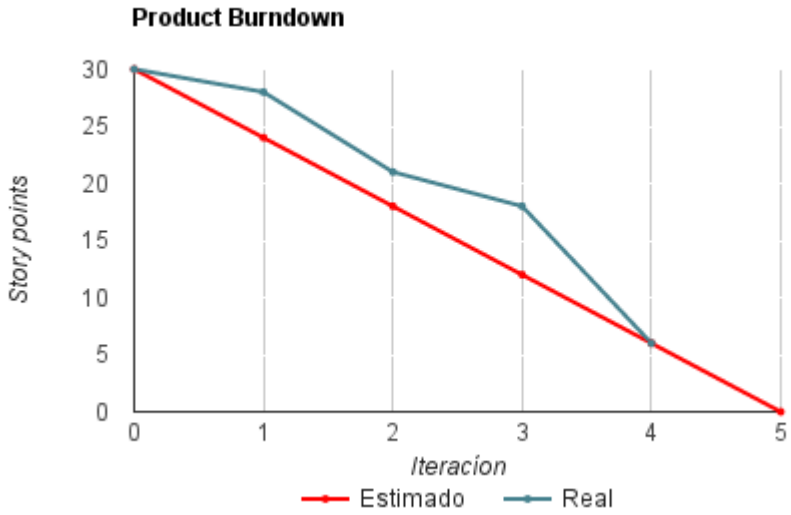

Figura 6.B.4.5. Product Burndown Chart correspondiente a la cuarta iteración.

Product backlog actualizado

A continuación se muestra en la Tabla 6.B.4.6 el Product Backlog correspondiente a la finalización de la iteración, donde se detallan las historias finalizadas y las pendientes.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	6	Media
5	Reconocer inconsistencias en SPL	2	Media
6	Determinar características muertas en SPL	1	Media
7	Determinar características opcionales en SPL	1	Media
8	Validar producto ingresado	2	Media
9	Obtener cantidad de productos derivables en SPL	1	Media
10	Validar configuración parcial de un producto	2	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	1	Baja

Historias de usuario desarrolladas.

Historias de usuario pendientes.

Tabla 6.B.4.6. Product Backlog actualizado.

Gestión de riesgos

Teniendo en cuenta la re estimación hecha a partir del análisis sobre la biblioteca SPLAR, la cual permitió agregar más funcionalidades de las originalmente planificadas en el plan del proyecto, se detectó la activación del R18− "Estimaciones de tiempo incorrectas en la etapa de planificación". En contraste con las otras iteraciones donde se activó este riesgo, en esta ocasión se pudieron ejecutar más historias, permitiendo un balance de los tiempos. Notar que la línea "real" del Product Burndown Chart coincide con la "ideal", lo que significa que en este momento se está alineado con la planificación.

Comentarios y conclusiones de la iteración

Como cierre de esta iteración, se desea comentar sobre la inesperada facilidad brindada por la biblioteca. Dentro de la misma se encontraron clases con ejemplos muy bien documentados para su uso, los cuales sirvieron para realizar de una manera muy sencilla las funcionalidades para las historias 6 (Determinar características muertas en SPL) y 7 (Determinar características opcionales en SPL). También, en la misma clase se encontraba disponible un método el cual reconocía las características comunes, por lo que se decidió incluir el mismo con el objetivo de brindar más información a quien realice el análisis. Además de indicar las características comunes y opcionales, a modo de resumen se muestra la cantidad detectada de cada grupo.

La funcionalidad de la historia 5 (Reconocer inconsistencias en SPL) también se logró a través de ésta biblioteca, aunque fue necesario detenerse mayor tiempo para analizar el modo de obtenerla. Luego de unos días investigando el código a través de depuraciones, bastó hacer un "catch" de una excepción e imprimir el contenido de una variable para mostrar al usuario la cláusula responsable de hacer un modelo inconsistente.

Para lograr la funcionalidad de la historia 8 (Validar producto ingresado), no se requirió de demasiado tiempo ni esfuerzo, ya que la documentación era bastante clara y el equipo pudo resolver qué métodos de la biblioteca utilizar rápidamente. Sumado a esto, se consideró conveniente incluir una funcionalidad mínima pero muy útil para el usuario la cual permite autocompletar un modelo partiendo de la base de características previamente seleccionadas por el usuario, entregando así un producto completo y válido.

Con respecto a las tareas de interfaz de usuario de las historias 5, 6 y 7 se desea agregar que la tarea de interfaz de la historia 5 (la cual fue comenzada primero) demandó un poco más tiempo del planificado aunque esto fue posteriormente contrarrestado con las otras tareas, ya que el esfuerzo de desarrollo de interfaz fue único para las tres. Esto era esperado de antemano.

6.B.5 Desarrollo de la iteración 5

En la siguiente Tabla 6.B.5.1 se muestra un resumen de la iteración:

Resumen de iteración 4	Estimado	Realizado
Período de ejecución	26/12/2016 - 15/01/2017	26/12/2016 - 15/01/2017
Cantidad de historias	2	3
Story Points	6	6 (según planificación inicial) y 4 luego de re estimación
Revisión de la iteración: Lunes, 16 de enero de 2017		

Tabla 6.B.5.1. Resumen de la quinta iteración.

Para la ejecución de esta iteración se tomaron las historias de usuario restantes del Product Backlog. Es importante destacar que la re estimación de las historias hecha en la iteración anterior impactó en la estimación de las tareas realizadas en esta iteración. En la Tabla 6.B.5.2 se muestra un listado de las tareas, junto con los Task Points asignados a las mismas luego de la re estimación. Las

tareas junto con sus id se corresponden con los presentados en el plan de este proyecto.

Id historia	Id tarea	Task points	Nombre
9	17	1	Interfaz: Obtener cantidad de productos derivables en SPL
9	18	2	Lógica: Obtener cantidad de productos derivables en SPL
10	19	4	Interfaz: Validar configuración parcial de un producto
10	20	2	Lógica: Validar configuración parcial de un producto
11	21	2	Interfaz: Obtener cantidad de productos derivables a partir de configuración parcial
11	22	1	Lógica: Obtener cantidad de productos derivables a partir de configuración parcial

Tabla 6.B.5.2. Tareas planificadas para la quinta iteración.

A la tarea 17 se le asignó un sólo punto de Interfaz debido a que estaba prácticamente hecha, a partir de lo obtenido en la iteración anterior. Sólo fue necesario un retoque, por lo que no se utilizó ese Task Point en su totalidad para esta historia.

Gráficos de avance

A continuación se presentan el Sprint Burndown Chart en la Figura 6.B.5.3 y el Product Burndown Chart en la Figura 6.B.5.4. Los mismos reflejan la completa realización de las tareas mencionadas anteriormente.

El lector notará en el Sprint Burndown Chart la finalización por anticipado de las actividades, empleando 2 de las 3 semanas propuestas para la iteración. Esto se

debe a la re estimación hecha, la cual permitió ahorrar 2 Story Points.

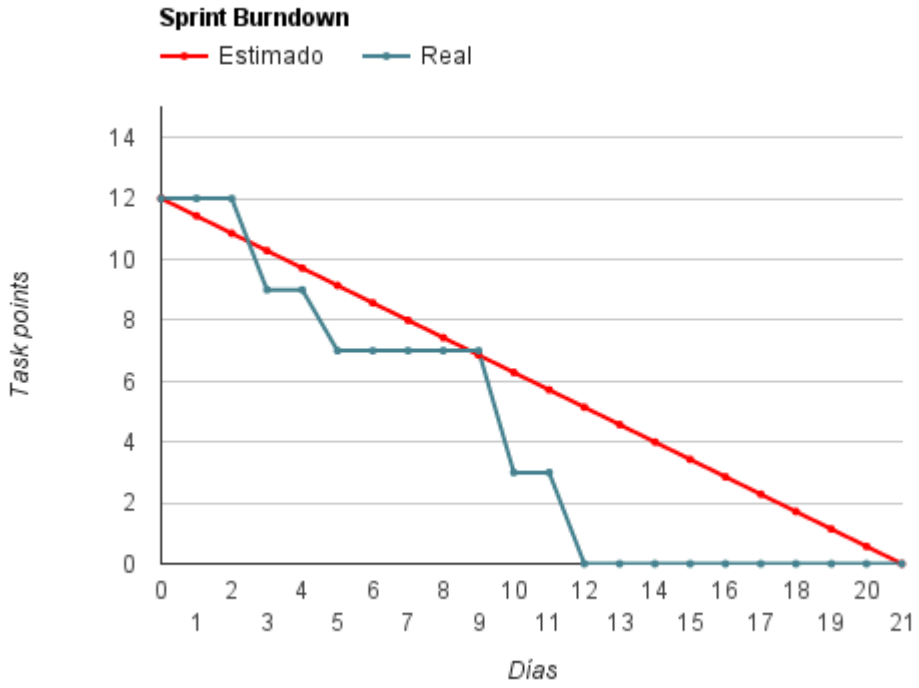

Figura 6.B.5.3. Sprint Burndown Chart correspondiente a la quinta iteración.

Al igual que lo sucedido en la iteración 3 y 4, para mantener la consistencia de los Product Burndown Chart, se conservan los Story Points asignados en la etapa de planificación.

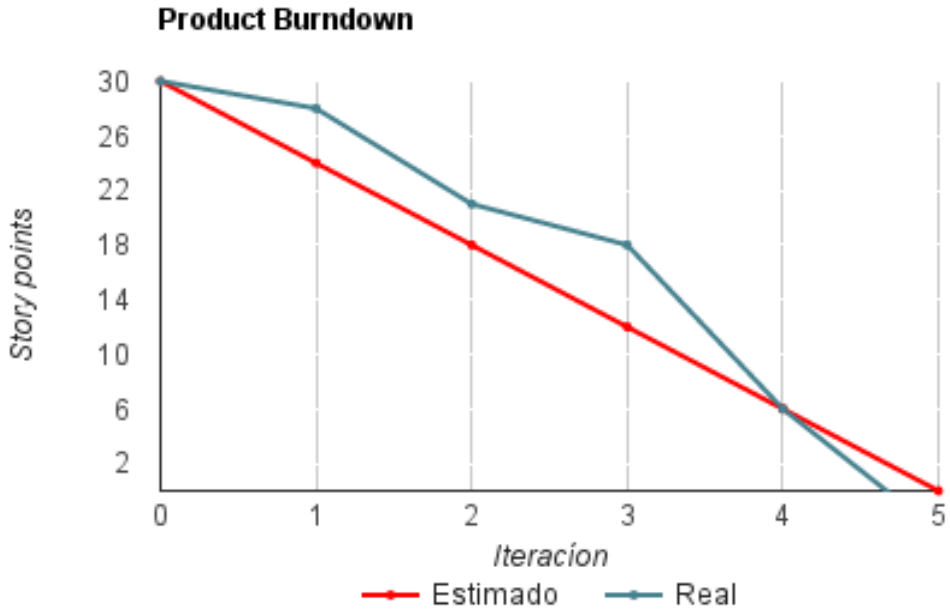

Figura 6.B.5.4. Product Burndown Chart correspondiente a la quinta iteración.

Product backlog actualizado

A continuación se muestra en la Tabla 6.B.5.5 el Product Backlog correspondiente a la finalización de la iteración, donde se detallan las historias realizadas.

Product Backlog			
ID historia	Nombre historia	Estimación Story Points	Prioridad de desarrollo
1	Gestionar (Alta/Baja) modelo KConfig	3	Alta
2	Parsear archivo KConfig	2	Alta
3	Instanciar modelo KConfig a modelo de clases	4	Alta
4	Traducir modelo de clases a modelo SPLOT	6	Media
5	Reconocer inconsistencias en SPL	2	Media
6	Determinar características muertas en SPL	1	Media
7	Determinar características opcionales en SPL	1	Media
8	Validar producto ingresado	2	Media
9	Obtener cantidad de productos derivables en SPL	1	Media
10	Validar configuración parcial de un producto	2	Baja
11	Obtener cantidad de productos derivables a partir de una configuración parcial	1	Baja

Historias de usuario desarrolladas.

Tabla 6.B.5.5. Product Backlog actualizado.

Gestión de riesgos

No se detectó la ocurrencia de riesgos en esta iteración de una manera directa. Sí se sufrieron las consecuencias de la activación del riesgo R18 en la iteración anterior y las acciones correctivas aplicadas, lo cual inevitablemente impactó en los tiempos de esta última iteración.

Comentarios y conclusiones de la iteración

Como comentarios sobre la finalización de esta iteración, es importante resaltar el cumplimiento de los deadlines de manera holgada. Esto fue debido a que la funcionalidad de lógica de las historias se obtuvo de una manera sencilla, tal

como en iteración 4. Principalmente, los esfuerzos se enfocaron en el desarrollo de interfaces de usuario atractivas y funcionales.

Con respecto a la historia 9 (Obtener cantidad de productos derivables en SPL), la ejecución de la misma fue más rápida de lo esperado debido a que la interfaz fue desarrollada en la iteración anterior y la funcionalidad de la tarea de lógica fue obtenida a partir de un ejemplo, sólo fue necesario su adaptación.

Aunque la historia 10 (Validar configuración parcial de un producto) fue separada en tareas, fue necesario desarrollar las mismas en paralelo, debido al alto nivel de interacción y dependencia existente entre la lógica y la interfaz. Como punto a favor e importante de mencionar, para elaborar esta historia se reutilizó código anteriormente generado para la historia 8 (Validar producto ingresado), permitiendo acortar los tiempos. Algo similar sucedió con la historia 11 (Obtener cantidad de productos derivables a partir de una configuración parcial), reutilización de código y agilización de los tiempos

Inicialmente, para la ejecución de esta iteración las actividades estaban planificadas para tres semanas. Luego de la re estimación, sólo fueron necesarias dos semanas. La semana restante fue utilizada para realizar pruebas del sistema cruzadas con el objetivo de descubrir errores y posteriormente corregirlos.

Conclusión sobre la planificación en general

Finalmente, el Product Backlog obtenido una vez finalizadas las cinco iteraciones es el mostrado en la Tabla 6.B.4. En el mismo se pueden apreciar los Story Points asignados en la tapa de planificación, antes de comenzar el desarrollo, versus los Story Points estipulados luego de las re planificaciones hechas al comienzo de las iteraciones 3 y 4. También se muestra la iteración asignada versus la realmente ejecutada para cada historia.

Product Backlog

ID historia	Nombre historia	SP planificados	SP re estimados	Iteración planificada	Iteración ejecutada
1	Gestionar (Alta/Baja) modelo KConfig	3	3	2	2
2	Parsear archivo KConfig	2	2	1	1
3	Instanciar modelo KConfig a modelo de clases	4	4	1	1 y 2
4	Traducir modelo de clases a modelo SPLOT	3	6	2	3
5	Reconocer inconsistencias en SPL	3	2	3	4
6	Determinar características muertas en SPL	3	1	3	4
7	Determinar características opcionales en SPL	3	1	4	4
8	Validar producto ingresado	3	2	4	4
9	Obtener cantidad de productos derivables en SPL	3	1	5	5
10	Validar configuración parcial de un producto	1	2	5	5
11	Obtener cantidad de productos derivables a partir de una configuración parcial	2	1	5	5
	Total	30	25		

Tabla 6.B.4. Product Backlog final.

Adicionalmente a la Tabla 6.B.4, se desea agregar que a la suma de Story Points luego de la re estimación no se le computaron los SP necesarios para la finalización de la historia 3, los cuales son 2 / 3 puntos. Teniendo en cuenta estos puntos faltantes y finalización anticipada del desarrollo, una semana antes de lo planificado (lo cual representa 2 SP), se acerca a los 30 SP inicialmente estimados.

Las dificultades atravesadas se dieron básicamente por el desconocimiento del problema a resolver, el desconocimiento del lenguaje Kconfig y la manipulación de fórmulas lógicas programáticamente, las cuales requirieron mayor esfuerzo para su tratamiento. Estas dificultades impactaron de manera directa sobre los tiempos estimados, implicando demoras. Afortunadamente, fueron sólo dos

historias (historias 3 y 4) las afectadas y se pudo absorber su impacto en la planificación a través del alto nivel de adaptabilidad ofrecido por SCRUM.
Favorablemente, como contracara de estas dificultades encontradas se gozó de las ventajas acarreadas por la disposición de una expresión regular inicial descriptiva del lenguaje Kconfig (historia 2) y la utilización de la biblioteca SPLAR (historias 5, 6, 7, 8, 9, 10 y 11), la cual brinda una interfaz sencilla para su utilización, contrariamente a lo esperado por el equipo. Estas ventajas se vieron traducidas en un increíble ahorro de tiempos, permitiendo recuperar las demoras sufridas y terminar la herramienta antes de lo imaginado. De no haberse utilizado SPLAR, la demora hubiera sido de al menos una iteración más.
Otra tecnología que permitió agilizar los tiempos fue JavaFX. A través de la misma se logró una alta reutilización de código, lo cual permitió acotar el esfuerzo requerido para el desarrollo de las interfaces de usuario.

Como contraparte a la utilización de las tecnologías mencionadas en el párrafo anterior, se detectó que el empleo de la herramienta de soporte para Scrum–Kunagi no fue tan beneficioso como se esperaba. La funcionalidad de la misma es acotada y además presenta bugs, sumado a que no está siendo continuamente actualizada/mantenida. La única ventaja de la misma es su gratuidad, aunque existen mejores herramientas gratuitas para tal fin. En fin, el empleo de Kunagi no puede considerarse un acierto.

Con respecto a la Gestión de Riesgos se desea agregar que el equipo contínuamente estuvo enfocado a la prevención de ocurrencia de los mismos. Para evitar la ocurrencia del R7–”Deterioro de la calidad del código debido a la falta de comprensión del problema impactando negativamente en la mantenibilidad”, se decidió darle suficiente tiempo a aquellas historias de mayor complejidad, aceptando la ocurrencia del R18–"Estimaciones de tiempo incorrectas en la etapa de planificación”. Un riesgo catalogado con una alta probabilidad de ocurrencia y esperado por el equipo fue el R23–”Disminución de la calidad del producto generada por la utilización de una tecnología

desconocida". Al final del desarrollo se puede concluir que el mismo no se dió y, contrariamente a lo esperado se dió una situación inversa, ya que la utilización de JavaFX y SPLAR permitió un desempeño limpio y sencillo.

Por último, se concluye que aunque las estimaciones de las historias no fueron del todo acertadas, se pudo concluir dentro del tiempo estipulado para la ejecución de esta herramienta, considerándose un balance positivo por el equipo de desarrollo.

7. Pruebas del sistema y demostración de funcionalidades

En esta sección se demostrará al lector las capacidades de la herramienta, mediante una explicación detallada de sus funcionalidades y salidas de las mismas.

Con el objetivo de facilitar al lector la comprensión de los datos se utiliza un modelo Kconfig pequeño de entrada, el cual corresponde a una familia de productos de una bicicleta. Posteriormente, se utiliza un modelo de variabilidad de más de 600 líneas correspondiente a la descripción de una porción del kernel de Linux, con el objetivo de probar la capacidad de la herramienta.

Al iniciar la herramienta, la pantalla de presentación es la mostrada en la Figura 7.1. La misma presenta en el campo izquierdo los Modelos de Variabilidad disponibles para analizar. Al seleccionar un Modelo de Variabilidad, se despliega su contenido en el campo blanco ubicado a la derecha de la pantalla. A su vez, se habilitan los botones “Eliminar” o “Analizar” para su utilización. En la Figura 7.2 se muestra la pantalla obtenida luego de seleccionar un Modelo de Variabilidad.

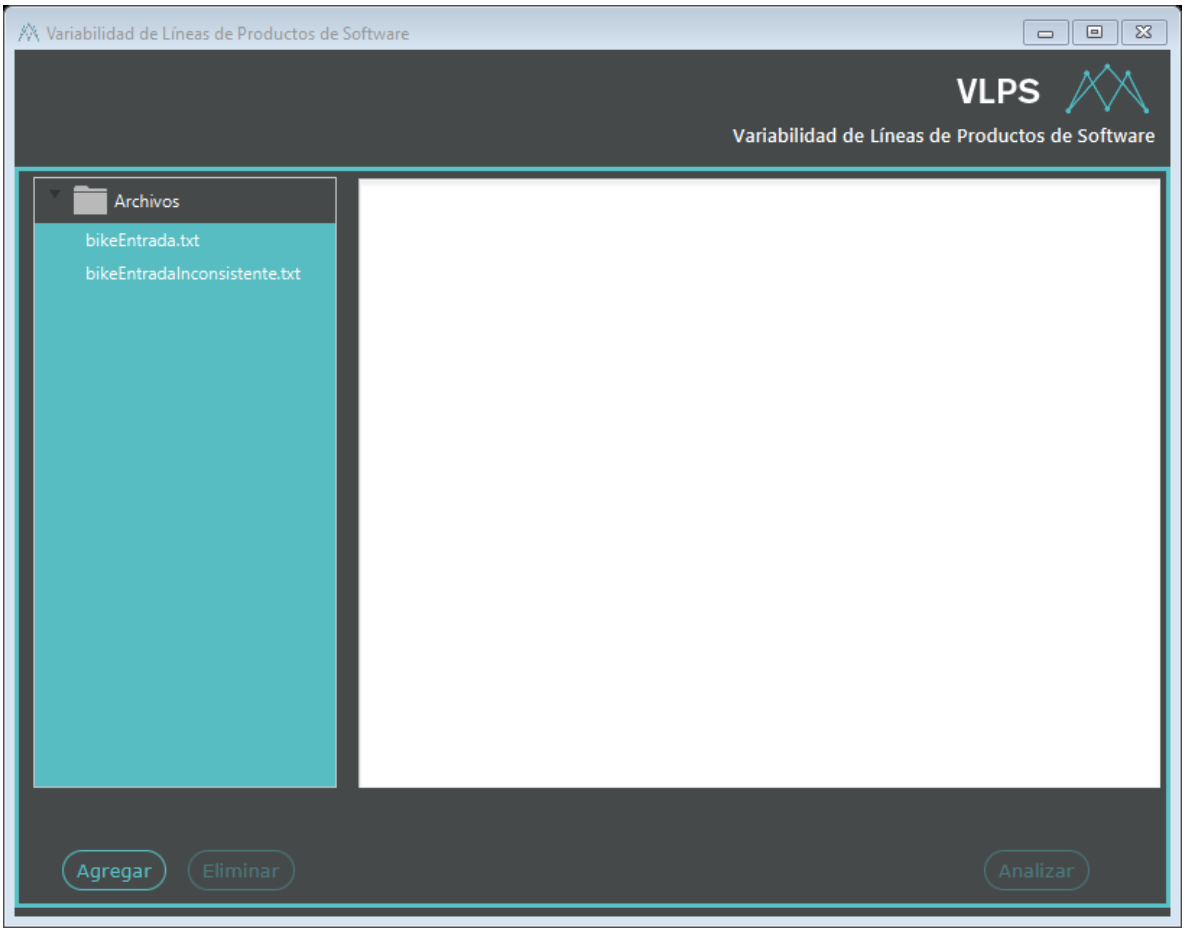

Figura 7.1. Pantalla de inicio de la herramienta.

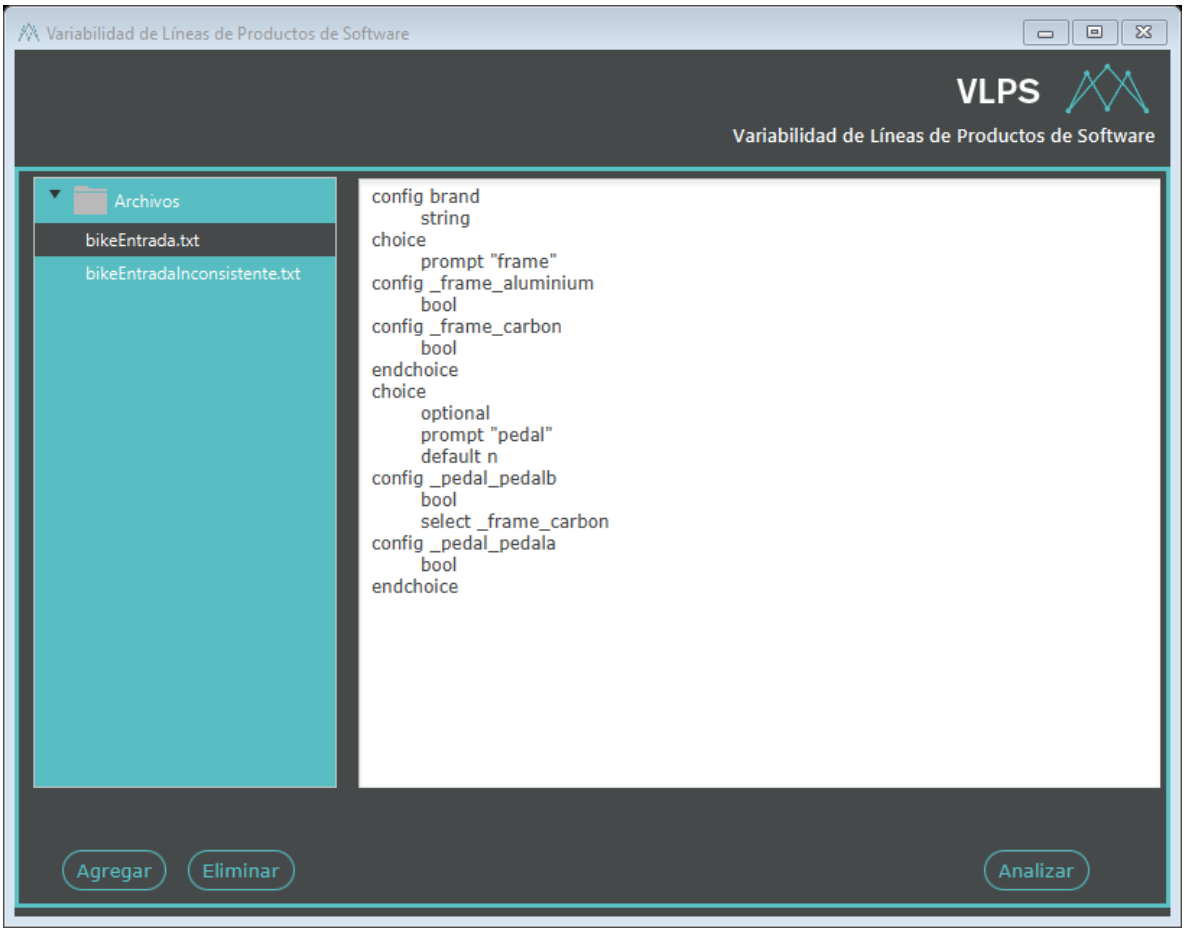

Figura 7.2. Selección de un Modelo de Variabilidad.

En caso de desearse la eliminación de un modelo, se presenta una alerta de confirmación para la eliminación, tal como se visualiza en la Figura 7.3.

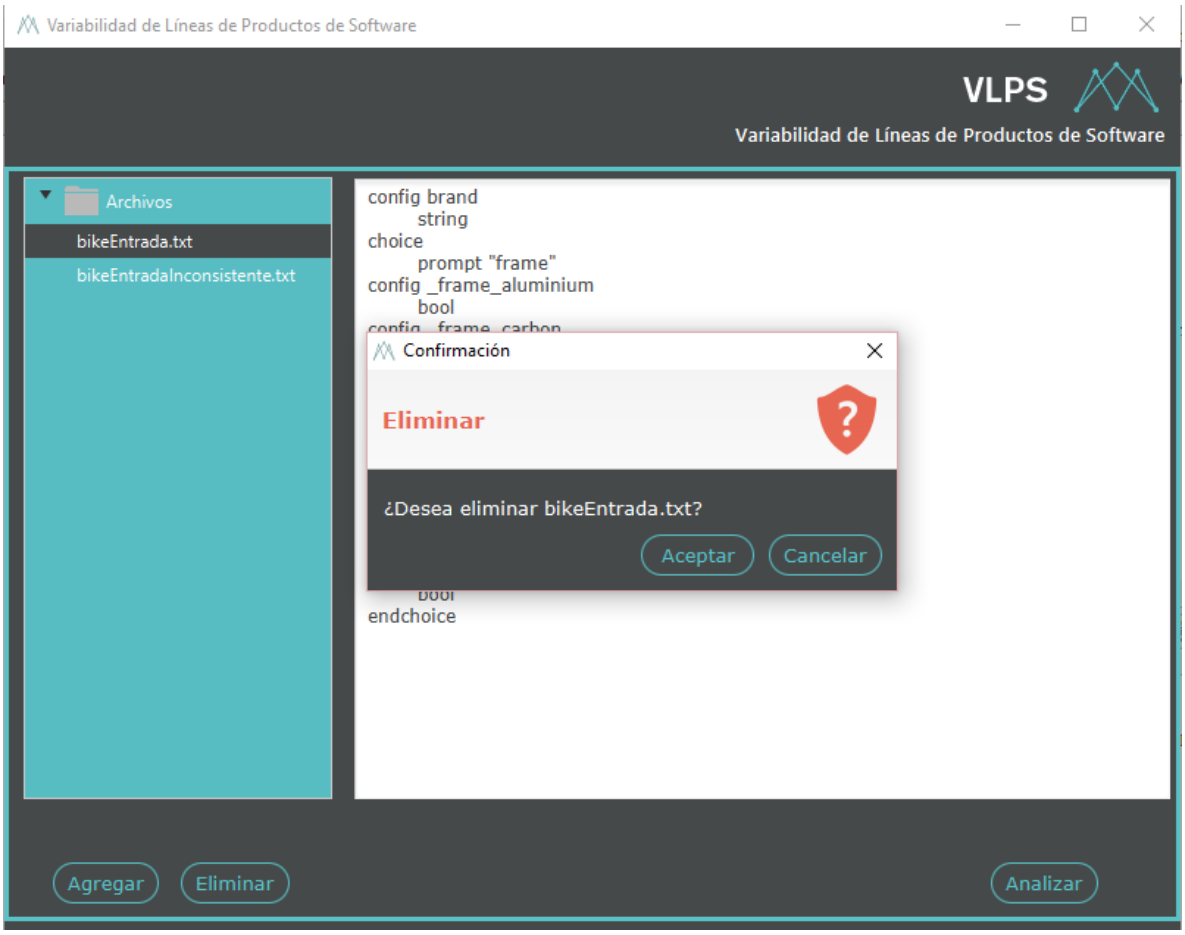

Figura 7.3. Eliminación de un Modelo de Variabilidad.

Para agregar un Modelo de Variabilidad, se debe presionar el botón “Agregar”. La acción de este botón abre en otra ventana el navegador del sistema de archivos. Esto puede ser apreciado en la Figura 7.4.

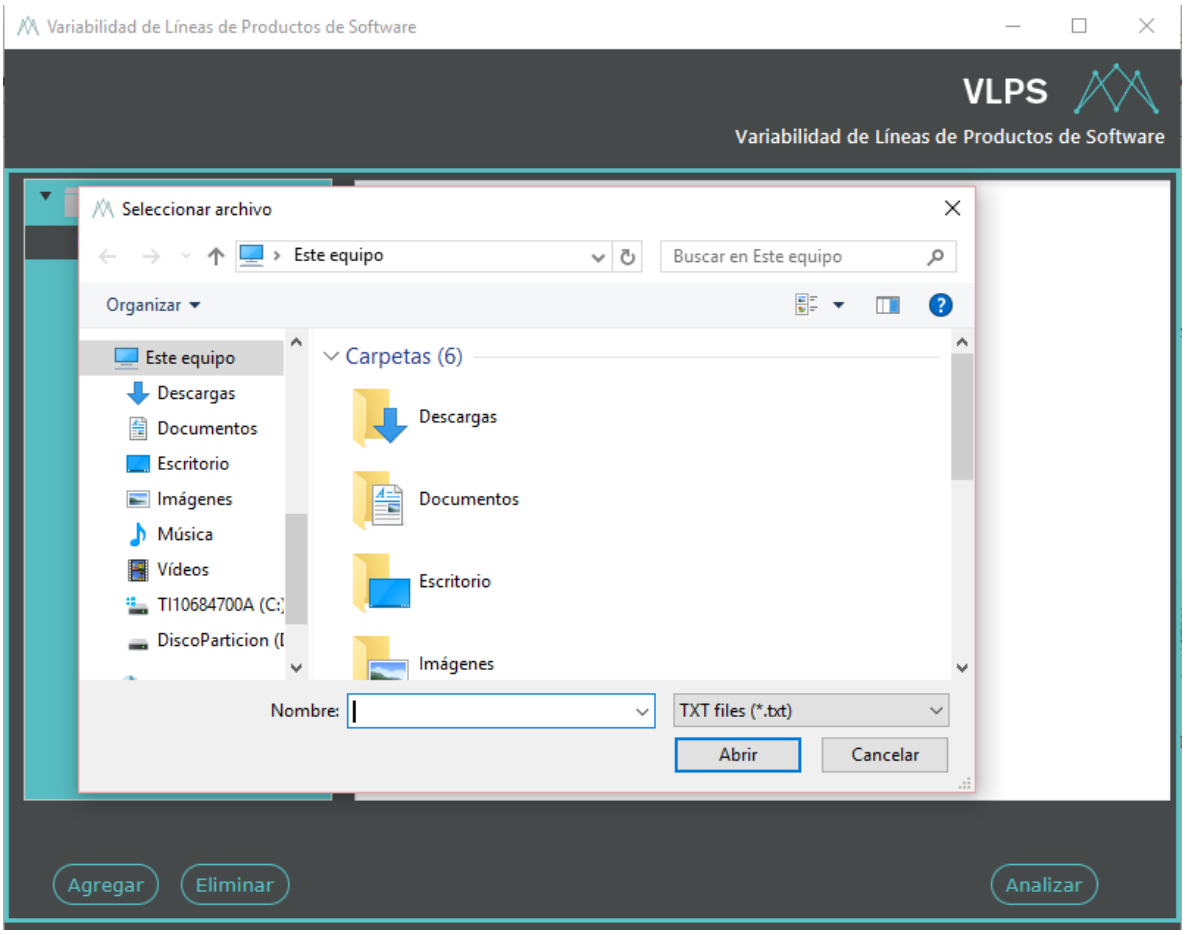

Figura 7.4. Adición de un Modelo de Variabilidad.

Retomando el Modelo de la Figura 7.2, al presionar el botón “Analizar” se transforma el modelo en Kconfig a un modelo de características para permitir su análisis mediante las facilidades aplicables a este tipo de modelos. Luego, al presionar el botón “Correr Análisis”, se realiza una serie de análisis sobre el modelo transformado, obteniéndose el resultado mostrado en la Figura 7.5. Dentro de los datos interesantes de analizar para una Línea de Productos de Software son las características comunes, las características opcionales y la cantidad de configuraciones. En la Figura 7.5, se muestra que para esa línea de productos se obtienen 3 características comunes, 5 opcionales y 5 configuraciones de productos posibles. Las características comunes son Menu Padre (_r), brand (_r_6), frame (_r_7). Particularmente, la característica Menu Padre (_r) es una característica ficticia utilizada para agrupar los componentes

Kconfig los cuales no tienen un padre, es decir, están en la raíz del archivo. Esta última característica estará presente en todos los modelos. Las características opcionales son _frame_aluminum (_r_7_8), _frame_carbon (_r_7_9), pedal (_r_10), _pedal_pedalb (_r_10_11) y _pedal_pedala (_r_10_12). Respecto de las configuraciones posibles, una de ellas es: Menu Padre (_r), brand (_r_6), frame (_r_7), _frame_carbon (_r_7_9), pedal (_r_10) y _pedal_pedalb (_r_10_11).

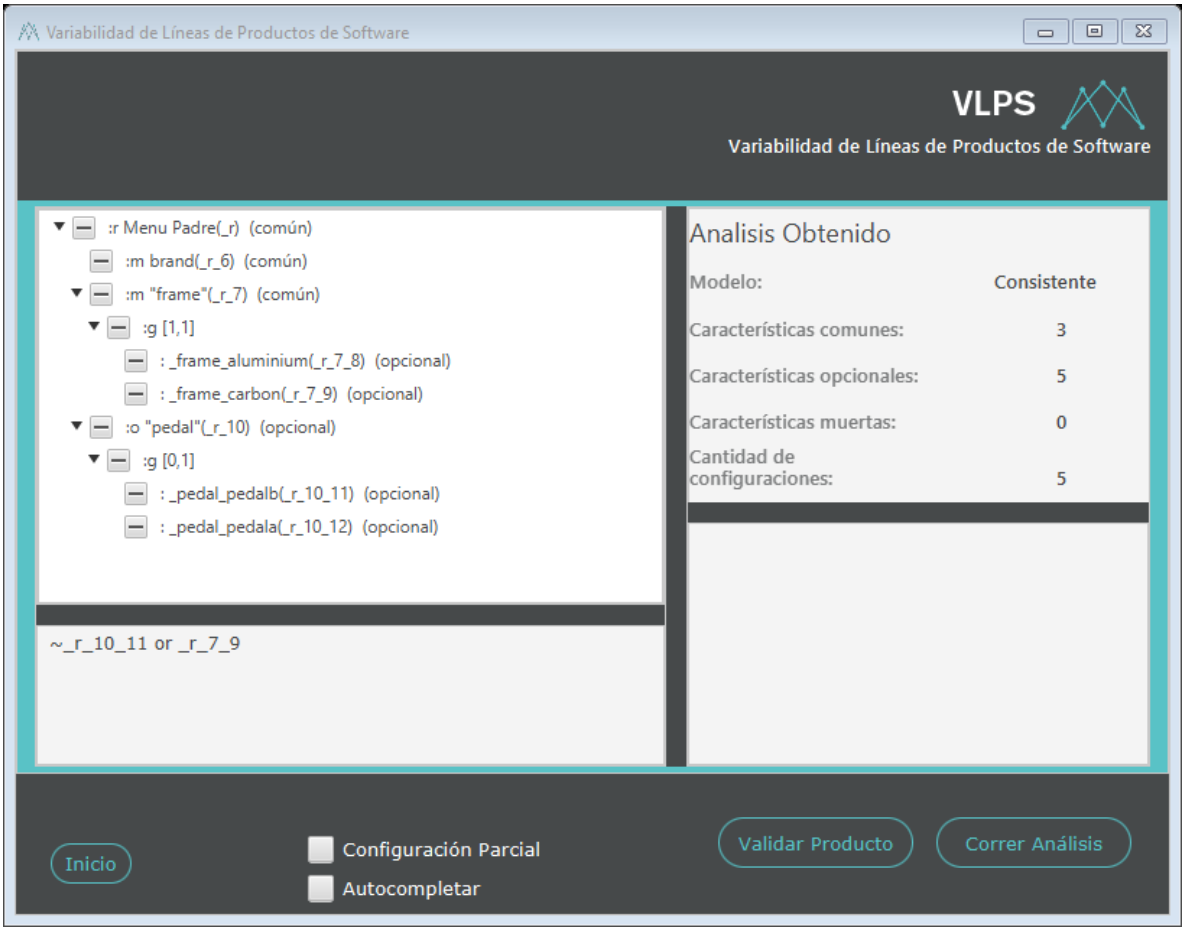

Figura 7.5. Características comunes, opcionales y productos derivables.

A continuación, en la Figura 7.6 se muestra el mismo Modelo de la Figura 7.2 pero con una modificación que hace a la “config _pedal_pedalb” convertirse en una característica muerta. Al modelo de la Figura 7.2 se le adiciona la relación “select _pedal_pedala” a la “config_frame_carbon”, provocando que la config “_pedal_pedalb” no pueda aparecer en algún producto ya que de ser seleccionada, implica la selección de “config_frame_carbon” y por transitividad, la selección de “_pedal_pedala”, violando así la restricción de grupo [0,1].

Además de indicar y contar la cantidad de características muertas, las mismas se deshabilitan para su selección. En la Figura 7.7 se muestra la salida obtenida del Análisis del Modelo de la Figura 7.6. Además se aprecia cómo se reduce la cantidad de configuraciones posibles debido a la restricción adicional.

```
config brand
        string
choice
        prompt "frame"
config _frame_aluminium
        bool
config _frame_carbon
        bool
        select _pedal_pedala
endchoice
choice
        optional
        prompt "pedal"
        default n
config _pedal_pedalb
        bool
        select _frame_carbon
config _pedal_pedala
        bool
endchoice
```

Figura 7.6. Modelo de Variabilidad con característica muerta.

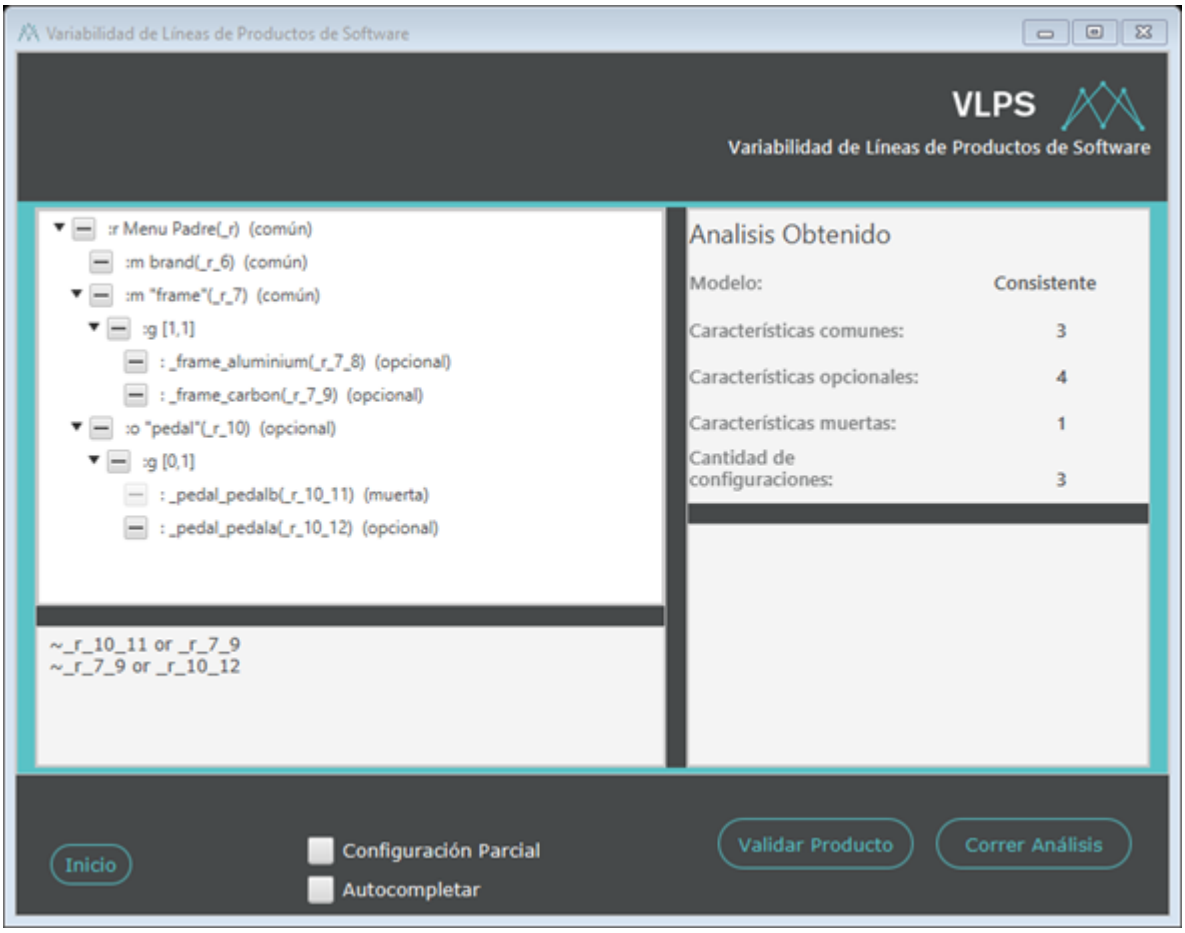

Figura 7.7. Detección de características muertas.

Ahora, al modelo de la Figura 7.2, se le adiciona la cláusula “select _frame_carbon” a la config “_frame_aluminium” y la cláusula “select _frame_aluminium” a la config “_frame_carbon”. Al presionar el botón “Correr Análisis”, se obtiene que el Modelo en cuestión es inconsistente, como se puede observar en la Figura 7.8. Esto se debe a que la selección de la configuración del cuadro de aluminio implica la selección de la configuración del cuadro de carbono y viceversa, violando la restricción del grupo, la cual permite la selección de una y sólo una configuración del tipo “frame” (cuadro). En la Figura 7.9 se muestra el Modelo de Variabilidad analizado, con la adición de las cláusulas anteriormente mencionadas.

Figura 7.8. Detección de Modelo inconsistente.

```
choice
        prompt "frame"
config _frame_aluminium
        bool
        select _frame_carbon
config _frame_carbon
        bool
        select _frame_aluminium
endchoice
choice
        optional
        prompt "pedal"
        default n
config _pedal_pedalb
        bool
        select _frame_carbon
config _pedal_pedala
        bool
endchoice
```

Figura 7.9. Modelo de Variabilidad inconsistente.

En la Figura 7.10 se muestra la funcionalidad correspondiente a la validación de un producto ingresado. Se le permite al usuario seleccionar las características que desea incluir para la configuración de un producto mediante la selección de los checkboxs asociados a cada característica. Estos checkboxs cuentan con tres estados: [✓], [ ] y [–]. El primer símbolo significa que se desea incluir la característica dentro de la configuración del producto, el segundo estado (vacío) significa que se desea excluir la característica de la configuración del producto y el último significa que la inclusión de la característica dentro del producto es indistinta para el usuario. Luego de esto, se debe presionar el botón “Validar Producto”. Para poder acceder a esta funcionalidad, primero se debe haber realizado el análisis sobre el modelo dado, es decir, haber presionado el botón “Correr Análisis” y adicionalmente, el modelo de características debe haber sido consistente. Para la Figura 7.10, se ha decidido incluir sólo una característica, brand (_r_6), la cual es común. Luego de validar el producto, la herramienta ha detectado que la características frame (_r_7) y _frame_aluminium (_r_7_8) o frame_carbon (_r_7_9) deben ser incluidas de manera obligatoria. La herramienta también detecta cuando hay características seleccionadas que no pueden ser incluidas. Además, se deshabilitan los checkboxs de las características muertas, las cuales no pueden ser parte de ninguna configuración de productos.

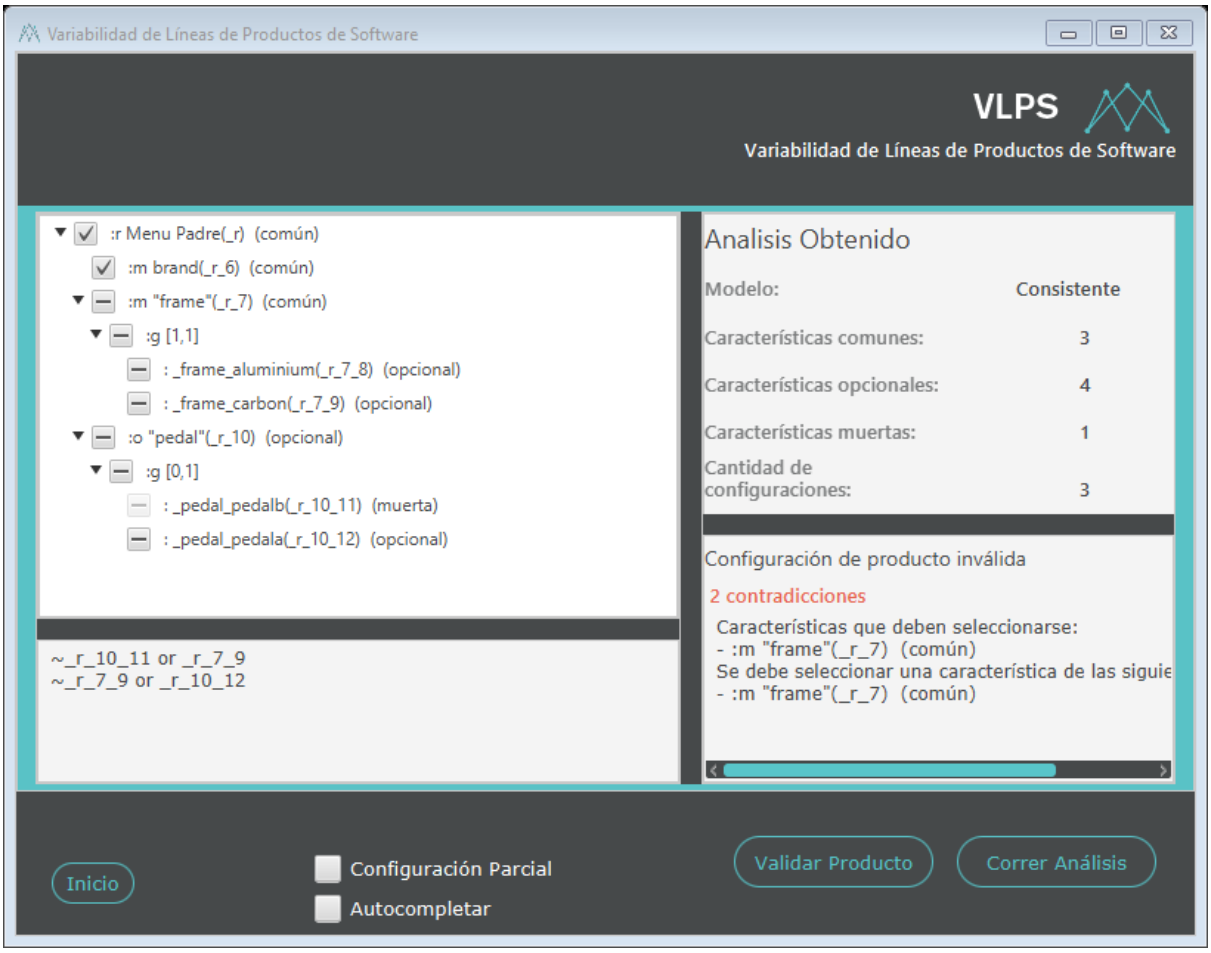

Figura 7.10. Validación de una configuración de un producto.

Se desea destacar que, ante la selección de una característica dentro del árbol, en caso de la misma poseer configuraciones antecesoras, las mismas también son seleccionadas para ahorrar tiempo al usuario (ya que deberán ser incluidas para obtener un producto válido). Durante el desarrollo de la herramienta, se consideró muy útil incluir la funcionalidad de “Autocompletar”, la cual funciona a modo de guía/ayuda para el usuario, seleccionando o deseleccionando las configuraciones según lo requiera el modelo cuando la misma está activada. Esta interacción permite al usuario la validación de un producto a medida que sus características son incluidas. Cuando la función Autocompletar se encuentra actividad, se indica en rojo en la parte superior de la pantalla.

En caso del usuario desear la validación un producto parcialmente ingresado, deberá seleccionar el checkbox “Configuración Parcial”, tal como se muestra en

la Figura 7.11 y luego de haber marcado la configuración parcial deseada, deberá presionar el botón “Validar Producto”. Notar que cuando el checkbox de Configuración Parcial está seleccionado, se indica en color rojo en la parte superior izquierda de la pantalla. Además, en la Figura 7.11 también se podrá apreciar la funcionalidad la cual entrega la cantidad de configuraciones de productos posibles dada una configuración parcial ingresada. Para obtener esta funcionalidad, se debe seleccionar “Configuración Parcial” y luego “Correr Análisis” nuevamente.

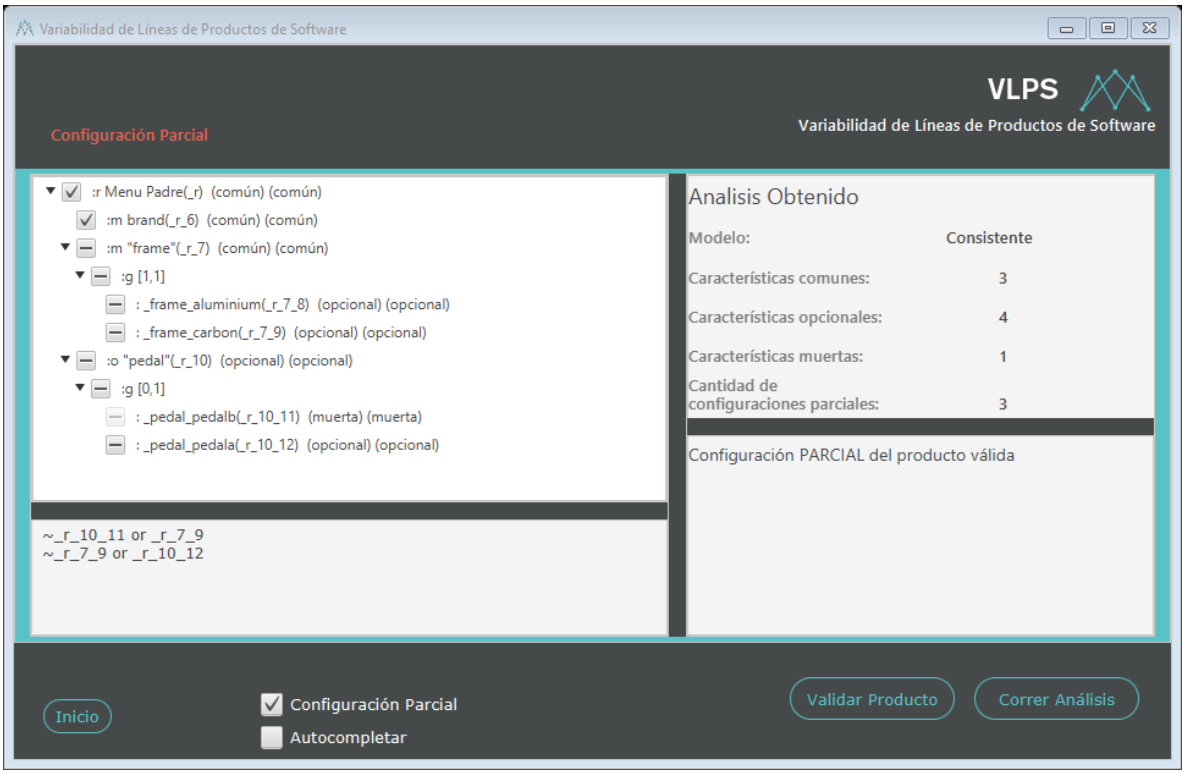

Figura 7.11. Validación y Análisis de configuración parcial.

Tal como se anunció al principio de esta sección, en la Figura 7.12 se puede observar el modelo de características obtenido a partir de un Modelo de Variabilidad con 1453 líneas de código Kconfig. A simple vista se puede apreciar que dicho modelo es consistente, posee 42 características comunes, 383 opcionales y 4 muertas y se pueden derivar mediante el mismo más de $2.147.483.647 \cong 2.15E^9$ productos. Los modelos utilizados pueden ser

visualizados en la sección C del Anexo “Modelos utilizados en pruebas del sistema” de este informe.

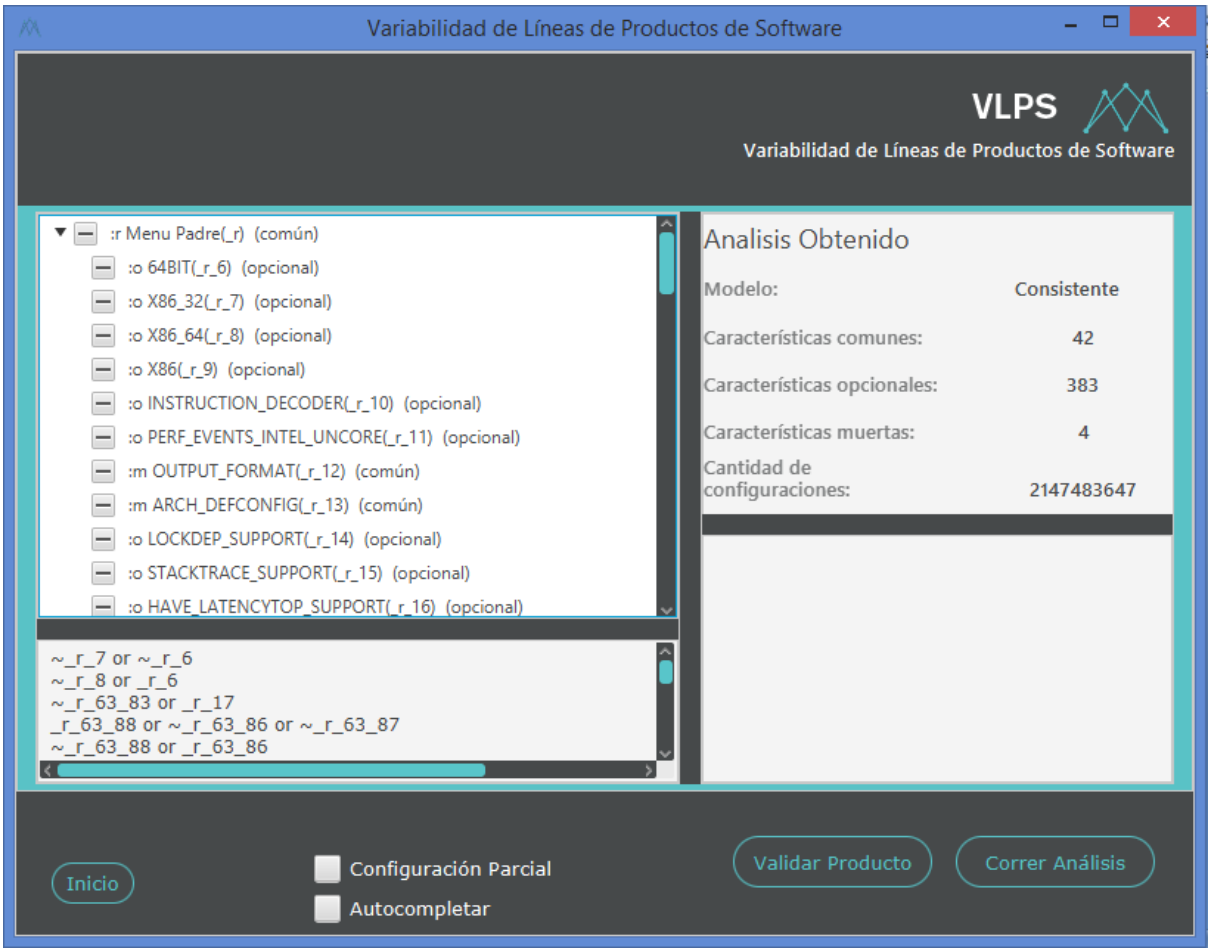

Figura 12. Salida obtenida para modelo de mayor escala.

8. Conclusiones

Conclusión sobre la herramienta

Luego de finalizado el desarrollo de la herramienta se realizó un análisis sobre la misma, teniendo en cuenta la funcionalidad ofrecida, el diseño de las interfaces, y la usabilidad. Debajo se presenta el balance obtenido, el cual incluye las fortalezas y debilidades detectadas.

Dentro de los aspectos a destacar, se encuentran:

- Interfaz de usuario intuitiva y agradable a la vista,
- Alta validación de botones y configs (cuando una config no puede seleccionarse porque hace inconsistente un modelo o porque es muerta, se deshabilita),
- Permite modelos de escala: se ha probado el análisis de modelos Kconfig de aproximadamente 3000 líneas aunque con una demora considerable según la complejidad del modelo.

Dentro de las limitaciones o aspectos los cuales hubiese sido interesante incluir:

- Falta de una sección de ayuda que brinde una explicación sobre la funcionalidad y el significado de los datos obtenidos en el análisis.
- Funcionalidad de búsqueda de palabras dentro de los modelos de variabilidad, expresados en Kconfig.
- Captación de mayor interacción entre las configs, tal como se explica en la sección “Trabajos futuros”.
- Información al usuario de errores encontrados en la gramática Kconfig para archivos de entrada. La herramienta, por medio de ANTLR actualmente detecta errores en la gramática, aunque estos no son comunicados al usuario.

Conclusión sobre el proyecto

En líneas generales, el desarrollo de la herramienta estuvo acompañado de manera contínua por gran incertidumbre, debido a la novedad y complejidad del tema en cuestión. El esfuerzo requerido para completar ciertas funcionalidades no siempre fue lineal, es decir, no se podía estimar con seguridad la complejidad asociada, como sí sucede al momento de estimar la realización de un ABM. Por suerte y, si bien no se utilizó SCRUM puro, esta metodología permitió al equipo adaptarse fácilmente a los inconvenientes surgidos en la planificación.

Además, es para destacar la experiencia lograda en el ámbito de investigación, como así también el conocimiento adquirido relacionado al análisis de líneas de productos de software, modelos de variabilidad, y modelos de características. Otro punto relevante es el aprendizaje de dos tecnologías consideradas muy útiles: ANTLR4 y JavaFX. Las mismas permitieron un ágil desenvolvimiento a través del proyecto.

Para ir finalizando, se desea comentar que los modelos de características obtenidos a partir de ésta herramienta pueden ser utilizados como entradas para la herramienta FM2PN [17], la cual fue desarrollada bajo el mismo área de estudio, permitiendo así el análisis de variabilidad mediante una metodología alternativa a la propuesta aquí.

A modo de cierre, se desea expresar la satisfacción del equipo con respecto al producto obtenido, a pesar de las debilidades y oportunidades detectadas en la sección anterior. Se espera que el fruto de este proyecto sea utilizado por la comunidad científica y aquellas empresas donde se mantengan líneas de productos de software.

Impacto del proyecto

La industria de software ha comenzado a centrar el desarrollo de sus productos siguiendo un enfoque de líneas de productos de software. En este tipo de

desarrollo la industria del software requiere de herramientas que permitan especificar la variabilidad de sus productos. En la actualidad, existe una marcada tendencia al empleo del lenguaje Kconfig para expresar tal variabilidad. Asimismo, el alto nivel de variabilidad y las interdependencia entre características posibles en los distintos productos requieren de herramientas que brinden soporte en el análisis de la variabilidad, identificando posibles inconsistencias. En la literatura existen distintas propuestas para realizar tales análisis, una buena discusión se presenta en el trabajo de Benavides y colab. [9]. Sin embargo, estas alternativas están basadas en modelos de características, no incluyendo a modelos expresados en Kconfig [7], por eso se requiere de herramientas que permitan el análisis de la variabilidad. Actualmente, existen proyectos formulados para el desarrollo de herramientas que aborden esta problemática, por ejemplo, en https://kernelnewbies.org/KernelProjects/kconfig-sat se propone una herramienta basada en un SAT-solver. Incluso, en la última actualización de la definición de Kconfig se formula la necesidad de tales herramientas [11]. La herramienta desarrollada en este proyecto es un primer paso para satisfacer esta necesidad. La herramienta permite analizar modelos Kconfig mediante su traducción a modelos de características.

La herramienta presentada es de gran utilidad para el soporte a proyectos de investigación en la temática de líneas de producto. La arquitectura de “pipe and filter” propuesta en este trabajo para el desarrollo de la herramienta permite emplear y/o explorar otro tipo de análisis a los modelos de variabilidad, facilitando así actividades futuras de investigación. Por ejemplo, es posible reemplazar el Analizador (Figura 5.1) por otro componente que realice los análisis sin emplear un SAT-solver. En la sección previa se indica esta posibilidad empleando la herramienta FM2PN [17]. Incluso, se podría tomar la salida del Parser (Figura 5.1) para realizar un análisis directo sobre el modelo de variabilidad en Kconfig, tal como se discute en las siguientes secciones de Discusión y Trabajo Futuro.

Discusión

La problemática encontrada a partir de la traducción de un modelo de variabilidad a un modelo de características, es decir, traducción de un modelo expresado en Kconfig a la notación utilizada por SPLOT, se basa en la pérdida de expresividad del modelo de variabilidad en Kconfig. Esto se debe a la imposibilidad de incluir muchos atributos de las configs, los cuales pueden hacer que las mismas interactúen de manera distinta. Sin embargo, dada la complejidad del lenguaje Kconfig, se permite analizar un gran número de problemas que pueden presentarse a nivel estructural, lo cual es destacable.
Para evitar la pérdida de información, se debería haber realizado el análisis directamente sobre los modelos de variabilidad en Kconfig. Esto representa un tema de estudio aparte ya que supone un mayor esfuerzo al no contar con suficiente bibliografía comparado con el análisis de modelos de características ni bibliotecas de código para facilitar el trabajo.

Trabajos futuros

Como continuación de esta línea de trabajo, se propone el desarrollo de un analizador de modelos de variabilidad, el cual trabajará directamente sobre el código Kconfig, dando solución a la problemática presentada encima. Esta solución haría uso de la arquitectura “pipe & filter” propuesta en este trabajo (Figura 5.1) y permitiría el reuso del primer componente, el Parser.

Otra opción para dar solución y, tal vez, más sencilla para su ejecución sería extender esta herramienta con el objetivo de contemplar interacciones entre las configs. Esto no fue incluido en el alcance del proyecto debido a un mayor requerimiento de tiempo del dispuesto por el equipo de desarrollo. Contemplar la interacción entre las configs sería tener en cuenta el valor de las mismas (por ejemplo, una config “bool” con valor “y”) y el impacto de este valor en el resto de las configs dependientes.

9. Referencias

[1] P. Clements, L. Northrop, Software Product Lines – Practice and Patterns, Addison-Wesley, 2001.
[2] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering – Foundations, Principles, and Techniques, Springer, 2005.
[3] S. Apel, D. Batory, C. Kästner, G. Saake, Feature-Oriented Software Product Lines, Springer-Verlag, 2013.
[4] K. Kang, S. Cohen, J. Hess, W. Nowak, S. Peterson, Feature-oriented domain analysis (FODA) feasibility study, Tech.Rep.CMU/SEI-90-TR-21, CMU-SEI, 1990.
[5] K. Kang, J. Lee, P. Donohoe, "Featured-Oriented Product Line Engineering", IEEE Software, 19(4), 2002, pp. 58-65.
[6] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, A. Wasowski, "A survey of variability modeling in industrial practice". VaMoS, ACM, 2013.
[7] T. Berger, S. She, R. Lotufo, A. Wasowski, K. Czarnecki, "A study of variability models and languages in the systems software domain", IEEE Transaction on Software Engineering, 39 (12), 2013, pp 1611-1640.
[8] I. Abal, C. Brabrand, A. Wasowski , "42 Variability Bugs in the Linux Kernel: A Qualitative Analysis", ASE'14, 2014, pp. 421-432.
[9] D. Benavides, S. Segura, A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review", Journal of Information Systems, 35, 2010, pp. 615-636.
[10] M. Mendonca, M. Branco, D. Cowan, "S.P.L.O.T.: Software Product Lines Online Tools", Proc. 24th ACM SIGPLAN Conf. OOPSLA, 2009.
[11] R. Zippel y otros autores, "kconfig-language.txt," Disponible en https://github.com/torvalds/linux/blob/master/Documentation/kbuild/kconfig-language.txt, accedido 2016.
[12] S. She, R. Lotufo, T. Berger, A. Wasowski, K. Czarnecki, "The Variability Model of The Linux Kernel". VaMoS, ACM, 2010.
[13] J. Sincero, H. Schirmeier, W. Schröder-Preikschat, O. Spinczyk, "Is The Linux Kernel a Software Product Line?". SPLC-OSSPL 2007, 2007.
[14] Clements P., Northrop L.: A framework for software product line practice, Version 5.0, 2007. Disponible en http://www.sei.cmu.edu/productlines/frame_report/index.html, accedido 2016.
[15] Svahnberg, M., van Gurp, J., and Bosch, J., "On the Notion of Variability in Software Product Lines", In Proc. of the Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society Press, 2001. pp: 45-54.
[16] Parnas, D.L. "On the Design and Development of Program Families," IEEE Trans. on Software Eng. 2(1), March 1976, pp: 1-9
[17] Duttweiler, J. "F2MPN (Feature Models 2 Petri Net)". Proyecto final de Carrera, Universidad Tecnológica Nacional, Facultad Regional Santa Fe, 2016

Anexos

A. Bocetos

Tal como fue presentado en la etapa de Investigación, en esta sección se adjuntan las maquetas de interfaces de usuario, las cuales fueron realizadas con el objetivo de realizar una puesta en común de ideas dentro del equipo de desarrollo.

En la Figura A.1 se presenta la maqueta correspondiente a la pantalla de inicio de la herramienta.

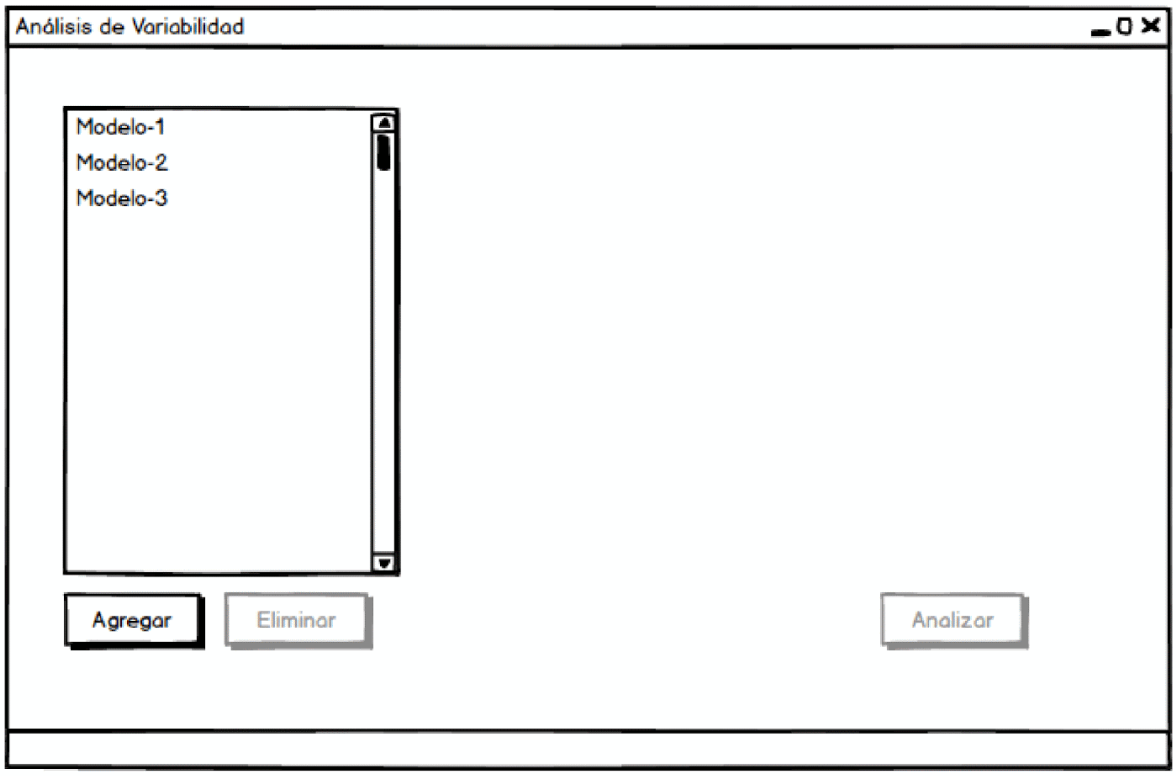

Figura A.1. Pantalla de inicio de la herramienta (Maqueta Inicio)

En la Figura A.2 se presenta la maqueta correspondiente a la pantalla luego de que se presiona sobre el botón “Agregar” (Modelo de Variabilidad), abriéndose el navegador del Sistema de Archivos del sistema operativo.

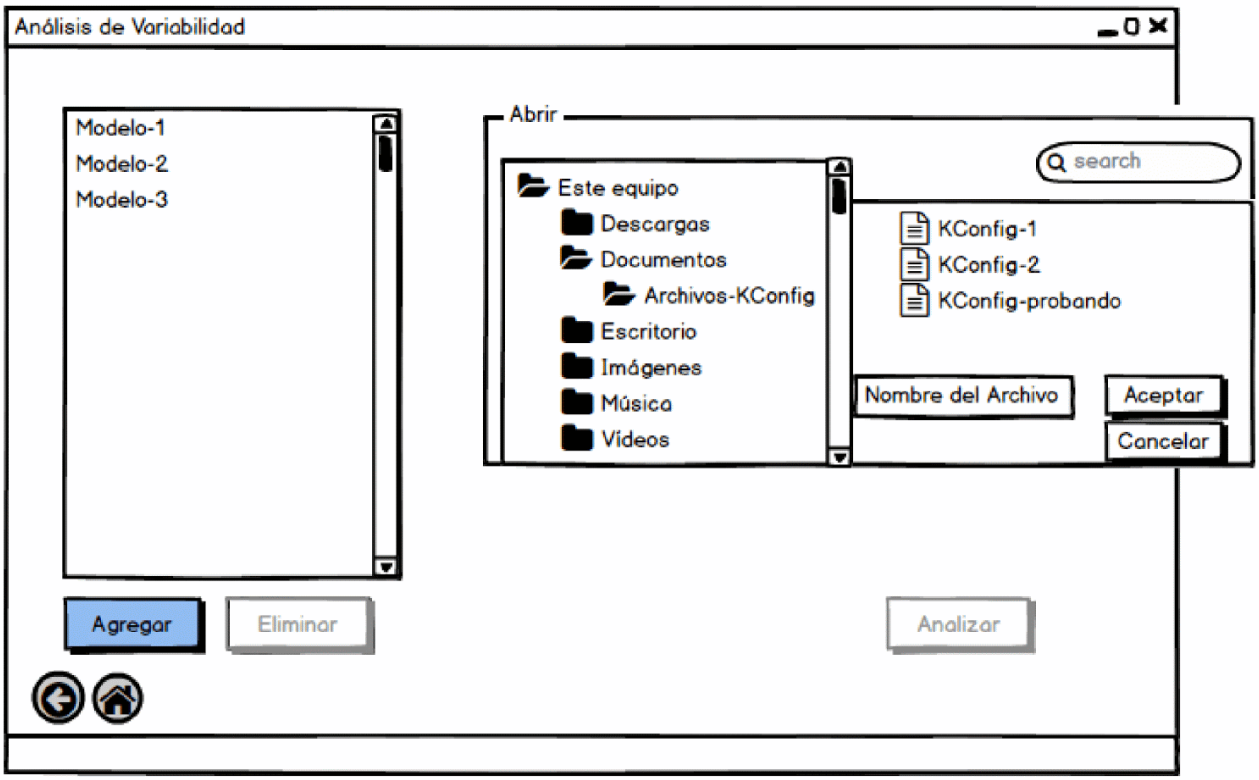

Figura A.2. Pantalla búsqueda Modelo Kconfig (Maqueta “Botón Agregar”).

En la Figura A.3 se muestra la maqueta correspondiente a la pantalla presentada luego de que el usuario selecciona un Modelo Kconfig para su visualización.

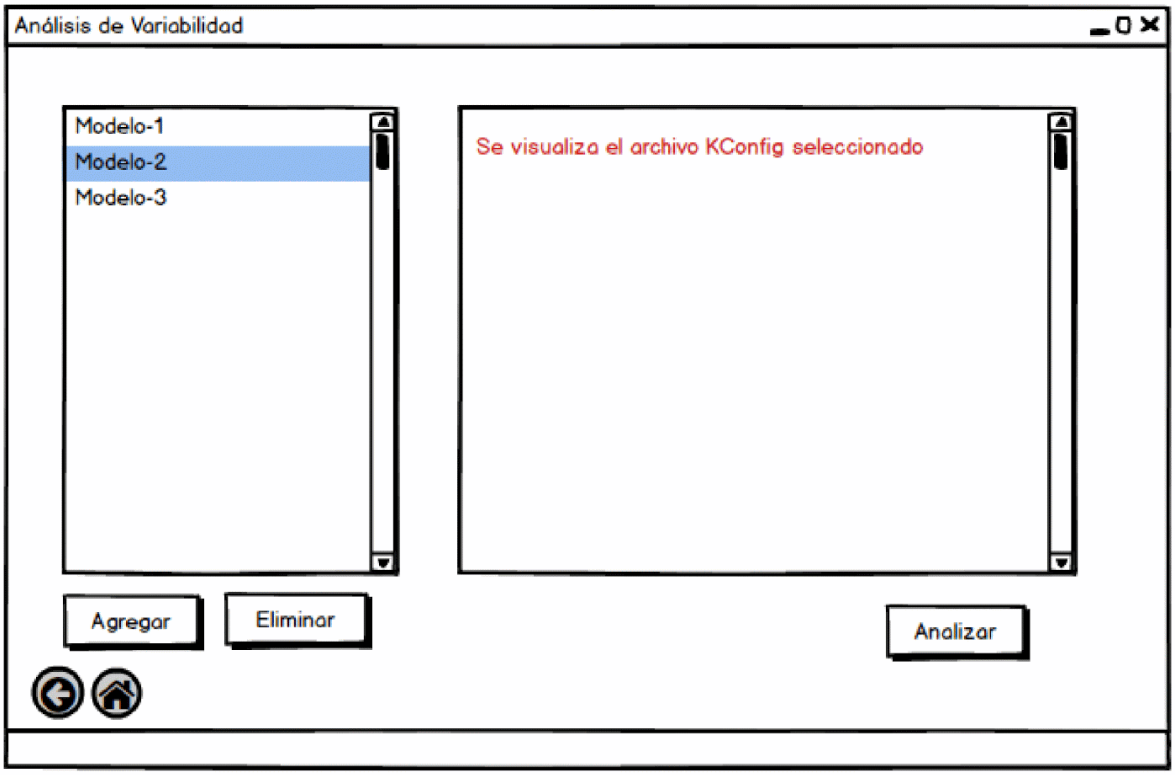

Figura A.3. Pantalla visualización Modelo Kconfig (Maqueta Selección Modelo-Kconfig)

En la Figura A.4 se muestra la maqueta correspondiente a la pantalla presentada luego de que el usuario presiona el botón “Eliminar”, presentándose un cuadro de diálogo para confirmar la eliminación.

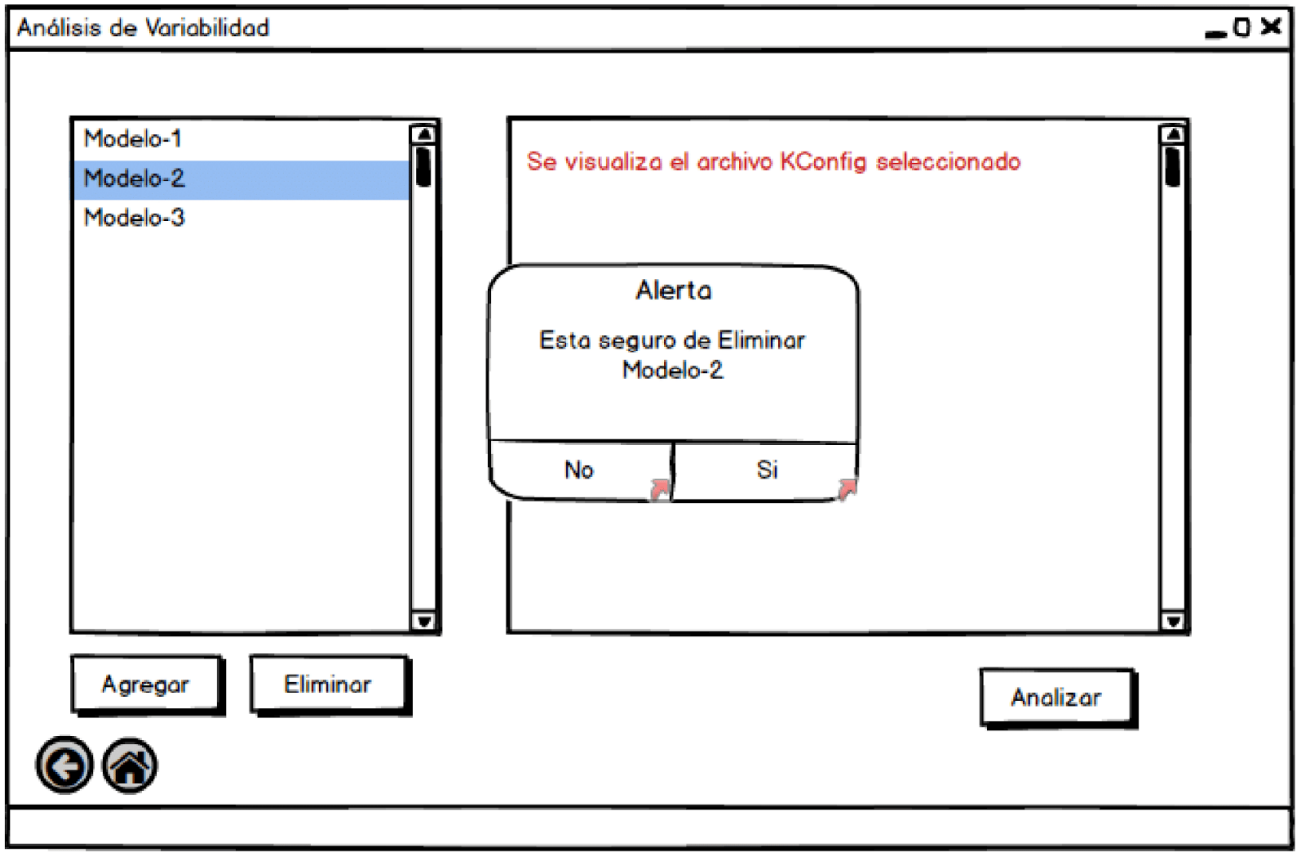

Figura A.4. Pantalla confirmación eliminación modelo Kconfig (Maqueta Eliminar)

En la Figura A.5 se muestra la maqueta correspondiente a la pantalla presentada luego de que el usuario presiona el botón “Analizar”. Se puede reconocer el área donde se mostrará el modelo de características y el área donde se imprimirá el análisis del modelo correspondiente.

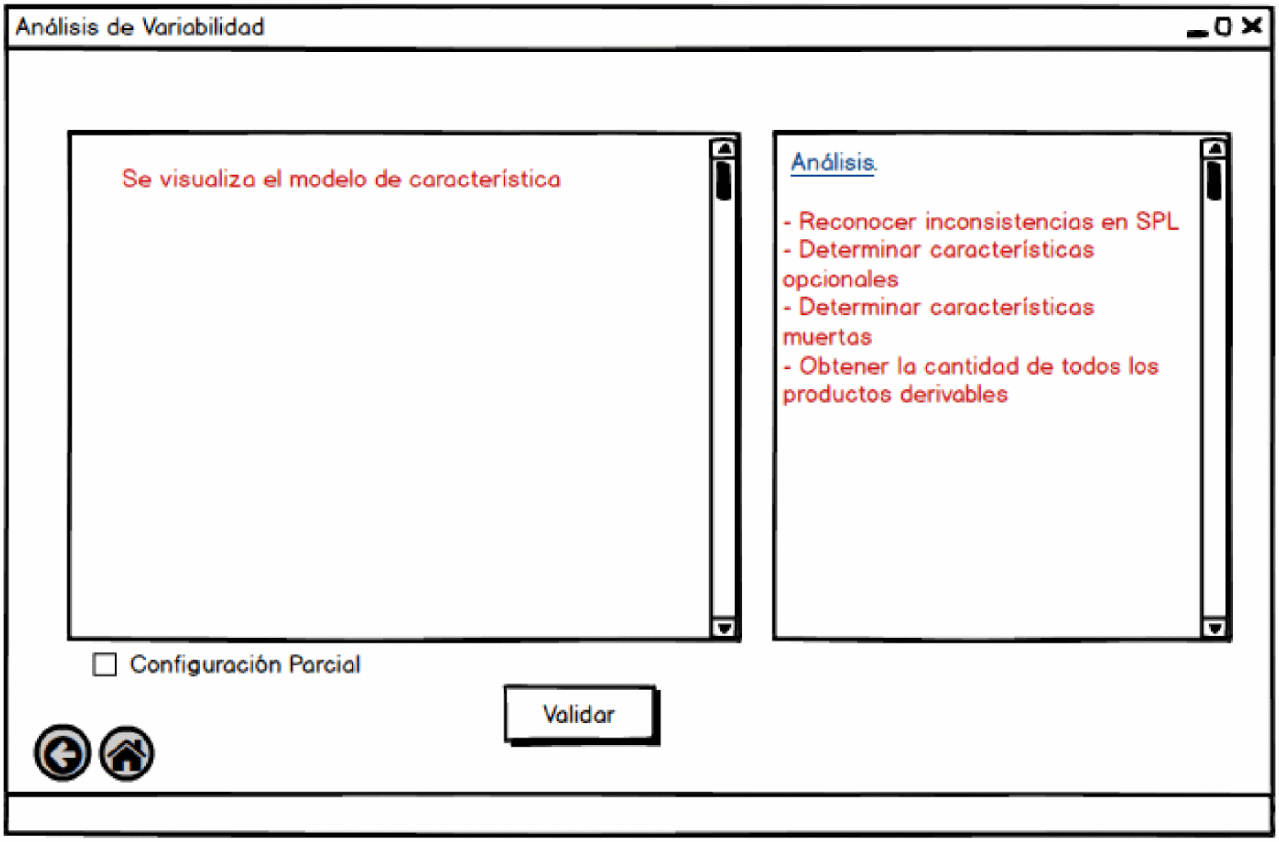

Figura A.5. Pantalla correspondiente al analizador (maqueta “Botón Analizar”).

Adicionalmente, en esta sección también debería incluirse el modelo conceptual de clases Kconfig obtenido en la etapa de Investigación, el cual fue utilizado como guía al momento de desarrollar la historia 3 “Instanciar modelo KConfig a modelo de clases”. Este modelo se muestra en la sección 5, donde se explica el funcionamiento de la herramienta.

B. Historias de usuario

Dentro de esta sección se listan las historias de usuarios, en sus versiones preliminares, al momento de realizar el plan de este proyecto y las versiones refinadas de las mismas, hechas al comienzo de cada iteración. Además, las historias se presentarán con la iteración planificada vs la iteración real y con sus estimaciones preliminares, re estimaciones de haberse realizado y Story Points (SP) realmente utilizados.

Número: 1	**Nombre: Gestionar modelo Kconfig**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 2	**Iteración ejecutada: 2**
Story Points estimados: 3	**Story Points utilizados: 3**
Historia preliminar: Se deberá poder mantener un conjunto de modelos/archivos KConfig dentro de la herramienta. Para esto, se deberá poder ingresar archivos KConfig mediante el File System (es decir, proveyendo el path del archivo), realizar una copia de este archivo y mantenerla en la base de datos de la herramienta. También se deberá permitir la eliminación de modelos KConfig almacenados en la herramienta. Además de la funcionalidad mencionada anteriormente se deberá listar todos los modelos KConfig cargados; ésta funcionalidad será mostrada en la pantalla principal de la herramienta. Observaciones: La modificación dentro de la herramienta de los archivos KConfig no será soportada.	**Historia refinada:** Se deberá poder mantener un conjunto de modelos/archivos KConfig dentro de la herramienta. Para esto, se deberá poder ingresar archivos KConfig mediante el File System (es decir, proveyendo el path del archivo), realizar una copia de este archivo y mantenerla en la base de datos de la herramienta. Es suficiente con manejar los archivos KConfig dentro de una carpeta dentro del sistema. También se deberá permitir la eliminación de modelos KConfig almacenados en la herramienta. Para esto, se deberá borrar el archivo almacenado en la carpeta designada y actualizar el listado de modelos disponibles dentro del sistema. Además de la funcionalidad mencionada anteriormente se deberá listar todos los modelos KConfig cargados; ésta funcionalidad será mostrada en la pantalla principal de la herramienta. El usuario podrá seleccionar un modelo

	KConfig mediante un click y la herramienta deberá mostrar su contenido. Para la realización de esta historia de usuario se dispondrá de una maqueta (maqueta "Botón Agregar"). Observaciones: La modificación dentro de la herramienta de los archivos KConfig no será soportada.

Número: 2	**Nombre: Parsear archivo KConfig**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 1	**Iteración ejecutada: 1**
Story Points estimados: 2	**Story Points utilizados: 2**
Historia preliminar: Se deberá poder reconocer sintácticamente los componentes de un archivo KConfig identificados en la etapa de Investigación. Para esto, se deberá utilizar una herramienta diseñada para tal fin (en lugar de desarrollarla). Observaciones: La herramienta de análisis sintáctico será una de las salidas esperadas del bloque de investigación.	**Historia refinada:** Se deberá poder reconocer sintácticamente los componentes de un archivo KConfig identificados en la etapa de Investigación. Para esto se propone la utilización de la herramienta ANTLR4. Esta herramienta requiere de un archivo con extensión .g4, el cual describirá la sintaxis del lenguaje KConfig mediante expresiones regulares. Luego de generado el archivo mencionado anteriormente, se deberá ejecutar ANTLR4. Como resultado se obtendrá un conjunto de clases y métodos en java, los cuales deberán ser modificados para adaptarse a las necesidades del equipo. La expresiones regulares deberán tener en cuenta la mayor cantidad de atributos posibles: símbolo (nombre), prompt (descripción), tipo (boolean, tristate,

	integer o string), default, select y depends on. También deberán ser tenidas en cuenta las formas de anidamiento o agrupaciones entre diferentes opciones de características (configs), estos agrupadores son: "menú", "menuconfig", "choice" o bloques "if". Los elementos del tipo "menú" o los bloques "if" pueden contener cualquier otro elemento dentro, los "menuconfig" pueden contener una sentencia "if" dentro, los "choice" contienen "configs" o "choice". Observaciones: Detrás del select puede encontrarse una cláusula "if". Es de importancia reconocer el atributo "Optional" dentro de las choice.

Número: 3	**Nombre: Instanciar modelo KConfig a modelo de clases**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 1	**Iteración ejecutada: 1 y 2**
Story Points estimados: 4	**Story Points utilizados: 6**
Historia preliminar: A partir del reconocimiento sintáctico desarrollado en la historia de usuario 2, se deberá traducir el modelo KConfig a un modelo de clases con el mayor nivel de detalle posible. Cuando se refiere a un alto nivel de detalle, se quiere decir a capturar la funcionalidad e información de cada elemento que compone el lenguaje. Observaciones:	**Historia refinada:** En base a la información provista en la historia de usuario 2 y las clases .java obtenidas del análisis sintáctico, se deberá traducir el modelo KConfig a un modelo de clases con el mayor nivel de detalle posible. Observaciones: Se encuentra disponible el modelo de clases a ser utilizado para instanciar el modelo KConfig (surgido en etapa de investigación). De surgir nuevas

Notar que las historias de usuario 2 y 3 se desarrollan en el mismo Sprint (la historia 3 depende en un alto grado de la historia 2). El modelo de clases será una salida del bloque de investigación, como una metáfora para guiar el desarrollo posterior.	versiones para este modelo al momento de la implementación deberán ser documentadas.

Número: 4	**Nombre: Traducir modelo de clases a modelo SPLOT**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 2	**Iteración ejecutada: 3**
Story Points estimados y re-estimados: 3 y 6	**Story Points utilizados: 6**
Historia preliminar: Se deberá representar un modelo KConfig previamente instanciado a un modelo de características en un lenguaje admitido por la herramienta SPLOT. La traducción variará según las características de cada Config (dependiendo si es una Config opcional o mandatoria) y sus dependencias y relaciones con las demás.	**Historia refinada:** Se deberá representar un modelo KConfig previamente instanciado a un modelo de características en un lenguaje admitido por la herramienta SPLOT. La traducción variará según las características de cada Config (dependiendo si es una Config opcional o mandatoria) y sus dependencias y relaciones con las demás. Este modelo de características obtenido deberá ser guardado en el FileSystem, tal como lo hecho con los archivos KConfig. Con el objetivo de respetar el formato utilizado por SPLOT, se deberá disponer de un archivo idéntico a modo de guía al momento de la codificación de la lógica de esta historia. Se provee de información útil para la traducción al modelo de características:

	• Como raíz (_r), se deberá crear un menú que contenga los elementos raíz del archivo KConfig. • Aquellas configs que son del tipo “bool” o “tristate” y poseen un “prompt” se deberán traducir como configs opcionales (:o). El resto se tendrán que tomar como obligatorias (:m). • Se deberá considerar que todo lo que está dentro de un menú pasará a ser elemento hijo del mismo (configs, menuconfigs, if, choices, y otros menús). • Las choices deberán ser traducidas según el “type”: si son del tipo “bool”, sólo permitir seleccionar una de las opciones [1..1]; en caso de ser del tipo “tristate”, se deberá permitir seleccionar uno o más componentes con el valor “m” [1..*]. Si la choice posee “OPTIONAL” dentro de sus opciones, significa que las restricciones anteriores permiten que no se seleccionen componentes (que tomen valor n), por lo que deberá quedar [0..1] y [0..*], respectivamente. • Los menuconfigs se deberán traducir de forma similar a las configs, ya que son configs que agrupan otras configs. • Cuando una entrada posee la opción “depends on” y la expresión que le sigue hace referencia a sólo una entrada (un menú, una config, etc.), se deberá traducir como hija de la entrada referenciada en la expresión. Por lo contrario, si la expresión hace referencia a más de una entrada, se deberá considerar como una equivalencia con la entrada referenciada. La herramienta lo

	deberá traducir en fórmulas proposicionales de forma normal conjuntiva para cada dirección de la implicancia. • Cuando una config posee la opción “select” precedida por una expresión, se deberá traducir como una restricción donde la config implica la expresión. Si al final de esta opción se encuentra un “if”, se deberá considerar que la entrada continúa al if implica la restricción declarada. • Los elementos de un bloque if (configs, menuconfigs, menu, choice y otros bloques if), se deberá considerar como hijos de la config o menuconfig mencionada en la entrada “if”.

Número: 5	**Nombre: Reconocer inconsistencias en SPL**
Prioridad de Negocio: (Alta/Media/Baja**)**	
Iteración planificada: 3	**Iteración ejecutada: 4**
Story Points estimados y re-estimados: 3 y 2	**Story Points utilizados: 2**
Historia preliminar: Se deberá detectar inconsistencias existentes en una línea de producto de software especificada en KConfig. La inconsistencia ocurre cuando se dan contradicciones en las reglas y restricciones de dependencias. En un modelo de característica ocurre cuando existe una relación entre características que no pueden ser verdaderas al mismo tiempo. Las	**Historia refinada:** Se deberá detectar inconsistencias existentes en una línea de producto de software especificada en KConfig. La inconsistencia ocurre cuando se dan contradicciones en las reglas y restricciones de dependencias. En un modelo de característica ocurre cuando existe una relación entre características que no pueden ser verdaderas al mismo tiempo. Las inconsistencias se pueden dar también de manera transitiva, por

inconsistencias se pueden dar también de manera transitiva.	ejemplo: una configuración A podría requerir B y B requerir C, se puede dar una inconsistencia transitiva si C excluye A. Observaciones: Es deseable la utilización de la biblioteca SPLAR. Notar que en src/test/java/splar/samples/SATReasoningExample se utiliza el método isConsistent().

Número: 6	**Nombre: Determinar características muertas en SPL**
Prioridad de Negocio: (Alta/Media/Baja**)**	
Iteración planificada: 3	**Iteración ejecutada: 4**
Story Points estimados y re-estimados: 3 y 1	**Story Points utilizados: 1**
Historia preliminar: Se deberá detectar características muertas en una línea de producto de software especificada en KConfig. Las características muertas son las Configs que no aparecen en ningún producto correcto de una SPL.	**Historia refinada:** Se deberá detectar características muertas en una línea de producto de software especificada en KConfig. Las características muertas son las Configs que no aparecen en ningún producto correcto de una SPL debido a un conjunto de restricciones. Observaciones: Para esta historia, sería recomendable utilizar la misma metodología que en src/test/java/splar/samples/SATReasoningExample:80 de la biblioteca SPLAR para la detección de características muertas.

Número: 7	**Nombre: Determinar características opcionales**

Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 4	**Iteración ejecutada: 4**
Story Points estimados y re-estimados: 3 y 1	**Story Points utilizados: 1**
Historia preliminar: Se deberá detectar características opcionales en una línea de producto de software especificada en KConfig. Estas características derivan un producto distinto por cada variante.Una opción define una alternativa para el producto base que puede ser seleccionada como parte de un producto final o no.	**Historia refinada:** Se deberá detectar características opcionales en una línea de producto de software especificada en KConfig. Estas características derivan un producto distinto por cada variante.Una opción define una alternativa para el producto base que puede ser seleccionada como parte de un producto final o no. Observaciones: en src/test/java/splar/samples/ SATReasoningExample de la biblioteca SPLAR, cuando se imprime los resultados, aquellas características opcional son impresas junto al texto [false true], lo cual indica que son opcionales (y pueden tomar ambos valores).

Número: 8	**Nombre: Validar producto ingresado**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 4	**Iteración ejecutada: 4**
Story Points estimados y re-estimados: 3 y 2	**Story Points utilizados: 2**
Historia preliminar: Se deberá poder determinar si un conjunto de configuraciones seleccionadas por el usuario de una SPL definen un producto válido. Se	**Historia refinada:** Se deberá poder determinar si un conjunto de configuraciones seleccionadas por el usuario de una SPL definen un producto válido. Se

considerará válido un producto cuando no contenga inconsistencias ni configuraciones faltantes (que sean mandatorias).	considerará válido un producto cuando no contenga inconsistencias ni configuraciones mandatorias faltantes. Se deberá enfocar en que el usuario ingrese un producto entero, es decir, que no haya configuraciones mandatorias faltantes. Observaciones: en src/test/java/splar/samples/ SATConfigurationExample de la biblioteca SPLAR, se dispone de un método para seleccionar las características, configure(), y un método para chequear las características impactadas por dichas selecciones. También se dispone de un método toggleDecision(). También, se podría utilizar el método autoComplete() para averiguar si a esa configuración dada le resta activar características para que sea un modelo válido.

Número: 9	**Nombre: Obtener cantidad de los productos derivables en SPL**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 5	**Iteración ejecutada: 5**
Story Points estimados y re-estimados: 3 y 1	**Story Points utilizados: 1**
Historia preliminar: Se deberá contar la cantidad de productos derivables de una SPL. Se deberá tener en cuenta las configuraciones opcionales (las cuales derivan un producto distinto por cada variante) y restricciones que componen una SPL, como así también	**Historia refinada:** Se deberá contar la cantidad de productos derivables de una SPL. Se deberá tener en cuenta las configuraciones opcionales (las cuales derivan un producto distinto por cada variante) y restricciones que componen

las inconsistencias que se puedan aparecer.	una SPL, como así también las inconsistencias que se puedan aparecer. Observaciones: para esto, puede ser útil utilizar src/test/java/splar/samples/BDDReasoningExample de la biblioteca SPLAR. El método nodeCount() entrega la cantidad de productos configurables.

Número: 10	**Nombre: Validar configuración parcial de un producto**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 5	**Iteración ejecutada: 5**
Story Points estimados y re-estimados: 1 y 2	**Story Points utilizados: 2**
Historia preliminar: Se permitirá al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo KConfig que describa una SPL. Dada dicha selección y el modelo KConfig, se deberá decidir si las configuraciones seleccionadas son compatibles, es decir, si pueden convivir en un producto válido.	**Historia refinada:** Se permitirá al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo KConfig que describa una SPL. Dada dicha selección y el modelo KConfig, se deberá decidir si las configuraciones seleccionadas son compatibles, es decir, si pueden convivir en un producto válido. Observaciones: se deberán analizar los métodos configure() y toggleDecision() de src/test/java/splar/samples/SATConfigurationExample de la biblioteca SPLAR. Podría utilizarse el método getPropagations() y, las features muertas podrían ser desactivadas para evitar que el usuario las pueda seleccionar al momento de seleccionar otras configuraciones.

Número: 11	**Nombre: Obtener cantidad de productos derivables a partir de una configuración parcial**
Prioridad de Negocio: (Alta/Media/Baja)	
Iteración planificada: 5	**Iteración ejecutada: 5**
Story Points estimados y re-estimados: 2 y 1	**Story Points utilizados: 1**
Historia preliminar: Se permitirá al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo KConfig que describa una SPL. Dada dicha selección y el modelo KConfig, se deberá entregar la cantidad de productos pertenecientes a la SPL que admitan las configuraciones seleccionadas.	**Historia refinada:** Se permitirá al usuario seleccionar un subconjunto de configuraciones pertenecientes a un modelo KConfig que describa una SPL. Dada dicha selección y el modelo KConfig, se deberá entregar la cantidad de productos pertenecientes a la SPL que admitan las configuraciones seleccionadas. Observaciones: cada vez que se selecciona una opción, debería ser agregada como constraint (sin or) en el feature model mediante el método addConstraint() de src/test/java/splar/samples/BDDReasoningExample de la biblioteca SPLAR. Luego, se podría utilizar el método nodeCount() para obtener la cantidad de productos derivables.

C. Modelos utilizados en pruebas del sistema

En esta sección se presenta un ejemplo utilizado como prueba de sistema para demostrar el comportamiento del sistema para modelos de variabilidad a mayor escala. Primero se mostrará el modelo de variabilidad correspondiente a una porción de código Kconfig y posteriormente el modelo de características obtenido por la herramienta. Adicionalmente, los modelos podrán ser descargados y/o visualizados desde la web, a través de los siguientes links:

- Modelo de variabilidad: https://goo.gl/Jq5ZWI.
- Modelo de características: https://goo.gl/O8cBLX.

Modelo de Variabilidad

```
config 64BIT
      bool "64-bit kernel" if ARCH = "x86"
      default ARCH != "i386"
config X86_32
      def_bool y
      depends on !64BIT
      select CLKSRC_I8253
      select HAVE_UID16
config X86_64
      def_bool y
      depends on 64BIT
      select X86_DEV_DMA_OPS
      select ARCH_USE_CMPXCHG_LOCKREF
      select HAVE_LIVEPATCH
config X86
      def_bool y
      select ACPI_SYSTEM_POWER_STATES_SUPPORT if ACPI
      select ARCH_MIGHT_HAVE_ACPI_PDC if ACPI
      select ARCH_HAS_DEBUG_STRICT_USER_COPY_CHECKS
      select ARCH_HAS_FAST_MULTIPLIER
      select ARCH_HAS_GCOV_PROFILE_ALL
      select ARCH_MIGHT_HAVE_PC_PARPORT
      select ARCH_MIGHT_HAVE_PC_SERIO
      select HAVE_AOUT if X86_32
      select HAVE_UNSTABLE_SCHED_CLOCK
      select ARCH_SUPPORTS_NUMA_BALANCING if X86_64
      select ARCH_SUPPORTS_INT128 if X86_64
      select HAVE_IDE
      select HAVE_OPROFILE
      select HAVE_PCSPKR_PLATFORM
      select HAVE_PERF_EVENTS
      select HAVE_IOREMAP_PROT
      select HAVE_KPROBES
      select HAVE_MEMBLOCK
```

```
select HAVE_MEMBLOCK_NODE_MAP
select ARCH_DISCARD_MEMBLOCK
select ARCH_WANT_OPTIONAL_GPIOLIB
select ARCH_WANT_FRAME_POINTERS
select HAVE_DMA_ATTRS
select HAVE_DMA_CONTIGUOUS
select HAVE_KRETPROBES
select GENERIC_EARLY_IOREMAP
select HAVE_OPTPROBES
select HAVE_KPROBES_ON_FTRACE
select HAVE_FTRACE_MCOUNT_RECORD
select HAVE_FENTRY if X86_64
select HAVE_C_RECORDMCOUNT
select HAVE_DYNAMIC_FTRACE
select HAVE_DYNAMIC_FTRACE_WITH_REGS
select HAVE_FUNCTION_TRACER
select HAVE_FUNCTION_GRAPH_TRACER
select HAVE_FUNCTION_GRAPH_FP_TEST
select HAVE_SYSCALL_TRACEPOINTS
select SYSCTL_EXCEPTION_TRACE
select HAVE_KVM
select HAVE_ARCH_KGDB
select HAVE_ARCH_TRACEHOOK
select HAVE_GENERIC_DMA_COHERENT if X86_32
select HAVE_EFFICIENT_UNALIGNED_ACCESS
select USER_STACKTRACE_SUPPORT
select HAVE_REGS_AND_STACK_ACCESS_API
select HAVE_DMA_API_DEBUG
select HAVE_KERNEL_GZIP
select HAVE_KERNEL_BZIP2
select HAVE_KERNEL_LZMA
select HAVE_KERNEL_XZ
select HAVE_KERNEL_LZO
select HAVE_KERNEL_LZ4
select HAVE_HW_BREAKPOINT
select HAVE_MIXED_BREAKPOINTS_REGS
select PERF_EVENTS
select HAVE_PERF_EVENTS_NMI
select HAVE_PERF_REGS
select HAVE_PERF_USER_STACK_DUMP
select HAVE_DEBUG_KMEMLEAK
select ANON_INODES
select HAVE_ALIGNED_STRUCT_PAGE if SLUB
select HAVE_CMPXCHG_LOCAL
select HAVE_CMPXCHG_DOUBLE
select HAVE_ARCH_KMEMCHECK
select HAVE_ARCH_KASAN if X86_64 && SPARSEMEM_VMEMMAP
select HAVE_USER_RETURN_NOTIFIER
select ARCH_HAS_ELF_RANDOMIZE
select HAVE_ARCH_JUMP_LABEL
select ARCH_HAS_ATOMIC64_DEC_IF_POSITIVE
select SPARSE_IRQ
select GENERIC_FIND_FIRST_BIT
```

```
	select GENERIC_IRQ_PROBE
	select GENERIC_PENDING_IRQ if SMP
	select GENERIC_IRQ_SHOW
	select GENERIC_CLOCKEVENTS_MIN_ADJUST
	select IRQ_FORCED_THREADING
	select HAVE_BPF_JIT if X86_64
	select HAVE_ARCH_TRANSPARENT_HUGEPAGE
	select HAVE_ARCH_HUGE_VMAP if X86_64 || (X86_32 && X86_PAE)
	select ARCH_HAS_SG_CHAIN
	select CLKEVT_I8253
	select ARCH_HAVE_NMI_SAFE_CMPXCHG
	select GENERIC_IOMAP
	select DCACHE_WORD_ACCESS
	select GENERIC_SMP_IDLE_THREAD
	select ARCH_WANT_IPC_PARSE_VERSION if X86_32
	select HAVE_ARCH_SECCOMP_FILTER
	select BUILDTIME_EXTABLE_SORT
	select GENERIC_CMOS_UPDATE
	select HAVE_ARCH_SOFT_DIRTY if X86_64
	select CLOCKSOURCE_WATCHDOG
	select GENERIC_CLOCKEVENTS
	select ARCH_CLOCKSOURCE_DATA
	select CLOCKSOURCE_VALIDATE_LAST_CYCLE
	select GENERIC_CLOCKEVENTS_BROADCAST if X86_64 || (X86_32 &&
X86_LOCAL_APIC)
	select GENERIC_TIME_VSYSCALL
	select GENERIC_STRNCPY_FROM_USER
	select GENERIC_STRNLEN_USER
	select HAVE_CONTEXT_TRACKING if X86_64
	select HAVE_IRQ_TIME_ACCOUNTING
	select VIRT_TO_BUS
	select MODULES_USE_ELF_REL if X86_32
	select MODULES_USE_ELF_RELA if X86_64
	select CLONE_BACKWARDS if X86_32
	select ARCH_USE_BUILTIN_BSWAP
	select ARCH_USE_QUEUE_RWLOCK
	select OLD_SIGSUSPEND3 if X86_32 || IA32_EMULATION
	select OLD_SIGACTION if X86_32
	select COMPAT_OLD_SIGACTION if IA32_EMULATION
	select RTC_LIB
	select HAVE_DEBUG_STACKOVERFLOW
	select HAVE_IRQ_EXIT_ON_IRQ_STACK if X86_64
	select HAVE_CC_STACKPROTECTOR
	select GENERIC_CPU_AUTOPROBE
	select HAVE_ARCH_AUDITSYSCALL
	select ARCH_SUPPORTS_ATOMIC_RMW
	select HAVE_ACPI_APEI if ACPI
	select HAVE_ACPI_APEI_NMI if ACPI
	select ACPI_LEGACY_TABLES_LOOKUP if ACPI
	select X86_FEATURE_NAMES if PROC_FS
	select SRCU
config INSTRUCTION_DECODER
	def_bool y
```

```
      depends on KPROBES || PERF_EVENTS || UPROBES
config PERF_EVENTS_INTEL_UNCORE
      def_bool y
      depends on PERF_EVENTS && CPU_SUP_INTEL && PCI
config OUTPUT_FORMAT
      string
      default "elf32-i386" if X86_32
      default "elf64-x86-64" if X86_64
config ARCH_DEFCONFIG
      string
      default "arch/x86/configs/i386_defconfig" if X86_32
      default "arch/x86/configs/x86_64_defconfig" if X86_64
config LOCKDEP_SUPPORT
      def_bool y
config STACKTRACE_SUPPORT
      def_bool y
config HAVE_LATENCYTOP_SUPPORT
      def_bool y
config MMU
      def_bool y
config SBUS
      bool
config NEED_DMA_MAP_STATE
      def_bool y
      depends on X86_64 || INTEL_IOMMU || DMA_API_DEBUG || SWIOTLB
config NEED_SG_DMA_LENGTH
      def_bool y
config GENERIC_ISA_DMA
      def_bool y
      depends on ISA_DMA_API
config GENERIC_BUG
      def_bool y
      depends on BUG
      select GENERIC_BUG_RELATIVE_POINTERS if X86_64
config GENERIC_BUG_RELATIVE_POINTERS
      bool
config GENERIC_HWEIGHT
      def_bool y
config ARCH_MAY_HAVE_PC_FDC
      def_bool y
      depends on ISA_DMA_API
config RWSEM_XCHGADD_ALGORITHM
      def_bool y
config GENERIC_CALIBRATE_DELAY
      def_bool y
config ARCH_HAS_CPU_RELAX
      def_bool y
config ARCH_HAS_CACHE_LINE_SIZE
      def_bool y
config HAVE_SETUP_PER_CPU_AREA
      def_bool y
config NEED_PER_CPU_EMBED_FIRST_CHUNK
      def_bool y
```

```
config NEED_PER_CPU_PAGE_FIRST_CHUNK
      def_bool y
config ARCH_HIBERNATION_POSSIBLE
      def_bool y
config ARCH_SUSPEND_POSSIBLE
      def_bool y
config ARCH_WANT_HUGE_PMD_SHARE
      def_bool y
config ARCH_WANT_GENERAL_HUGETLB
      def_bool y
config ZONE_DMA32
      def_bool y if X86_64
config AUDIT_ARCH
      def_bool y if X86_64
config ARCH_SUPPORTS_OPTIMIZED_INLINING
      def_bool y
config ARCH_SUPPORTS_DEBUG_PAGEALLOC
      def_bool y
config HAVE_INTEL_TXT
      def_bool y
      depends on INTEL_IOMMU && ACPI
config X86_32_SMP
      def_bool y
      depends on X86_32 && SMP
config X86_64_SMP
      def_bool y
      depends on X86_64 && SMP
config X86_HT
      def_bool y
      depends on SMP
config X86_32_LAZY_GS
      def_bool y
      depends on X86_32 && !CC_STACKPROTECTOR
config ARCH_HWEIGHT_CFLAGS
      string
      default "-fcall-saved-ecx -fcall-saved-edx" if X86_32
      default "-fcall-saved-rdi -fcall-saved-rsi -fcall-saved-rdx
-fcall-saved-rcx -fcall-saved-r8 -fcall-saved-r9 -fcall-saved-r10
-fcall-saved-r11" if X86_64
config ARCH_SUPPORTS_UPROBES
      def_bool y
config FIX_EARLYCON_MEM
      def_bool y
config PGTABLE_LEVELS
      int
      default 4 if X86_64
      default 3 if X86_PAE
      default 2
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/init/
Kconfig
config ARCH
      string
```

```
	option env="ARCH"
config KERNELVERSION
	string
	option env="KERNELVERSION"
config DEFCONFIG_LIST
	string
	depends on !UML
	option defconfig_list
	default "/lib/modules/$UNAME_RELEASE/.config"
	default "/etc/kernel-config"
	default "/boot/config-$UNAME_RELEASE"
	default "$ARCH_DEFCONFIG"
	default "arch/$ARCH/defconfig"
config CONSTRUCTORS
	bool
	depends on !UML
config IRQ_WORK
	bool
config BUILDTIME_EXTABLE_SORT
	bool
menu "General setup"
config BROKEN
	bool
config BROKEN_ON_SMP
	bool
	depends on BROKEN || !SMP
	default y
config INIT_ENV_ARG_LIMIT
	int
	default 32 if !UML
	default 128 if UML
config CROSS_COMPILE
	string "Cross-compiler tool prefix"
config COMPILE_TEST
	bool "Compile also drivers which will not load"
	default n
config LOCALVERSION
	string "Local version - append to kernel release"
config LOCALVERSION_AUTO
	bool "Automatically append version information to the version string"
	default y
config HAVE_KERNEL_GZIP
	bool
config HAVE_KERNEL_BZIP2
	bool
config HAVE_KERNEL_LZMA
	bool
config HAVE_KERNEL_XZ
	bool
config HAVE_KERNEL_LZO
	bool
config HAVE_KERNEL_LZ4
	bool
```

```
choice
      prompt "Kernel compression mode"
      default KERNEL_GZIP
      depends on HAVE_KERNEL_GZIP || HAVE_KERNEL_BZIP2 || HAVE_KERNEL_LZMA
|| HAVE_KERNEL_XZ || HAVE_KERNEL_LZO || HAVE_KERNEL_LZ4
config KERNEL_GZIP
      bool "Gzip"
      depends on HAVE_KERNEL_GZIP
config KERNEL_BZIP2
      bool "Bzip2"
      depends on HAVE_KERNEL_BZIP2
config KERNEL_LZMA
      bool "LZMA"
      depends on HAVE_KERNEL_LZMA
config KERNEL_XZ
      bool "XZ"
      depends on HAVE_KERNEL_XZ
config KERNEL_LZO
      bool "LZO"
      depends on HAVE_KERNEL_LZO
config KERNEL_LZ4
      bool "LZ4"
      depends on HAVE_KERNEL_LZ4
endchoice
config DEFAULT_HOSTNAME
      string "Default hostname"
      default "(none)"
config SWAP
      bool "Support for paging of anonymous memory (swap)"
      depends on MMU && BLOCK
      default y
config SYSVIPC
      bool "System V IPC"
config SYSVIPC_SYSCTL
      bool
      depends on SYSVIPC
      depends on SYSCTL
      default y
config POSIX_MQUEUE
      bool "POSIX Message Queues"
      depends on NET
config POSIX_MQUEUE_SYSCTL
      bool
      depends on POSIX_MQUEUE
      depends on SYSCTL
      default y
config CROSS_MEMORY_ATTACH
      bool "Enable process_vm_readv/writev syscalls"
      depends on MMU
      default y
config FHANDLE
      bool "open by fhandle syscalls"
      select EXPORTFS
```

```
config USELIB
      bool "uselib syscall"
      default y
config AUDIT
      bool "Auditing support"
      depends on NET
config HAVE_ARCH_AUDITSYSCALL
      bool
config AUDITSYSCALL
      bool "Enable system-call auditing support"
      depends on AUDIT && HAVE_ARCH_AUDITSYSCALL
      default y if SECURITY_SELINUX
config AUDIT_WATCH
      def_bool y
      depends on AUDITSYSCALL
      select FSNOTIFY
config AUDIT_TREE
      def_bool y
      depends on AUDITSYSCALL
      select FSNOTIFY
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/kerne
l/irq/Kconfig
menu "IRQ subsystem"
config MAY_HAVE_SPARSE_IRQ
       bool
config GENERIC_IRQ_LEGACY
       bool
config GENERIC_IRQ_PROBE
      bool
config GENERIC_IRQ_SHOW
       bool
config GENERIC_IRQ_SHOW_LEVEL
       bool
config GENERIC_IRQ_LEGACY_ALLOC_HWIRQ
       bool
config GENERIC_PENDING_IRQ
      bool
config AUTO_IRQ_AFFINITY
       bool
config HARDIRQS_SW_RESEND
       bool
config IRQ_PREFLOW_FASTEOI
       bool
config IRQ_EDGE_EOI_HANDLER
       bool
config GENERIC_IRQ_CHIP
       bool
       select IRQ_DOMAIN
config IRQ_DOMAIN
      bool
config IRQ_DOMAIN_HIERARCHY
      bool
```

```
      select IRQ_DOMAIN
config GENERIC_MSI_IRQ
      bool
config GENERIC_MSI_IRQ_DOMAIN
      bool
      select IRQ_DOMAIN_HIERARCHY
      select GENERIC_MSI_IRQ
config HANDLE_DOMAIN_IRQ
      bool
config IRQ_DOMAIN_DEBUG
      bool "Expose hardware/virtual IRQ mapping via debugfs"
      depends on IRQ_DOMAIN && DEBUG_FS
config IRQ_FORCED_THREADING
       bool
config SPARSE_IRQ
      bool "Support sparse irq numbering" if MAY_HAVE_SPARSE_IRQ
endmenu
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/kerne
l/time/Kconfig
config CLOCKSOURCE_WATCHDOG
      bool
config ARCH_CLOCKSOURCE_DATA
      bool
config CLOCKSOURCE_VALIDATE_LAST_CYCLE
      bool
config GENERIC_TIME_VSYSCALL
      bool
config GENERIC_TIME_VSYSCALL_OLD
      bool
config ARCH_USES_GETTIMEOFFSET
      bool
config GENERIC_CLOCKEVENTS
      bool
config ARCH_HAS_TICK_BROADCAST
      bool
config GENERIC_CLOCKEVENTS_BROADCAST
      bool
      depends on GENERIC_CLOCKEVENTS
config GENERIC_CLOCKEVENTS_MIN_ADJUST
      bool
config GENERIC_CMOS_UPDATE
      bool
if GENERIC_CLOCKEVENTS
menu "Timers subsystem"
config TICK_ONESHOT
      bool
config NO_HZ_COMMON
      bool
      depends on !ARCH_USES_GETTIMEOFFSET && GENERIC_CLOCKEVENTS
      select TICK_ONESHOT
choice
      prompt "Timer tick handling"
```

```
      default NO_HZ_IDLE if NO_HZ
config HZ_PERIODIC
      bool "Periodic timer ticks (constant rate, no dynticks)"
config NO_HZ_IDLE
      bool "Idle dynticks system (tickless idle)"
      depends on !ARCH_USES_GETTIMEOFFSET && GENERIC_CLOCKEVENTS
      select NO_HZ_COMMON
config NO_HZ_FULL
      bool "Full dynticks system (tickless)"

      depends on !ARCH_USES_GETTIMEOFFSET && GENERIC_CLOCKEVENTS

      depends on SMP

      depends on HAVE_CONTEXT_TRACKING

      depends on HAVE_VIRT_CPU_ACCOUNTING_GEN
      select NO_HZ_COMMON
      select RCU_USER_QS
      select RCU_NOCB_CPU
      select VIRT_CPU_ACCOUNTING_GEN
      select IRQ_WORK
endchoice
config NO_HZ_FULL_ALL
       bool "Full dynticks system on all CPUs by default (except CPU 0)"
       depends on NO_HZ_FULL
config NO_HZ_FULL_SYSIDLE
      bool "Detect full-system idle state for full dynticks system"
      depends on NO_HZ_FULL
      default n
config NO_HZ_FULL_SYSIDLE_SMALL
      int "Number of CPUs above which large-system approach is used"
      depends on NO_HZ_FULL_SYSIDLE
      range 1 NR_CPUS
      default 8
config NO_HZ
      bool "Old Idle dynticks config"
      depends on !ARCH_USES_GETTIMEOFFSET && GENERIC_CLOCKEVENTS
config HIGH_RES_TIMERS
      bool "High Resolution Timer Support"
      depends on !ARCH_USES_GETTIMEOFFSET && GENERIC_CLOCKEVENTS
      select TICK_ONESHOT
endmenu
endif
menu "CPU/Task time and stats accounting"
config VIRT_CPU_ACCOUNTING
      bool
choice
      prompt "Cputime accounting"
      default TICK_CPU_ACCOUNTING if !PPC64
      default VIRT_CPU_ACCOUNTING_NATIVE if PPC64
config TICK_CPU_ACCOUNTING
      bool "Simple tick based cputime accounting"
```

```
	depends on !S390 && !NO_HZ_FULL
config VIRT_CPU_ACCOUNTING_NATIVE
	bool "Deterministic task and CPU time accounting"
	depends on HAVE_VIRT_CPU_ACCOUNTING && !NO_HZ_FULL
	select VIRT_CPU_ACCOUNTING
config VIRT_CPU_ACCOUNTING_GEN
	bool "Full dynticks CPU time accounting"
	depends on HAVE_CONTEXT_TRACKING
	depends on HAVE_VIRT_CPU_ACCOUNTING_GEN
	select VIRT_CPU_ACCOUNTING
	select CONTEXT_TRACKING
config IRQ_TIME_ACCOUNTING
	bool "Fine granularity task level IRQ time accounting"
	depends on HAVE_IRQ_TIME_ACCOUNTING && !NO_HZ_FULL
endchoice
config BSD_PROCESS_ACCT
	bool "BSD Process Accounting"
	depends on MULTIUSER
config BSD_PROCESS_ACCT_V3
	bool "BSD Process Accounting version 3 file format"
	depends on BSD_PROCESS_ACCT
	default n
config TASKSTATS
	bool "Export task/process statistics through netlink"
	depends on NET
	depends on MULTIUSER
	default n
config TASK_DELAY_ACCT
	bool "Enable per-task delay accounting"
	depends on TASKSTATS
config TASK_XACCT
	bool "Enable extended accounting over taskstats"
	depends on TASKSTATS
config TASK_IO_ACCOUNTING
	bool "Enable per-task storage I/O accounting"
	depends on TASK_XACCT
endmenu
menu "RCU Subsystem"
choice
	prompt "RCU Implementation"
	default TREE_RCU
config TREE_RCU
	bool "Tree-based hierarchical RCU"
	depends on !PREEMPT && SMP
config PREEMPT_RCU
	bool "Preemptible tree-based hierarchical RCU"
	depends on PREEMPT
config TINY_RCU
	bool "UP-only small-memory-footprint RCU"
	depends on !PREEMPT && !SMP
endchoice
config SRCU
	bool
```

```
config TASKS_RCU
      bool "Task_based RCU implementation using voluntary context switch"
      default n
      select SRCU
config RCU_STALL_COMMON
      def_bool ( TREE_RCU || PREEMPT_RCU || RCU_TRACE )
config CONTEXT_TRACKING
       bool
config RCU_USER_QS
      bool "Consider userspace as in RCU extended quiescent state"
      depends on HAVE_CONTEXT_TRACKING && SMP
      select CONTEXT_TRACKING
config CONTEXT_TRACKING_FORCE
      bool "Force context tracking"
      depends on CONTEXT_TRACKING
      default y if !NO_HZ_FULL
config RCU_FANOUT
      int "Tree-based hierarchical RCU fanout value"
      range 2 64 if 64BIT
      range 2 32 if !64BIT
      depends on TREE_RCU || PREEMPT_RCU
      default 64 if 64BIT
      default 32 if !64BIT
config RCU_FANOUT_LEAF
      int "Tree-based hierarchical RCU leaf-level fanout value"
      range 2 RCU_FANOUT if 64BIT
      range 2 RCU_FANOUT if !64BIT
      depends on TREE_RCU || PREEMPT_RCU
      default 16
config RCU_FANOUT_EXACT
      bool "Disable tree-based hierarchical RCU auto-balancing"
      depends on TREE_RCU || PREEMPT_RCU
      default n
config RCU_FAST_NO_HZ
      bool "Accelerate last non-dyntick-idle CPU's grace periods"
      depends on NO_HZ_COMMON && SMP
      default n
config TREE_RCU_TRACE
      def_bool RCU_TRACE && ( TREE_RCU || PREEMPT_RCU )
      select DEBUG_FS
config RCU_BOOST
      bool "Enable RCU priority boosting"
      depends on RT_MUTEXES && PREEMPT_RCU
      default n
config RCU_KTHREAD_PRIO
      int "Real-time priority to use for RCU worker threads"
      range 1 99 if RCU_BOOST
      range 0 99 if !RCU_BOOST
      default 1 if RCU_BOOST
      default 0 if !RCU_BOOST
config RCU_BOOST_DELAY
      int "Milliseconds to delay boosting after RCU grace-period start"
      range 0 3000
```

```
	depends on RCU_BOOST
	default 500
config RCU_NOCB_CPU
	bool "Offload RCU callback processing from boot-selected CPUs"
	depends on TREE_RCU || PREEMPT_RCU
	default n
choice
	prompt "Build-forced no-CBs CPUs"
	default RCU_NOCB_CPU_NONE
	depends on RCU_NOCB_CPU
config RCU_NOCB_CPU_NONE
	bool "No build_forced no-CBs CPUs"
config RCU_NOCB_CPU_ZERO
	bool "CPU 0 is a build_forced no-CBs CPU"
config RCU_NOCB_CPU_ALL
	bool "All CPUs are build_forced no-CBs CPUs"
endchoice
config RCU_EXPEDITE_BOOT
	bool
	default n
endmenu
config BUILD_BIN2C
	bool
	default n
config IKCONFIG
	tristate "Kernel .config support"
	select BUILD_BIN2C
config IKCONFIG_PROC
	bool "Enable access to .config through /proc/config.gz"
	depends on IKCONFIG && PROC_FS
config LOG_BUF_SHIFT
	int "Kernel log buffer size (16 => 64KB, 17 => 128KB)"
	range 12 21
	default 17
	depends on PRINTK
config LOG_CPU_MAX_BUF_SHIFT
	int "CPU kernel log buffer size contribution (13 => 8 KB, 17 =>
128KB)"
	depends on SMP
	range 0 21
	default 12 if !BASE_SMALL
	default 0 if BASE_SMALL
	depends on PRINTK
config HAVE_UNSTABLE_SCHED_CLOCK
	bool
config GENERIC_SCHED_CLOCK
	bool
config ARCH_SUPPORTS_NUMA_BALANCING
	bool
config ARCH_SUPPORTS_INT128
	bool
config ARCH_WANT_NUMA_VARIABLE_LOCALITY
	bool
```

```
config NUMA_BALANCING
      bool "Memory placement aware NUMA scheduler"
      depends on ARCH_SUPPORTS_NUMA_BALANCING
      depends on !ARCH_WANT_NUMA_VARIABLE_LOCALITY
      depends on SMP && NUMA && MIGRATION
config NUMA_BALANCING_DEFAULT_ENABLED
      bool "Automatically enable NUMA aware memory/task placement"
      default y
      depends on NUMA_BALANCING
menuconfig CGROUPS
      bool "Control Group support"
      select KERNFS
if CGROUPS
config CGROUP_DEBUG
      bool "Example debug cgroup subsystem"
      default n
config CGROUP_FREEZER
      bool "Freezer cgroup subsystem"
config CGROUP_DEVICE
      bool "Device controller for cgroups"
config CPUSETS
      bool "Cpuset support"
config PROC_PID_CPUSET
      bool "Include legacy /proc/<pid>/cpuset file"
      depends on CPUSETS
      default y
config CGROUP_CPUACCT
      bool "Simple CPU accounting cgroup subsystem"
config PAGE_COUNTER
       bool
config MEMCG
      bool "Memory Resource Controller for Control Groups"
      select PAGE_COUNTER
      select EVENTFD
config MEMCG_SWAP
      bool "Memory Resource Controller Swap Extension"
      depends on MEMCG && SWAP
config MEMCG_SWAP_ENABLED
      bool "Memory Resource Controller Swap Extension enabled by default"
      depends on MEMCG_SWAP
      default y
config MEMCG_KMEM
      bool "Memory Resource Controller Kernel Memory accounting"
      depends on MEMCG
      depends on SLUB || SLAB
config CGROUP_HUGETLB
      bool "HugeTLB Resource Controller for Control Groups"
      depends on HUGETLB_PAGE
      select PAGE_COUNTER
      default n
config CGROUP_PERF
      bool "Enable perf_event per-cpu per-container group (cgroup)
monitoring"
```

```
      depends on PERF_EVENTS && CGROUPS
menuconfig CGROUP_SCHED
      bool "Group CPU scheduler"
      default n
if CGROUP_SCHED
config FAIR_GROUP_SCHED
      bool "Group scheduling for SCHED_OTHER"
      depends on CGROUP_SCHED
      default CGROUP_SCHED
config CFS_BANDWIDTH
      bool "CPU bandwidth provisioning for FAIR_GROUP_SCHED"
      depends on FAIR_GROUP_SCHED
      default n
config RT_GROUP_SCHED
      bool "Group scheduling for SCHED_RR/FIFO"
      depends on CGROUP_SCHED
      default n
endif
config BLK_CGROUP
      bool "Block IO controller"
      depends on BLOCK
      default n
config DEBUG_BLK_CGROUP
      bool "Enable Block IO controller debugging"
      depends on BLK_CGROUP
      default n
endif
config CHECKPOINT_RESTORE
      bool "Checkpoint/restore support" if EXPERT
      default n
menuconfig NAMESPACES
      bool "Namespaces support" if EXPERT
      depends on MULTIUSER
      default !EXPERT
if NAMESPACES
config UTS_NS
      bool "UTS namespace"
      default y
config IPC_NS
      bool "IPC namespace"
      depends on (SYSVIPC || POSIX_MQUEUE)
      default y
config USER_NS
      bool "User namespace"
      default n
config PID_NS
      bool "PID Namespaces"
      default y
config NET_NS
      bool "Network namespace"
      depends on NET
      default y
endif
```

```
config SCHED_AUTOGROUP
      bool "Automatic process group scheduling"
      select CGROUPS
      select CGROUP_SCHED
      select FAIR_GROUP_SCHED
config SYSFS_DEPRECATED
      bool "Enable deprecated sysfs features to support old userspace
tools"
      depends on SYSFS
      default n
config SYSFS_DEPRECATED_V2
      bool "Enable deprecated sysfs features by default"
      default n
      depends on SYSFS
      depends on SYSFS_DEPRECATED
config RELAY
      bool "Kernel->user space relay support (formerly relayfs)"
config BLK_DEV_INITRD
      bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
      depends on BROKEN || !FRV
if BLK_DEV_INITRD
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/usr/K
config
config INITRAMFS_SOURCE
      string "Initramfs source file(s)"
      default ""
config INITRAMFS_ROOT_UID
      int "User ID to map to 0 (user root)"
      depends on INITRAMFS_SOURCE!=""
      default "0"
config INITRAMFS_ROOT_GID
      int "Group ID to map to 0 (group root)"
      depends on INITRAMFS_SOURCE!=""
      default "0"
config RD_GZIP
      bool "Support initial ramdisks compressed using gzip"
      depends on BLK_DEV_INITRD
      default y
      select DECOMPRESS_GZIP
config RD_BZIP2
      bool "Support initial ramdisks compressed using bzip2"
      default y
      depends on BLK_DEV_INITRD
      select DECOMPRESS_BZIP2
config RD_LZMA
      bool "Support initial ramdisks compressed using LZMA"
      default y
      depends on BLK_DEV_INITRD
      select DECOMPRESS_LZMA
config RD_XZ
      bool "Support initial ramdisks compressed using XZ"
      depends on BLK_DEV_INITRD
```

```
	default y
	select DECOMPRESS_XZ
config RD_LZO
	bool "Support initial ramdisks compressed using LZO"
	default y
	depends on BLK_DEV_INITRD
	select DECOMPRESS_LZO
config RD_LZ4
	bool "Support initial ramdisks compressed using LZ4"
	default y
	depends on BLK_DEV_INITRD
	select DECOMPRESS_LZ4
endif
config CC_OPTIMIZE_FOR_SIZE
	bool "Optimize for size"
config SYSCTL
	bool
config ANON_INODES
	bool
config HAVE_UID16
	bool
config SYSCTL_EXCEPTION_TRACE
	bool
config SYSCTL_ARCH_UNALIGN_NO_WARN
	bool
config SYSCTL_ARCH_UNALIGN_ALLOW
	bool
config HAVE_PCSPKR_PLATFORM
	bool
config BPF
	bool
menuconfig EXPERT
	bool "Configure standard kernel features (expert users)"
	select DEBUG_KERNEL
config UID16
	bool "Enable 16-bit UID system calls" if EXPERT
	depends on HAVE_UID16 && MULTIUSER
	default y
config MULTIUSER
	bool "Multiple users, groups and capabilities support" if EXPERT
	default y
config SGETMASK_SYSCALL
	bool "sgetmask/ssetmask syscalls support" if EXPERT
	def_bool PARISC || MN10300 || BLACKFIN || M68K || PPC || MIPS || X86
|| SPARC || CRIS || MICROBLAZE || SUPERH
config SYSFS_SYSCALL
	bool "Sysfs syscall support" if EXPERT
	default y
config SYSCTL_SYSCALL
	bool "Sysctl syscall support" if EXPERT
	depends on PROC_SYSCTL
	default n
	select SYSCTL
```

```
config KALLSYMS
	bool "Load all symbols for debugging/ksymoops" if EXPERT
	default y
config KALLSYMS_ALL
	bool "Include all symbols in kallsyms"
	depends on DEBUG_KERNEL && KALLSYMS
config PRINTK
	default y
	bool "Enable support for printk" if EXPERT
	select IRQ_WORK
config BUG
	bool "BUG() support" if EXPERT
	default y
config ELF_CORE
	depends on COREDUMP
	default y
	bool "Enable ELF core dumps" if EXPERT
config PCSPKR_PLATFORM
	bool "Enable PC-Speaker support" if EXPERT
	depends on HAVE_PCSPKR_PLATFORM
	select I8253_LOCK
	default y
config BASE_FULL
	default y
	bool "Enable full-sized data structures for core" if EXPERT
config FUTEX
	bool "Enable futex support" if EXPERT
	default y
	select RT_MUTEXES
config HAVE_FUTEX_CMPXCHG
	bool
	depends on FUTEX
config EPOLL
	bool "Enable eventpoll support" if EXPERT
	default y
	select ANON_INODES
config SIGNALFD
	bool "Enable signalfd() system call" if EXPERT
	select ANON_INODES
	default y
config TIMERFD
	bool "Enable timerfd() system call" if EXPERT
	select ANON_INODES
	default y
config EVENTFD
	bool "Enable eventfd() system call" if EXPERT
	select ANON_INODES
	default y
config BPF_SYSCALL
	bool "Enable bpf() system call"
	select ANON_INODES
	select BPF
	default n
```

```
config SHMEM
	bool "Use full shmem filesystem" if EXPERT
	default y
	depends on MMU
config AIO
	bool "Enable AIO support" if EXPERT
	default y
config ADVISE_SYSCALLS
	bool "Enable madvise/fadvise syscalls" if EXPERT
	default y
config PCI_QUIRKS
	default y
	bool "Enable PCI quirk workarounds" if EXPERT
	depends on PCI
config EMBEDDED
	bool "Embedded system"
	option allnoconfig_y
	select EXPERT
config HAVE_PERF_EVENTS
	bool
config PERF_USE_VMALLOC
	bool
menu "Kernel Performance Events And Counters"
config PERF_EVENTS
	bool "Kernel performance events and counters"
	default y if PROFILING
	depends on HAVE_PERF_EVENTS
	select ANON_INODES
	select IRQ_WORK
	select SRCU
config DEBUG_PERF_USE_VMALLOC
	default n
	bool "Debug: use vmalloc to back perf mmap() buffers"
	depends on PERF_EVENTS && DEBUG_KERNEL
	select PERF_USE_VMALLOC
endmenu
config VM_EVENT_COUNTERS
	default y
	bool "Enable VM event counters for /proc/vmstat" if EXPERT
config SLUB_DEBUG
	default y
	bool "Enable SLUB debugging support" if EXPERT
	depends on SLUB && SYSFS
config COMPAT_BRK
	bool "Disable heap randomization"
	default y
choice
	prompt "Choose SLAB allocator"
	default SLUB
config SLAB
	bool "SLAB"
config SLUB
	bool "SLUB (Unqueued Allocator)"
```

```
config SLOB
      depends on EXPERT
      bool "SLOB (Simple Allocator)"
endchoice
config SLUB_CPU_PARTIAL
      default y
      depends on SLUB && SMP
      bool "SLUB per cpu partial cache"
config MMAP_ALLOW_UNINITIALIZED
      bool "Allow mmapped anonymous memory to be uninitialized"
      depends on EXPERT && !MMU
      default n
config SYSTEM_TRUSTED_KEYRING
      bool "Provide system-wide ring of trusted keys"
      depends on KEYS
config PROFILING
      bool "Profiling support"
config TRACEPOINTS
      bool
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/arch/
Kconfig
config OPROFILE
      tristate "OProfile system profiling"
      depends on PROFILING
      depends on HAVE_OPROFILE
      select RING_BUFFER
      select RING_BUFFER_ALLOW_SWAP
config OPROFILE_EVENT_MULTIPLEX
      bool "OProfile multiplexing support (EXPERIMENTAL)"
      default n
      depends on OPROFILE && X86
config HAVE_OPROFILE
      bool
config OPROFILE_NMI_TIMER
      def_bool y
      depends on PERF_EVENTS && HAVE_PERF_EVENTS_NMI && !PPC64
config KPROBES
      bool "Kprobes"
      depends on MODULES
      depends on HAVE_KPROBES
      select KALLSYMS
config JUMP_LABEL
       bool "Optimize very unlikely/likely branches"
       depends on HAVE_ARCH_JUMP_LABEL
config OPTPROBES
      def_bool y
      depends on KPROBES && HAVE_OPTPROBES
      depends on !PREEMPT
config KPROBES_ON_FTRACE
      def_bool y
      depends on KPROBES && HAVE_KPROBES_ON_FTRACE
      depends on DYNAMIC_FTRACE_WITH_REGS
```

```
config UPROBES
	def_bool n
	select PERCPU_RWSEM
config HAVE_64BIT_ALIGNED_ACCESS
	def_bool 64BIT && !HAVE_EFFICIENT_UNALIGNED_ACCESS
config HAVE_EFFICIENT_UNALIGNED_ACCESS
	bool
config ARCH_USE_BUILTIN_BSWAP
	 bool
config KRETPROBES
	def_bool y
	depends on KPROBES && HAVE_KRETPROBES
config USER_RETURN_NOTIFIER
	bool
	depends on HAVE_USER_RETURN_NOTIFIER
config HAVE_IOREMAP_PROT
	bool
config HAVE_KPROBES
	bool
config HAVE_KRETPROBES
	bool
config HAVE_OPTPROBES
	bool
config HAVE_KPROBES_ON_FTRACE
	bool
config HAVE_NMI_WATCHDOG
	bool
config HAVE_ARCH_TRACEHOOK
	bool
config HAVE_DMA_ATTRS
	bool
config HAVE_DMA_CONTIGUOUS
	bool
config GENERIC_SMP_IDLE_THREAD
	 bool
config GENERIC_IDLE_POLL_SETUP
	 bool
config ARCH_INIT_TASK
	 bool
config ARCH_TASK_STRUCT_ALLOCATOR
	bool
config ARCH_THREAD_INFO_ALLOCATOR
	bool
config HAVE_REGS_AND_STACK_ACCESS_API
	bool
config HAVE_CLK
	bool
config HAVE_DMA_API_DEBUG
	bool
config HAVE_HW_BREAKPOINT
	bool
	depends on PERF_EVENTS
config HAVE_MIXED_BREAKPOINTS_REGS
```

```
	bool
	depends on HAVE_HW_BREAKPOINT
config HAVE_USER_RETURN_NOTIFIER
	bool
config HAVE_PERF_EVENTS_NMI
	bool
config HAVE_PERF_REGS
	bool
config HAVE_PERF_USER_STACK_DUMP
	bool
config HAVE_ARCH_JUMP_LABEL
	bool
config HAVE_RCU_TABLE_FREE
	bool
config ARCH_HAVE_NMI_SAFE_CMPXCHG
	bool
config HAVE_ALIGNED_STRUCT_PAGE
	bool
config HAVE_CMPXCHG_LOCAL
	bool
config HAVE_CMPXCHG_DOUBLE
	bool
config ARCH_WANT_IPC_PARSE_VERSION
	bool
config ARCH_WANT_COMPAT_IPC_PARSE_VERSION
	bool
config ARCH_WANT_OLD_COMPAT_IPC
	select ARCH_WANT_COMPAT_IPC_PARSE_VERSION
	bool
config HAVE_ARCH_SECCOMP_FILTER
	bool
config SECCOMP_FILTER
	def_bool y
	depends on HAVE_ARCH_SECCOMP_FILTER && SECCOMP && NET
config HAVE_CC_STACKPROTECTOR
	bool
config CC_STACKPROTECTOR
	def_bool n
choice
	prompt "Stack Protector buffer overflow detection"
	depends on HAVE_CC_STACKPROTECTOR
	default CC_STACKPROTECTOR_NONE
config CC_STACKPROTECTOR_NONE
	bool "None"
config CC_STACKPROTECTOR_REGULAR
	bool "Regular"
	select CC_STACKPROTECTOR
config CC_STACKPROTECTOR_STRONG
	bool "Strong"
	select CC_STACKPROTECTOR
endchoice
config HAVE_CONTEXT_TRACKING
	bool
```

```
config HAVE_VIRT_CPU_ACCOUNTING
      bool
config HAVE_VIRT_CPU_ACCOUNTING_GEN
      bool
      default y if 64BIT
config HAVE_IRQ_TIME_ACCOUNTING
      bool
config HAVE_ARCH_TRANSPARENT_HUGEPAGE
      bool
config HAVE_ARCH_HUGE_VMAP
      bool
config HAVE_ARCH_SOFT_DIRTY
      bool
config HAVE_MOD_ARCH_SPECIFIC
      bool
config MODULES_USE_ELF_RELA
      bool
config MODULES_USE_ELF_REL
      bool
config HAVE_UNDERSCORE_SYMBOL_PREFIX
      bool
config HAVE_IRQ_EXIT_ON_IRQ_STACK
      bool
config PGTABLE_LEVELS
      int
      default 2
config ARCH_HAS_ELF_RANDOMIZE
      bool
config CLONE_BACKWARDS
      bool
config CLONE_BACKWARDS2
      bool
config CLONE_BACKWARDS3
      bool
config ODD_RT_SIGACTION
      bool
config OLD_SIGSUSPEND
      bool
config OLD_SIGSUSPEND3
      bool
config OLD_SIGACTION
      bool
config COMPAT_OLD_SIGACTION
      bool
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/kerne
l/gcov/Kconfig
menu "GCOV-based kernel profiling"
config GCOV_KERNEL
      bool "Enable gcov-based kernel profiling"
      depends on DEBUG_FS
      select CONSTRUCTORS if !UML
      default n
```

```
config ARCH_HAS_GCOV_PROFILE_ALL
      def_bool n
config GCOV_PROFILE_ALL
      bool "Profile entire Kernel"
      depends on GCOV_KERNEL
      depends on ARCH_HAS_GCOV_PROFILE_ALL
      default n
choice
      prompt "Specify GCOV format"
      depends on GCOV_KERNEL
      default GCOV_FORMAT_AUTODETECT
config GCOV_FORMAT_AUTODETECT
      bool "Autodetect"
config GCOV_FORMAT_3_4
      bool "GCC 3.4 format"
config GCOV_FORMAT_4_7
      bool "GCC 4.7 format"
endchoice
endmenu
endmenu
config HAVE_GENERIC_DMA_COHERENT
      bool
      default n
config SLABINFO
      bool
      depends on PROC_FS
      depends on SLAB || SLUB_DEBUG
      default y
config RT_MUTEXES
      bool
config BASE_SMALL
      int
      default 0 if BASE_FULL
      default 1 if !BASE_FULL
menuconfig MODULES
      bool "Enable loadable module support"
      option modules
if MODULES
config MODULE_FORCE_LOAD
      bool "Forced module loading"
      default n
config MODULE_UNLOAD
      bool "Module unloading"
config MODULE_FORCE_UNLOAD
      bool "Forced module unloading"
      depends on MODULE_UNLOAD
config MODVERSIONS
      bool "Module versioning support"
config MODULE_SRCVERSION_ALL
      bool "Source checksum for all modules"
config MODULE_SIG
      bool "Module signature verification"
      depends on MODULES
```

```
	select SYSTEM_TRUSTED_KEYRING
	select KEYS
	select CRYPTO
	select ASYMMETRIC_KEY_TYPE
	select ASYMMETRIC_PUBLIC_KEY_SUBTYPE
	select PUBLIC_KEY_ALGO_RSA
	select ASN1
	select OID_REGISTRY
	select X509_CERTIFICATE_PARSER
config MODULE_SIG_FORCE
	bool "Require modules to be validly signed"
	depends on MODULE_SIG
config MODULE_SIG_ALL
	bool "Automatically sign all modules"
	default y
	depends on MODULE_SIG
comment "Do not forget to sign required modules with scripts/sign-file"
	depends on MODULE_SIG_FORCE && !MODULE_SIG_ALL
choice
	prompt "Which hash algorithm should modules be signed with?"
	depends on MODULE_SIG
config MODULE_SIG_SHA1
	bool "Sign modules with SHA-1"
	select CRYPTO_SHA1
config MODULE_SIG_SHA224
	bool "Sign modules with SHA-224"
	select CRYPTO_SHA256
config MODULE_SIG_SHA256
	bool "Sign modules with SHA-256"
	select CRYPTO_SHA256
config MODULE_SIG_SHA384
	bool "Sign modules with SHA-384"
	select CRYPTO_SHA512
config MODULE_SIG_SHA512
	bool "Sign modules with SHA-512"
	select CRYPTO_SHA512
endchoice
config MODULE_SIG_HASH
	string
	depends on MODULE_SIG
	default "sha1" if MODULE_SIG_SHA1
	default "sha224" if MODULE_SIG_SHA224
	default "sha256" if MODULE_SIG_SHA256
	default "sha384" if MODULE_SIG_SHA384
	default "sha512" if MODULE_SIG_SHA512
config MODULE_COMPRESS
	bool "Compress modules on installation"
	depends on MODULES
choice
	prompt "Compression algorithm"
	depends on MODULE_COMPRESS
	default MODULE_COMPRESS_GZIP
config MODULE_COMPRESS_GZIP
```

```
      bool "GZIP"
config MODULE_COMPRESS_XZ
      bool "XZ"
endchoice
endif
config INIT_ALL_POSSIBLE
      bool
config STOP_MACHINE
      bool
      default y
      depends on (SMP && MODULE_UNLOAD) || HOTPLUG_CPU
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/block
/Kconfig
menuconfig BLOCK
       bool "Enable the block layer" if EXPERT
       default y
if BLOCK
config LBDAF
      bool "Support for large (2TB+) block devices and files"
      depends on !64BIT
      default y
config BLK_DEV_BSG
      bool "Block layer SG support v4"
      default y
config BLK_DEV_BSGLIB
      bool "Block layer SG support v4 helper lib"
      default n
      select BLK_DEV_BSG
config BLK_DEV_INTEGRITY
      bool "Block layer data integrity support"
      select CRC_T10DIF if BLK_DEV_INTEGRITY
config BLK_DEV_THROTTLING
      bool "Block layer bio throttling support"
      depends on BLK_CGROUP=y
      default n
config BLK_CMDLINE_PARSER
      bool "Block device command line partition parser"
      default n
menu "Partition Types"
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/block
/partitions/Kconfig
config PARTITION_ADVANCED
      bool "Advanced partition selection"
config ACORN_PARTITION
      bool "Acorn partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
config ACORN_PARTITION_CUMANA
      bool "Cumana partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config ACORN_PARTITION_EESOX
```

```
      bool "EESOX partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config ACORN_PARTITION_ICS
      bool "ICS partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config ACORN_PARTITION_ADFS
      bool "Native filecore partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config ACORN_PARTITION_POWERTEC
      bool "PowerTec partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config ACORN_PARTITION_RISCIX
      bool "RISCiX partition support" if PARTITION_ADVANCED
      default y if ARCH_ACORN
      depends on ACORN_PARTITION
config AIX_PARTITION
      bool "AIX basic partition table support" if PARTITION_ADVANCED
config OSF_PARTITION
      bool "Alpha OSF partition support" if PARTITION_ADVANCED
      default y if ALPHA
config AMIGA_PARTITION
      bool "Amiga partition table support" if PARTITION_ADVANCED
      default y if (AMIGA || AFFS_FS=y)
config ATARI_PARTITION
      bool "Atari partition table support" if PARTITION_ADVANCED
      default y if ATARI
config IBM_PARTITION
      bool "IBM disk label and partition support"
      depends on PARTITION_ADVANCED && S390
config MAC_PARTITION
      bool "Macintosh partition map support" if PARTITION_ADVANCED
      default y if (MAC || PPC_PMAC)
config MSDOS_PARTITION
      bool "PC BIOS (MSDOS partition tables) support" if PARTITION_ADVANCED
      default y
config BSD_DISKLABEL
      bool "BSD disklabel (FreeBSD partition tables) support"
      depends on PARTITION_ADVANCED && MSDOS_PARTITION
config MINIX_SUBPARTITION
      bool "Minix subpartition support"
      depends on PARTITION_ADVANCED && MSDOS_PARTITION
config SOLARIS_X86_PARTITION
      bool "Solaris (x86) partition table support"
      depends on PARTITION_ADVANCED && MSDOS_PARTITION
config UNIXWARE_DISKLABEL
      bool "Unixware slices support"
      depends on PARTITION_ADVANCED && MSDOS_PARTITION
config LDM_PARTITION
      bool "Windows Logical Disk Manager (Dynamic Disk) support"
```

```
      depends on PARTITION_ADVANCED
config LDM_DEBUG
      bool "Windows LDM extra logging"
      depends on LDM_PARTITION
config SGI_PARTITION
      bool "SGI partition support" if PARTITION_ADVANCED
      default y if DEFAULT_SGI_PARTITION
config ULTRIX_PARTITION
      bool "Ultrix partition table support" if PARTITION_ADVANCED
      default y if MACH_DECSTATION
config SUN_PARTITION
      bool "Sun partition tables support" if PARTITION_ADVANCED
      default y if (SPARC || SUN3 || SUN3X)
config KARMA_PARTITION
      bool "Karma Partition support"
      depends on PARTITION_ADVANCED
config EFI_PARTITION
      bool "EFI GUID Partition support" if PARTITION_ADVANCED
      default y
      select CRC32
config SYSV68_PARTITION
      bool "SYSV68 partition table support" if PARTITION_ADVANCED
      default y if VME
config CMDLINE_PARTITION
      bool "Command line partition support" if PARTITION_ADVANCED
      select BLK_CMDLINE_PARSER
endmenu
endif
config BLOCK_COMPAT
      bool
      depends on BLOCK && COMPAT
      default y
####################
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/plain/block
/Kconfig.iosched
if BLOCK
menu "IO Schedulers"
config IOSCHED_NOOP
      bool
      default y
config IOSCHED_DEADLINE
      tristate "Deadline I/O scheduler"
      default y
config IOSCHED_CFQ
      tristate "CFQ I/O scheduler"
      default y
config CFQ_GROUP_IOSCHED
      bool "CFQ Group Scheduling support"
      depends on IOSCHED_CFQ && BLK_CGROUP
      default n
choice
      prompt "Default I/O scheduler"
      default DEFAULT_CFQ
```

```
	config DEFAULT_DEADLINE
		bool "Deadline" if IOSCHED_DEADLINE=y
	config DEFAULT_CFQ
		bool "CFQ" if IOSCHED_CFQ=y
	config DEFAULT_NOOP
		bool "No-op"
endchoice
config DEFAULT_IOSCHED
	string
	default "deadline" if DEFAULT_DEADLINE
	default "cfq" if DEFAULT_CFQ
	default "noop" if DEFAULT_NOOP
endmenu
endif
```

Modelo de Características

El modelo de características obtenido posee 429 características y unas 273 restricciones.

```
<feature_model name='Analisis variabilidad'>
<feature_tree>
:r Menu Padre(_r)
	:o 64BIT(_r_6)
	:o X86_32(_r_7)
	:o X86_64(_r_8)
	:o X86(_r_9)
	:o INSTRUCTION_DECODER(_r_10)
	:o PERF_EVENTS_INTEL_UNCORE(_r_11)
	:m OUTPUT_FORMAT(_r_12)
	:m ARCH_DEFCONFIG(_r_13)
	:o LOCKDEP_SUPPORT(_r_14)
	:o STACKTRACE_SUPPORT(_r_15)
	:o HAVE_LATENCYTOP_SUPPORT(_r_16)
	:o MMU(_r_17)
	:o SBUS(_r_18)
	:o NEED_DMA_MAP_STATE(_r_19)
	:o NEED_SG_DMA_LENGTH(_r_20)
	:o GENERIC_ISA_DMA(_r_21)
	:o GENERIC_BUG(_r_22)
	:o GENERIC_BUG_RELATIVE_POINTERS(_r_23)
	:o GENERIC_HWEIGHT(_r_24)
	:o ARCH_MAY_HAVE_PC_FDC(_r_25)
	:o RWSEM_XCHGADD_ALGORITHM(_r_26)
	:o GENERIC_CALIBRATE_DELAY(_r_27)
	:o ARCH_HAS_CPU_RELAX(_r_28)
	:o ARCH_HAS_CACHE_LINE_SIZE(_r_29)
	:o HAVE_SETUP_PER_CPU_AREA(_r_30)
	:o NEED_PER_CPU_EMBED_FIRST_CHUNK(_r_31)
	:o NEED_PER_CPU_PAGE_FIRST_CHUNK(_r_32)
```

```
:o ARCH_HIBERNATION_POSSIBLE(_r_33)
:o ARCH_SUSPEND_POSSIBLE(_r_34)
:o ARCH_WANT_HUGE_PMD_SHARE(_r_35)
:o ARCH_WANT_GENERAL_HUGETLB(_r_36)
:o ZONE_DMA32(_r_37)
:o AUDIT_ARCH(_r_38)
:o ARCH_SUPPORTS_OPTIMIZED_INLINING(_r_39)
:o ARCH_SUPPORTS_DEBUG_PAGEALLOC(_r_40)
:o HAVE_INTEL_TXT(_r_41)
:o X86_32_SMP(_r_42)
:o X86_64_SMP(_r_43)
:o X86_HT(_r_44)
:o X86_32_LAZY_GS(_r_45)
:m ARCH_HWEIGHT_CFLAGS(_r_46)
:o ARCH_SUPPORTS_UPROBES(_r_47)
:o FIX_EARLYCON_MEM(_r_48)
:m PGTABLE_LEVELS(_r_49)
:m ARCH(_r_50)
:m KERNELVERSION(_r_51)
:m DEFCONFIG_LIST(_r_52)
:o CONSTRUCTORS(_r_53)
:o IRQ_WORK(_r_54)
:o BUILDTIME_EXTABLE_SORT(_r_55)
:o HAVE_GENERIC_DMA_COHERENT(_r_56)
:o SLABINFO(_r_57)
:o RT_MUTEXES(_r_58)
:m BASE_SMALL(_r_59)
:o INIT_ALL_POSSIBLE(_r_60)
:o STOP_MACHINE(_r_61)
:o BLOCK_COMPAT(_r_62)
:m "General setup"(_r_63)
      :o BROKEN(_r_63_64)
      :o BROKEN_ON_SMP(_r_63_65)
      :m INIT_ENV_ARG_LIMIT(_r_63_66)
      :m CROSS_COMPILE(_r_63_67)
      :o COMPILE_TEST(_r_63_68)
      :m LOCALVERSION(_r_63_69)
      :o LOCALVERSION_AUTO(_r_63_70)
      :o HAVE_KERNEL_GZIP(_r_63_71)
      :o HAVE_KERNEL_BZIP2(_r_63_72)
      :o HAVE_KERNEL_LZMA(_r_63_73)
      :o HAVE_KERNEL_XZ(_r_63_74)
      :o HAVE_KERNEL_LZO(_r_63_75)
      :o HAVE_KERNEL_LZ4(_r_63_76)
      :m DEFAULT_HOSTNAME(_r_63_77)
      :o SWAP(_r_63_78)
      :o SYSVIPC(_r_63_79)
      :o SYSVIPC_SYSCTL(_r_63_80)
      :o POSIX_MQUEUE(_r_63_81)
      :o POSIX_MQUEUE_SYSCTL(_r_63_82)
      :o CROSS_MEMORY_ATTACH(_r_63_83)
      :o FHANDLE(_r_63_84)
      :o USELIB(_r_63_85)
```

```
:o AUDIT(_r_63_86)
:o HAVE_ARCH_AUDITSYSCALL(_r_63_87)
:o AUDITSYSCALL(_r_63_88)
:o AUDIT_WATCH(_r_63_89)
:o AUDIT_TREE(_r_63_90)
:o CLOCKSOURCE_WATCHDOG(_r_63_91)
:o ARCH_CLOCKSOURCE_DATA(_r_63_92)
:o CLOCKSOURCE_VALIDATE_LAST_CYCLE(_r_63_93)
:o GENERIC_TIME_VSYSCALL(_r_63_94)
:o GENERIC_TIME_VSYSCALL_OLD(_r_63_95)
:o ARCH_USES_GETTIMEOFFSET(_r_63_96)
:o GENERIC_CLOCKEVENTS(_r_63_97)
      :m "Timers subsystem"(_r_63_97_323)
            :o TICK_ONESHOT(_r_63_97_323_324)
            :o NO_HZ_COMMON(_r_63_97_323_325)
            :o NO_HZ_FULL_ALL(_r_63_97_323_326)
            :o NO_HZ_FULL_SYSIDLE(_r_63_97_323_327)
            :m NO_HZ_FULL_SYSIDLE_SMALL(_r_63_97_323_328)
            :o NO_HZ(_r_63_97_323_329)
            :o HIGH_RES_TIMERS(_r_63_97_323_330)
            :m "Timer tick handling"(_r_63_97_323_331)
                  :g [1,1]
                        : HZ_PERIODIC(_r_63_97_323_331_332)
                        : NO_HZ_IDLE(_r_63_97_323_331_333)
                        : NO_HZ_FULL(_r_63_97_323_331_334)
:o ARCH_HAS_TICK_BROADCAST(_r_63_98)
:o GENERIC_CLOCKEVENTS_BROADCAST(_r_63_99)
:o GENERIC_CLOCKEVENTS_MIN_ADJUST(_r_63_100)
:o GENERIC_CMOS_UPDATE(_r_63_101)
:o BUILD_BIN2C(_r_63_102)
:o IKCONFIG(_r_63_103)
:o IKCONFIG_PROC(_r_63_104)
:m LOG_BUF_SHIFT(_r_63_105)
:m LOG_CPU_MAX_BUF_SHIFT(_r_63_106)
:o HAVE_UNSTABLE_SCHED_CLOCK(_r_63_107)
:o GENERIC_SCHED_CLOCK(_r_63_108)
:o ARCH_SUPPORTS_NUMA_BALANCING(_r_63_109)
:o ARCH_SUPPORTS_INT128(_r_63_110)
:o ARCH_WANT_NUMA_VARIABLE_LOCALITY(_r_63_111)
:o NUMA_BALANCING(_r_63_112)
:o NUMA_BALANCING_DEFAULT_ENABLED(_r_63_113)
:o CHECKPOINT_RESTORE(_r_63_114)
:o SCHED_AUTOGROUP(_r_63_115)
:o SYSFS_DEPRECATED(_r_63_116)
:o SYSFS_DEPRECATED_V2(_r_63_117)
:o RELAY(_r_63_118)
:o BLK_DEV_INITRD(_r_63_119)
      :m INITRAMFS_SOURCE(_r_63_119_335)
      :m INITRAMFS_ROOT_UID(_r_63_119_336)
      :m INITRAMFS_ROOT_GID(_r_63_119_337)
      :o RD_GZIP(_r_63_119_338)
      :o RD_BZIP2(_r_63_119_339)
      :o RD_LZMA(_r_63_119_340)
```

```
        :o RD_XZ(_r_63_119_341)
        :o RD_LZO(_r_63_119_342)
        :o RD_LZ4(_r_63_119_343)
:o CC_OPTIMIZE_FOR_SIZE(_r_63_120)
:o SYSCTL(_r_63_121)
:o ANON_INODES(_r_63_122)
:o HAVE_UID16(_r_63_123)
:o SYSCTL_EXCEPTION_TRACE(_r_63_124)
:o SYSCTL_ARCH_UNALIGN_NO_WARN(_r_63_125)
:o SYSCTL_ARCH_UNALIGN_ALLOW(_r_63_126)
:o HAVE_PCSPKR_PLATFORM(_r_63_127)
:o BPF(_r_63_128)
:o UID16(_r_63_129)
:o MULTIUSER(_r_63_130)
:o SGETMASK_SYSCALL(_r_63_131)
:o SYSFS_SYSCALL(_r_63_132)
:o SYSCTL_SYSCALL(_r_63_133)
:o KALLSYMS(_r_63_134)
:o KALLSYMS_ALL(_r_63_135)
:o PRINTK(_r_63_136)
:o BUG(_r_63_137)
:o ELF_CORE(_r_63_138)
:o PCSPKR_PLATFORM(_r_63_139)
:o BASE_FULL(_r_63_140)
:o FUTEX(_r_63_141)
:o HAVE_FUTEX_CMPXCHG(_r_63_142)
:o EPOLL(_r_63_143)
:o SIGNALFD(_r_63_144)
:o TIMERFD(_r_63_145)
:o EVENTFD(_r_63_146)
:o BPF_SYSCALL(_r_63_147)
:o SHMEM(_r_63_148)
:o AIO(_r_63_149)
:o ADVISE_SYSCALLS(_r_63_150)
:o PCI_QUIRKS(_r_63_151)
:o EMBEDDED(_r_63_152)
:o HAVE_PERF_EVENTS(_r_63_153)
:o PERF_USE_VMALLOC(_r_63_154)
:o VM_EVENT_COUNTERS(_r_63_155)
:o SLUB_DEBUG(_r_63_156)
:o COMPAT_BRK(_r_63_157)
:o SLUB_CPU_PARTIAL(_r_63_158)
:o MMAP_ALLOW_UNINITIALIZED(_r_63_159)
:o SYSTEM_TRUSTED_KEYRING(_r_63_160)
:o PROFILING(_r_63_161)
:o TRACEPOINTS(_r_63_162)
:o OPROFILE(_r_63_163)
:o OPROFILE_EVENT_MULTIPLEX(_r_63_164)
:o HAVE_OPROFILE(_r_63_165)
:o OPROFILE_NMI_TIMER(_r_63_166)
:o KPROBES(_r_63_167)
:o JUMP_LABEL(_r_63_168)
:o OPTPROBES(_r_63_169)
```

```
:o KPROBES_ON_FTRACE(_r_63_170)
:o UPROBES(_r_63_171)
:o HAVE_64BIT_ALIGNED_ACCESS(_r_63_172)
:o HAVE_EFFICIENT_UNALIGNED_ACCESS(_r_63_173)
:o ARCH_USE_BUILTIN_BSWAP(_r_63_174)
:o KRETPROBES(_r_63_175)
:o USER_RETURN_NOTIFIER(_r_63_176)
:o HAVE_IOREMAP_PROT(_r_63_177)
:o HAVE_KPROBES(_r_63_178)
:o HAVE_KRETPROBES(_r_63_179)
:o HAVE_OPTPROBES(_r_63_180)
:o HAVE_KPROBES_ON_FTRACE(_r_63_181)
:o HAVE_NMI_WATCHDOG(_r_63_182)
:o HAVE_ARCH_TRACEHOOK(_r_63_183)
:o HAVE_DMA_ATTRS(_r_63_184)
:o HAVE_DMA_CONTIGUOUS(_r_63_185)
:o GENERIC_SMP_IDLE_THREAD(_r_63_186)
:o GENERIC_IDLE_POLL_SETUP(_r_63_187)
:o ARCH_INIT_TASK(_r_63_188)
:o ARCH_TASK_STRUCT_ALLOCATOR(_r_63_189)
:o ARCH_THREAD_INFO_ALLOCATOR(_r_63_190)
:o HAVE_REGS_AND_STACK_ACCESS_API(_r_63_191)
:o HAVE_CLK(_r_63_192)
:o HAVE_DMA_API_DEBUG(_r_63_193)
:o HAVE_HW_BREAKPOINT(_r_63_194)
:o HAVE_MIXED_BREAKPOINTS_REGS(_r_63_195)
:o HAVE_USER_RETURN_NOTIFIER(_r_63_196)
:o HAVE_PERF_EVENTS_NMI(_r_63_197)
:o HAVE_PERF_REGS(_r_63_198)
:o HAVE_PERF_USER_STACK_DUMP(_r_63_199)
:o HAVE_ARCH_JUMP_LABEL(_r_63_200)
:o HAVE_RCU_TABLE_FREE(_r_63_201)
:o ARCH_HAVE_NMI_SAFE_CMPXCHG(_r_63_202)
:o HAVE_ALIGNED_STRUCT_PAGE(_r_63_203)
:o HAVE_CMPXCHG_LOCAL(_r_63_204)
:o HAVE_CMPXCHG_DOUBLE(_r_63_205)
:o ARCH_WANT_IPC_PARSE_VERSION(_r_63_206)
:o ARCH_WANT_COMPAT_IPC_PARSE_VERSION(_r_63_207)
:o ARCH_WANT_OLD_COMPAT_IPC(_r_63_208)
:o HAVE_ARCH_SECCOMP_FILTER(_r_63_209)
:o SECCOMP_FILTER(_r_63_210)
:o HAVE_CC_STACKPROTECTOR(_r_63_211)
:o CC_STACKPROTECTOR(_r_63_212)
:o HAVE_CONTEXT_TRACKING(_r_63_213)
:o HAVE_VIRT_CPU_ACCOUNTING(_r_63_214)
:o HAVE_VIRT_CPU_ACCOUNTING_GEN(_r_63_215)
:o HAVE_IRQ_TIME_ACCOUNTING(_r_63_216)
:o HAVE_ARCH_TRANSPARENT_HUGEPAGE(_r_63_217)
:o HAVE_ARCH_HUGE_VMAP(_r_63_218)
:o HAVE_ARCH_SOFT_DIRTY(_r_63_219)
:o HAVE_MOD_ARCH_SPECIFIC(_r_63_220)
:o MODULES_USE_ELF_RELA(_r_63_221)
:o MODULES_USE_ELF_REL(_r_63_222)
```

```
:o HAVE_UNDERSCORE_SYMBOL_PREFIX(_r_63_223)
:o HAVE_IRQ_EXIT_ON_IRQ_STACK(_r_63_224)
:m PGTABLE_LEVELS(_r_63_225)
:o ARCH_HAS_ELF_RANDOMIZE(_r_63_226)
:o CLONE_BACKWARDS(_r_63_227)
:o CLONE_BACKWARDS2(_r_63_228)
:o CLONE_BACKWARDS3(_r_63_229)
:o ODD_RT_SIGACTION(_r_63_230)
:o OLD_SIGSUSPEND(_r_63_231)
:o OLD_SIGSUSPEND3(_r_63_232)
:o OLD_SIGACTION(_r_63_233)
:o COMPAT_OLD_SIGACTION(_r_63_234)
:m "Kernel compression mode"(_r_63_235)
      :g [1,1]
            : KERNEL_GZIP(_r_63_235_236)
            : KERNEL_BZIP2(_r_63_235_237)
            : KERNEL_LZMA(_r_63_235_238)
            : KERNEL_XZ(_r_63_235_239)
            : KERNEL_LZO(_r_63_235_240)
            : KERNEL_LZ4(_r_63_235_241)
:m "Choose SLAB allocator"(_r_63_242)
      :g [1,1]
            : SLAB(_r_63_242_243)
            : SLUB(_r_63_242_244)
            : SLOB(_r_63_242_245)
:m "Stack Protector buffer overflow detection"(_r_63_246)
      :g [1,1]
            : CC_STACKPROTECTOR_NONE(_r_63_246_247)
            : CC_STACKPROTECTOR_REGULAR(_r_63_246_248)
            : CC_STACKPROTECTOR_STRONG(_r_63_246_249)
:o CGROUPS(_r_63_250)
      :o CGROUP_DEBUG(_r_63_250_356)
      :o CGROUP_FREEZER(_r_63_250_357)
      :o CGROUP_DEVICE(_r_63_250_358)
      :o CPUSETS(_r_63_250_359)
      :o PROC_PID_CPUSET(_r_63_250_360)
      :o CGROUP_CPUACCT(_r_63_250_361)
      :o PAGE_COUNTER(_r_63_250_362)
      :o MEMCG(_r_63_250_363)
      :o MEMCG_SWAP(_r_63_250_364)
      :o MEMCG_SWAP_ENABLED(_r_63_250_365)
      :o MEMCG_KMEM(_r_63_250_366)
      :o CGROUP_HUGETLB(_r_63_250_367)
      :o CGROUP_PERF(_r_63_250_368)
      :o BLK_CGROUP(_r_63_250_369)
      :o DEBUG_BLK_CGROUP(_r_63_250_370)
      :o CGROUP_SCHED(_r_63_250_371)
            :o FAIR_GROUP_SCHED(_r_63_250_371_431)
            :o CFS_BANDWIDTH(_r_63_250_371_432)
            :o RT_GROUP_SCHED(_r_63_250_371_433)
:o NAMESPACES(_r_63_251)
      :o UTS_NS(_r_63_251_372)
      :o IPC_NS(_r_63_251_373)
```

```
                        :o USER_NS(_r_63_251_374)
                        :o PID_NS(_r_63_251_375)
                        :o NET_NS(_r_63_251_376)
                  :o EXPERT(_r_63_252)
                  :m "IRQ subsystem"(_r_63_253)
                        :o MAY_HAVE_SPARSE_IRQ(_r_63_253_254)
                        :o GENERIC_IRQ_LEGACY(_r_63_253_255)
                        :o GENERIC_IRQ_PROBE(_r_63_253_256)
                        :o GENERIC_IRQ_SHOW(_r_63_253_257)
                        :o GENERIC_IRQ_SHOW_LEVEL(_r_63_253_258)
                        :o GENERIC_IRQ_LEGACY_ALLOC_HWIRQ(_r_63_253_259)
                        :o GENERIC_PENDING_IRQ(_r_63_253_260)
                        :o AUTO_IRQ_AFFINITY(_r_63_253_261)
                        :o HARDIRQS_SW_RESEND(_r_63_253_262)
                        :o IRQ_PREFLOW_FASTEOI(_r_63_253_263)
                        :o IRQ_EDGE_EOI_HANDLER(_r_63_253_264)
                        :o GENERIC_IRQ_CHIP(_r_63_253_265)
                        :o IRQ_DOMAIN(_r_63_253_266)
                        :o IRQ_DOMAIN_HIERARCHY(_r_63_253_267)
                        :o GENERIC_MSI_IRQ(_r_63_253_268)
                        :o GENERIC_MSI_IRQ_DOMAIN(_r_63_253_269)
                        :o HANDLE_DOMAIN_IRQ(_r_63_253_270)
                        :o IRQ_DOMAIN_DEBUG(_r_63_253_271)
                        :o IRQ_FORCED_THREADING(_r_63_253_272)
                        :o SPARSE_IRQ(_r_63_253_273)
                  :m "CPU/Task time and stats accounting"(_r_63_274)
                        :o VIRT_CPU_ACCOUNTING(_r_63_274_275)
                        :o BSD_PROCESS_ACCT(_r_63_274_276)
                        :o BSD_PROCESS_ACCT_V3(_r_63_274_277)
                        :o TASKSTATS(_r_63_274_278)
                        :o TASK_DELAY_ACCT(_r_63_274_279)
                        :o TASK_XACCT(_r_63_274_280)
                        :o TASK_IO_ACCOUNTING(_r_63_274_281)
                        :m "Cputime accounting"(_r_63_274_282)
                              :g [1,1]
                                    : TICK_CPU_ACCOUNTING(_r_63_274_282_283)
                                    :
VIRT_CPU_ACCOUNTING_NATIVE(_r_63_274_282_284)
                                    : VIRT_CPU_ACCOUNTING_GEN(_r_63_274_282_285)
                                    : IRQ_TIME_ACCOUNTING(_r_63_274_282_286)
                  :m "RCU Subsystem"(_r_63_287)
                        :o SRCU(_r_63_287_288)
                        :o TASKS_RCU(_r_63_287_289)
                        :o RCU_STALL_COMMON(_r_63_287_290)
                        :o CONTEXT_TRACKING(_r_63_287_291)
                        :o RCU_USER_QS(_r_63_287_292)
                        :o CONTEXT_TRACKING_FORCE(_r_63_287_293)
                        :m RCU_FANOUT(_r_63_287_294)
                        :m RCU_FANOUT_LEAF(_r_63_287_295)
                        :o RCU_FANOUT_EXACT(_r_63_287_296)
                        :o RCU_FAST_NO_HZ(_r_63_287_297)
                        :o TREE_RCU_TRACE(_r_63_287_298)
                        :o RCU_BOOST(_r_63_287_299)
```

```
                        :m RCU_KTHREAD_PRIO(_r_63_287_300)
                        :m RCU_BOOST_DELAY(_r_63_287_301)
                        :o RCU_NOCB_CPU(_r_63_287_302)
                        :o RCU_EXPEDITE_BOOT(_r_63_287_303)
                        :m "RCU Implementation"(_r_63_287_304)
                              :g [1,1]
                                    : TREE_RCU(_r_63_287_304_305)
                                    : PREEMPT_RCU(_r_63_287_304_306)
                                    : TINY_RCU(_r_63_287_304_307)
                        :m "Build-forced no-CBs CPUs"(_r_63_287_308)
                              :g [1,1]
                                    : RCU_NOCB_CPU_NONE(_r_63_287_308_309)
                                    : RCU_NOCB_CPU_ZERO(_r_63_287_308_310)
                                    : RCU_NOCB_CPU_ALL(_r_63_287_308_311)
                  :m "Kernel Performance Events And Counters"(_r_63_312)
                        :o PERF_EVENTS(_r_63_312_313)
                        :o DEBUG_PERF_USE_VMALLOC(_r_63_312_314)
                  :m "GCOV-based kernel profiling"(_r_63_315)
                        :o GCOV_KERNEL(_r_63_315_316)
                        :o ARCH_HAS_GCOV_PROFILE_ALL(_r_63_315_317)
                        :o GCOV_PROFILE_ALL(_r_63_315_318)
                        :m "Specify GCOV format"(_r_63_315_319)
                              :g [1,1]
                                    : GCOV_FORMAT_AUTODETECT(_r_63_315_319_320)
                                    : GCOV_FORMAT_3_4(_r_63_315_319_321)
                                    : GCOV_FORMAT_4_7(_r_63_315_319_322)
            :o MODULES(_r_344)
                  :o MODULE_FORCE_LOAD(_r_344_377)
                  :o MODULE_UNLOAD(_r_344_378)
                  :o MODULE_FORCE_UNLOAD(_r_344_379)
                  :o MODVERSIONS(_r_344_380)
                  :o MODULE_SRCVERSION_ALL(_r_344_381)
                  :o MODULE_SIG(_r_344_382)
                  :o MODULE_SIG_FORCE(_r_344_383)
                  :o MODULE_SIG_ALL(_r_344_384)
                  :m MODULE_SIG_HASH(_r_344_385)
                  :o MODULE_COMPRESS(_r_344_386)
                  :m "Which hash algorithm should modules be signed
with?"(_r_344_387)
                        :g [1,1]
                              : MODULE_SIG_SHA1(_r_344_387_388)
                              : MODULE_SIG_SHA224(_r_344_387_389)
                              : MODULE_SIG_SHA256(_r_344_387_390)
                              : MODULE_SIG_SHA384(_r_344_387_391)
                              : MODULE_SIG_SHA512(_r_344_387_392)
                  :m "Compression algorithm"(_r_344_393)
                        :g [1,1]
                              : MODULE_COMPRESS_GZIP(_r_344_393_394)
                              : MODULE_COMPRESS_XZ(_r_344_393_395)
            :o BLOCK(_r_345)
                  :o LBDAF(_r_345_396)
                  :o BLK_DEV_BSG(_r_345_397)
                  :o BLK_DEV_BSGLIB(_r_345_398)
```

```
                :o BLK_DEV_INTEGRITY(_r_345_399)
                :o BLK_DEV_THROTTLING(_r_345_400)
                :o BLK_CMDLINE_PARSER(_r_345_401)
                :m "Partition Types"(_r_345_402)
                      :o PARTITION_ADVANCED(_r_345_402_403)
                      :o ACORN_PARTITION(_r_345_402_404)
                      :o ACORN_PARTITION_CUMANA(_r_345_402_405)
                      :o ACORN_PARTITION_EESOX(_r_345_402_406)
                      :o ACORN_PARTITION_ICS(_r_345_402_407)
                      :o ACORN_PARTITION_ADFS(_r_345_402_408)
                      :o ACORN_PARTITION_POWERTEC(_r_345_402_409)
                      :o ACORN_PARTITION_RISCIX(_r_345_402_410)
                      :o AIX_PARTITION(_r_345_402_411)
                      :o OSF_PARTITION(_r_345_402_412)
                      :o AMIGA_PARTITION(_r_345_402_413)
                      :o ATARI_PARTITION(_r_345_402_414)
                      :o IBM_PARTITION(_r_345_402_415)
                      :o MAC_PARTITION(_r_345_402_416)
                      :o MSDOS_PARTITION(_r_345_402_417)
                      :o BSD_DISKLABEL(_r_345_402_418)
                      :o MINIX_SUBPARTITION(_r_345_402_419)
                      :o SOLARIS_X86_PARTITION(_r_345_402_420)
                      :o UNIXWARE_DISKLABEL(_r_345_402_421)
                      :o LDM_PARTITION(_r_345_402_422)
                      :o LDM_DEBUG(_r_345_402_423)
                      :o SGI_PARTITION(_r_345_402_424)
                      :o ULTRIX_PARTITION(_r_345_402_425)
                      :o SUN_PARTITION(_r_345_402_426)
                      :o KARMA_PARTITION(_r_345_402_427)
                      :o EFI_PARTITION(_r_345_402_428)
                      :o SYSV68_PARTITION(_r_345_402_429)
                      :o CMDLINE_PARTITION(_r_345_402_430)
                :m "IO Schedulers"(_r_345_346)
                      :o IOSCHED_NOOP(_r_345_346_347)
                      :o IOSCHED_DEADLINE(_r_345_346_348)
                      :o IOSCHED_CFQ(_r_345_346_349)
                      :o CFQ_GROUP_IOSCHED(_r_345_346_350)
                      :m DEFAULT_IOSCHED(_r_345_346_351)
                      :m "Default I/O scheduler"(_r_345_346_352)
                            :g [1,1]
                                  : DEFAULT_DEADLINE(_r_345_346_352_353)
                                  : DEFAULT_CFQ(_r_345_346_352_354)
                                  : DEFAULT_NOOP(_r_345_346_352_355)
</feature_tree>
<constraints>
constraint_0:~_r_7 or ~_r_6
constraint_1:~_r_8 or _r_6
constraint_2:~_r_63_83 or _r_17
constraint_3:_r_63_88 or ~_r_63_86 or ~_r_63_87
constraint_4:~_r_63_88 or _r_63_86
constraint_5:~_r_63_88 or _r_63_87
constraint_6:~_r_63_89 or _r_63_88
constraint_7:~_r_63_90 or _r_63_88
```

```
constraint_8:~_r_63_99 or _r_63_97
constraint_9:~_r_63_103 or _r_63_102
constraint_10:~_r_63_113 or _r_63_112
constraint_11:~_r_63_133 or _r_63_121
constraint_12:~_r_63_136 or _r_54
constraint_13:~_r_63_139 or _r_63_127
constraint_14:~_r_63_141 or _r_58
constraint_15:~_r_63_142 or _r_63_141
constraint_16:~_r_63_143 or _r_63_122
constraint_17:~_r_63_144 or _r_63_122
constraint_18:~_r_63_145 or _r_63_122
constraint_19:~_r_63_146 or _r_63_122
constraint_20:~_r_63_147 or _r_63_122
constraint_21:~_r_63_147 or _r_63_128
constraint_22:~_r_63_148 or _r_17
constraint_23:_r_63_164 or ~_r_63_163 or ~_r_9
constraint_24:~_r_63_164 or _r_63_163
constraint_25:~_r_63_164 or _r_9
constraint_26:~_r_63_167 or _r_63_134
constraint_27:~_r_63_195 or _r_63_194
constraint_28:~_r_63_208 or _r_63_207
constraint_29:~_r_63_235 or _r_63_71 or _r_63_72 or _r_63_73 or _r_63_74 or
_r_63_75 or _r_63_76
constraint_30:_r_63_235 or ~_r_63_71
constraint_31:_r_63_235 or ~_r_63_72
constraint_32:_r_63_235 or ~_r_63_73
constraint_33:_r_63_235 or ~_r_63_74
constraint_34:_r_63_235 or ~_r_63_75
constraint_35:_r_63_235 or ~_r_63_76
constraint_36:~_r_63_235_236 or _r_63_71
constraint_37:~_r_63_235_237 or _r_63_72
constraint_38:~_r_63_235_238 or _r_63_73
constraint_39:~_r_63_235_239 or _r_63_74
constraint_40:~_r_63_235_240 or _r_63_75
constraint_41:~_r_63_235_241 or _r_63_76
constraint_42:~_r_63_246 or _r_63_211
constraint_43:~_r_63_246_248 or _r_63_212
constraint_44:~_r_63_246_249 or _r_63_212
constraint_45:~_r_63_251 or _r_63_130
constraint_46:~_r_63_253_267 or _r_63_253_266
constraint_47:~_r_63_253_269 or _r_63_253_267
constraint_48:~_r_63_253_269 or _r_63_253_268
constraint_49:~_r_63_274_276 or _r_63_130
constraint_50:~_r_63_274_277 or _r_63_274_276
constraint_51:~_r_63_274_279 or _r_63_274_278
constraint_52:~_r_63_274_280 or _r_63_274_278
constraint_53:~_r_63_274_281 or _r_63_274_280
constraint_54:~_r_63_274_282_284 or _r_63_274_275
constraint_55:_r_63_274_282_285 or ~_r_63_213 or ~_r_63_215
constraint_56:~_r_63_274_282_285 or _r_63_213
constraint_57:~_r_63_274_282_285 or _r_63_215
constraint_58:~_r_63_274_282_285 or _r_63_274_275
constraint_59:~_r_63_287_289 or _r_63_287_288
```

```
constraint_60:~_r_63_287_292 or _r_63_287_291
constraint_61:~_r_63_287_293 or _r_63_287_291
constraint_62:~_r_63_287_301 or _r_63_287_299
constraint_63:~_r_63_287_308 or _r_63_287_302
constraint_64:~_r_63_312_313 or _r_63_153
constraint_65:~_r_63_312_313 or _r_63_122
constraint_66:~_r_63_312_313 or _r_54
constraint_67:~_r_63_312_313 or _r_63_287_288
constraint_68:~_r_63_312_314 or _r_63_154
constraint_69:_r_63_315_318 or ~_r_63_315_316 or ~_r_63_315_317
constraint_70:~_r_63_315_318 or _r_63_315_316
constraint_71:~_r_63_315_318 or _r_63_315_317
constraint_72:~_r_63_315_319 or _r_63_315_316
constraint_73:~_r_63_97_323_325 or _r_63_97_323_324
constraint_74:~_r_63_97_323_328 or _r_63_97_323_327
constraint_75:~_r_63_97_323_330 or _r_63_97_323_324
constraint_76:~_r_63_97_323_331_333 or _r_63_97_323_325
constraint_77:~_r_63_97_323_331_334 or _r_63_97_323_325
constraint_78:~_r_63_97_323_331_334 or _r_63_287_292
constraint_79:~_r_63_97_323_331_334 or _r_63_287_302
constraint_80:~_r_63_97_323_331_334 or _r_63_274_282_285
constraint_81:~_r_63_97_323_331_334 or _r_54
constraint_82:~_r_63_119_338 or _r_63_119
constraint_83:~_r_63_119_339 or _r_63_119
constraint_84:~_r_63_119_340 or _r_63_119
constraint_85:~_r_63_119_341 or _r_63_119
constraint_86:~_r_63_119_342 or _r_63_119
constraint_87:~_r_63_119_343 or _r_63_119
constraint_88:~_r_10 or _r_63_167 or _r_63_312_313 or _r_63_171
constraint_89:_r_10 or ~_r_63_167
constraint_90:_r_10 or ~_r_63_312_313
constraint_91:_r_10 or ~_r_63_171
constraint_92:~_r_22 or _r_63_137
constraint_93:~_r_57 or _r_63_242_243 or _r_63_156
constraint_94:_r_57 or ~_r_63_242_243
constraint_95:_r_57 or ~_r_63_156
constraint_96:_r_63_78 or ~_r_17 or ~_r_345
constraint_97:~_r_63_78 or _r_17
constraint_98:~_r_63_78 or _r_345
constraint_99:_r_63_80 or ~_r_63_79 or ~_r_63_121
constraint_100:~_r_63_80 or _r_63_79
constraint_101:~_r_63_80 or _r_63_121
constraint_102:_r_63_82 or ~_r_63_81 or ~_r_63_121
constraint_103:~_r_63_82 or _r_63_81
constraint_104:~_r_63_82 or _r_63_121
constraint_105:~_r_63_105 or _r_63_136
constraint_106:_r_63_129 or ~_r_63_123 or ~_r_63_130
constraint_107:~_r_63_129 or _r_63_123
constraint_108:~_r_63_129 or _r_63_130
constraint_109:_r_63_163 or ~_r_63_161 or ~_r_63_165
constraint_110:~_r_63_163 or _r_63_161
constraint_111:~_r_63_163 or _r_63_165
constraint_112:_r_63_167 or ~_r_344 or ~_r_63_178
```

```
constraint_113:~_r_63_167 or _r_344
constraint_114:~_r_63_167 or _r_63_178
constraint_115:~_r_63_168 or _r_63_200
constraint_116:_r_63_175 or ~_r_63_167 or ~_r_63_179
constraint_117:~_r_63_175 or _r_63_167
constraint_118:~_r_63_175 or _r_63_179
constraint_119:~_r_63_176 or _r_63_196
constraint_120:~_r_63_194 or _r_63_312_313
constraint_121:~_r_63_242_245 or _r_63_252
constraint_122:~_r_63_287_294 or _r_63_287_304_305 or _r_63_287_304_306
constraint_123:_r_63_287_294 or ~_r_63_287_304_305
constraint_124:_r_63_287_294 or ~_r_63_287_304_306
constraint_125:~_r_63_287_295 or _r_63_287_304_305 or _r_63_287_304_306
constraint_126:_r_63_287_295 or ~_r_63_287_304_305
constraint_127:_r_63_287_295 or ~_r_63_287_304_306
constraint_128:~_r_63_287_296 or _r_63_287_304_305 or _r_63_287_304_306
constraint_129:_r_63_287_296 or ~_r_63_287_304_305
constraint_130:_r_63_287_296 or ~_r_63_287_304_306
constraint_131:_r_63_287_299 or ~_r_58 or ~_r_63_287_304_306
constraint_132:~_r_63_287_299 or _r_58
constraint_133:~_r_63_287_299 or _r_63_287_304_306
constraint_134:~_r_63_287_302 or _r_63_287_304_305 or _r_63_287_304_306
constraint_135:_r_63_287_302 or ~_r_63_287_304_305
constraint_136:_r_63_287_302 or ~_r_63_287_304_306
constraint_137:~_r_63_97_323_326 or _r_63_97_323_331_334
constraint_138:~_r_63_97_323_327 or _r_63_97_323_331_334
constraint_139:~_r_7 or _r_63_123
constraint_140:~_r_9 or _r_63_315_317
constraint_141:~_r_9 or _r_63_107
constraint_142:~_r_8 or ~_r_9 or _r_63_109
constraint_143:~_r_8 or ~_r_9 or _r_63_110
constraint_144:~_r_9 or _r_63_165
constraint_145:~_r_9 or _r_63_127
constraint_146:~_r_9 or _r_63_153
constraint_147:~_r_9 or _r_63_177
constraint_148:~_r_9 or _r_63_178
constraint_149:~_r_9 or _r_63_184
constraint_150:~_r_9 or _r_63_185
constraint_151:~_r_9 or _r_63_179
constraint_152:~_r_9 or _r_63_180
constraint_153:~_r_9 or _r_63_181
constraint_154:~_r_9 or _r_63_124
constraint_155:~_r_9 or _r_63_183
constraint_156:~_r_7 or ~_r_9 or _r_56
constraint_157:~_r_9 or _r_63_173
constraint_158:~_r_9 or _r_63_191
constraint_159:~_r_9 or _r_63_193
constraint_160:~_r_9 or _r_63_71
constraint_161:~_r_9 or _r_63_72
constraint_162:~_r_9 or _r_63_73
constraint_163:~_r_9 or _r_63_74
constraint_164:~_r_9 or _r_63_75
constraint_165:~_r_9 or _r_63_76
```

```
constraint_166:~_r_9 or _r_63_194
constraint_167:~_r_9 or _r_63_195
constraint_168:~_r_9 or _r_63_312_313
constraint_169:~_r_9 or _r_63_197
constraint_170:~_r_9 or _r_63_198
constraint_171:~_r_9 or _r_63_199
constraint_172:~_r_9 or _r_63_122
constraint_173:~_r_63_242_244 or ~_r_9 or _r_63_203
constraint_174:~_r_9 or _r_63_204
constraint_175:~_r_9 or _r_63_205
constraint_176:~_r_9 or _r_63_196
constraint_177:~_r_9 or _r_63_226
constraint_178:~_r_9 or _r_63_200
constraint_179:~_r_9 or _r_63_253_273
constraint_180:~_r_9 or _r_63_253_256
constraint_181:~_r_9 or _r_63_253_257
constraint_182:~_r_9 or _r_63_100
constraint_183:~_r_9 or _r_63_253_272
constraint_184:~_r_9 or _r_63_217
constraint_185:~_r_8 or ~_r_9 or _r_63_218
constraint_186:~_r_7 or ~_r_8 or ~_r_9 or _r_63_218
constraint_187:~_r_9 or _r_63_202
constraint_188:~_r_9 or _r_63_186
constraint_189:~_r_7 or ~_r_9 or _r_63_206
constraint_190:~_r_9 or _r_63_209
constraint_191:~_r_9 or _r_55
constraint_192:~_r_9 or _r_63_101
constraint_193:~_r_8 or ~_r_9 or _r_63_219
constraint_194:~_r_9 or _r_63_91
constraint_195:~_r_9 or _r_63_97
constraint_196:~_r_9 or _r_63_92
constraint_197:~_r_9 or _r_63_93
constraint_198:~_r_8 or ~_r_9 or _r_63_99
constraint_199:~_r_7 or ~_r_8 or ~_r_9 or _r_63_99
constraint_200:~_r_9 or _r_63_94
constraint_201:~_r_8 or ~_r_9 or _r_63_213
constraint_202:~_r_9 or _r_63_216
constraint_203:~_r_7 or ~_r_9 or _r_63_222
constraint_204:~_r_8 or ~_r_9 or _r_63_221
constraint_205:~_r_7 or ~_r_9 or _r_63_227
constraint_206:~_r_9 or _r_63_174
constraint_207:~_r_7 or ~_r_9 or _r_63_233
constraint_208:~_r_8 or ~_r_9 or _r_63_224
constraint_209:~_r_9 or _r_63_211
constraint_210:~_r_9 or _r_63_87
constraint_211:~_r_9 or _r_63_287_288
constraint_212:~_r_8 or ~_r_22 or _r_23
constraint_213:~_r_63_115 or _r_63_250
constraint_214:~_r_63_152 or _r_63_252
constraint_215:~_r_63_253_265 or _r_63_253_266
constraint_216:~_r_63_274_282_285 or _r_63_287_291
constraint_217:~_r_63_250_360 or _r_63_250_359
constraint_218:~_r_63_250_363 or _r_63_250_362
```

```
constraint_219:~_r_63_250_363 or _r_63_146
constraint_220:_r_63_250_364 or ~_r_63_250_363 or ~_r_63_78
constraint_221:~_r_63_250_364 or _r_63_250_363
constraint_222:~_r_63_250_364 or _r_63_78
constraint_223:~_r_63_250_365 or _r_63_250_364
constraint_224:~_r_63_250_366 or _r_63_242_244 or _r_63_242_243
constraint_225:_r_63_250_366 or ~_r_63_242_244
constraint_226:_r_63_250_366 or ~_r_63_242_243
constraint_227:~_r_63_250_366 or _r_63_250_363
constraint_228:_r_63_250_366 or ~_r_63_250_363
constraint_229:~_r_63_250_367 or _r_63_250_362
constraint_230:_r_63_250_368 or ~_r_63_312_313 or ~_r_63_250
constraint_231:~_r_63_250_368 or _r_63_312_313
constraint_232:~_r_63_250_368 or _r_63_250
constraint_233:~_r_63_250_369 or _r_345
constraint_234:~_r_63_250_370 or _r_63_250_369
constraint_235:~_r_63_251_373 or _r_63_79 or _r_63_81
constraint_236:_r_63_251_373 or ~_r_63_79
constraint_237:_r_63_251_373 or ~_r_63_81
constraint_238:~_r_344_379 or _r_344_378
constraint_239:~_r_344_382 or _r_344
constraint_240:~_r_344_382 or _r_63_160
constraint_241:~_r_344_383 or _r_344_382
constraint_242:~_r_344_384 or _r_344_382
constraint_243:~_r_344_385 or _r_344_382
constraint_244:~_r_344_386 or _r_344
constraint_245:~_r_344_387 or _r_344_382
constraint_246:~_r_344_393 or _r_344_386
constraint_247:~_r_345_396 or ~_r_6
constraint_248:~_r_345_398 or _r_345_397
constraint_249:~_r_345_402_405 or _r_345_402_404
constraint_250:~_r_345_402_406 or _r_345_402_404
constraint_251:~_r_345_402_407 or _r_345_402_404
constraint_252:~_r_345_402_408 or _r_345_402_404
constraint_253:~_r_345_402_409 or _r_345_402_404
constraint_254:~_r_345_402_410 or _r_345_402_404
constraint_255:_r_345_402_418 or ~_r_345_402_403 or ~_r_345_402_417
constraint_256:~_r_345_402_418 or _r_345_402_403
constraint_257:~_r_345_402_418 or _r_345_402_417
constraint_258:_r_345_402_419 or ~_r_345_402_403 or ~_r_345_402_417
constraint_259:~_r_345_402_419 or _r_345_402_403
constraint_260:~_r_345_402_419 or _r_345_402_417
constraint_261:_r_345_402_420 or ~_r_345_402_403 or ~_r_345_402_417
constraint_262:~_r_345_402_420 or _r_345_402_403
constraint_263:~_r_345_402_420 or _r_345_402_417
constraint_264:_r_345_402_421 or ~_r_345_402_403 or ~_r_345_402_417
constraint_265:~_r_345_402_421 or _r_345_402_403
constraint_266:~_r_345_402_421 or _r_345_402_417
constraint_267:~_r_345_402_422 or _r_345_402_403
constraint_268:~_r_345_402_423 or _r_345_402_422
constraint_269:~_r_345_402_427 or _r_345_402_403
constraint_270:~_r_345_402_430 or _r_345_401
constraint_271:~_r_63_250_371_431 or _r_63_250_371
```

```
constraint_272:~_r_63_250_371_432 or _r_63_250_371_431
constraint_273:~_r_63_250_371_433 or _r_63_250_371
</constraints>
</feature_model>
```

D. Parser del archivo KConfig

En esta sección se muestra el archivo encargado de interpretar la gramática del lenguaje Kconfig, el cual posee extensión .g4 y es utilizado como entrada de la herramienta ANTLR4. Se puede observar en el mismo como se reconocen las componentes del lenguaje Kconfig mediante expresiones regulares.

```
grammar parserKConfig;
inicial:
        (config_stmts
        | menuconfig_menu_entry
        | choice_menu_entry
        | comment_menu_entry
        | menu_menu_entry
        | if_menu_entry
        )+;

config_stmts :
        config_menu_entry+
        ;
config_menu_entry
    : CONFIG SYMBOL NEWLINE (config_options NEWLINE*?)+
    ;
config_options
    : config_option
    ;
config_option
    : type WORD_QUOTE? if_expr? ';'?          #config_op_type
    | DEPENDS ON expr    /*if_expr?*/          #config_op_depends
    | PROMPT   WORD_QUOTE if_expr?                  #config_op_prompt
    | SELECT   SYMBOL    if_expr?                   #config_op_select
    | DEFAULT  expr if_expr?                        #config_op_default
    | RANGE    SYMBOL SYMBOL if_expr?               #config_op_range
    | OPTION   misc_option         if_expr?         #config_op_option
    | OPTIONAL       if_expr?                             #config_op_optional
    ;
misc_option
    : ENV '=' WORD_QUOTE
    | DEFCONFIG_LIST
    | ALLNOCONFIG_Y
    | MODULES
    ;
```

```
menuconfig_menu_entry
   : MENUCONFIG SYMBOL  (NEWLINE menuconfig_options)+ NEWLINE?
   ;

menuconfig_options
   : type WORD_QUOTE? if_expr? ';'?          #menuconfig_op_type
   | DEPENDS ON expr    /*if_expr?*/         #menuconfig_op_depends
   | SELECT   SYMBOL    if_expr?            #menuconfig_op_select
   | DEFAULT  expr if_expr?                 #menuconfig_op_default
   | PROMPT   WORD_QUOTE if_expr?           #menuconfig_op_prompt
   | OPTION MODULES                         #menuconfig_op_modules
   ;

choice_menu_entry
     : CHOICE SYMBOL? NEWLINE choice_options+ NEWLINE*? choice_block
NEWLINE*? choice_menu_entry*? ENDCHOICE NEWLINE?
   ;

choice_options
        : choice_option NEWLINE
        ;
choice_option
        : type_choice WORD_QUOTE?  ';'?          #choice_op_type
        | PROMPT   WORD_QUOTE     if_expr?       #choice_op_prompt
        | DEFAULT expr if_expr?                  #choice_op_default
        | DEPENDS ON expr                        #choice_op_depends
        | OPTIONAL                               #choice_op_optional
        ;
choice_block
        : config_menu_entry+
        ;

comment_menu_entry
   : COMMENT WORD_QUOTE NEWLINE comment_options*?
   ;
comment_options
        : comment_option NEWLINE
        ;
comment_option: DEPENDS ON expr
        ;
/*prompt_entry:
        PROMPT   (SYMBOL | WORD_QUOTE)
        ;
*/
```

```
menu_menu_entry
        : MENU WORD_QUOTE NEWLINE (visible NEWLINE)? menu_options?
inicial+ NEWLINE* ENDMENU NEWLINE?
    ;
menu_options
        : menu_option NEWLINE
        ;
menu_option: DEPENDS ON expr
        ;
menu_block:
        'TODO'
        ;
if_menu_entry
        : IF ifmenu_expr NEWLINE inicial+ NEWLINE* ENDIF NEWLINE?
        ;

/*source_menu_entry
   : SOURCE prompt_entry
    ;
*/
/**
Expressions
**/
ifmenu_expr

        :SYMBOL                                    #if_symbol
        | expr                                     #if_menu_expr
        ;

if_expr
    : IF expr
    ;

expr
    : or_expr expr?
    ;

or_expr
    : and_expr ('||' and_expr)*
    ;

and_expr
    : unary_expr ('&&' unary_expr)*
    ;
```

```
unary_expr
    : '!' unary_expr
    | equals_expr
    ;

equals_expr
    : primary (('='|'!=') primary)?
    ;


type        :  TRISTATE | BOOL | BOOLEAN | INT | STRING | HEX;
type_choice: TRISTATE | BOOL;

visible: VISIBLE if_expr;
/**
FIXME
**/
primary
    : SYMBOL
    | WORD_QUOTE
    | '(' expr ')'
    ;

CONFIG      : 'config';
MENUCONFIG      : 'menuconfig';
CHOICE          : 'choice';
ENDCHOICE       : 'endchoice';
COMMENT         : 'comment';
MENU            : 'menu';
ENDMENU         : 'endmenu';
MAINMENU : 'mainmenu';
SOURCE          : 'source';
DEPENDS     : 'depends';
ON          : 'on';
PROMPT      : 'prompt';
OPTION      : 'option';
SELECT      : 'select';
DEFAULT     : 'default' | 'def_bool' | 'def_tristate';
RANGE       : 'range';
OPTIONAL    : 'optional';
IF          : 'if';
ENDIF       : 'endif';
ENV         : 'env';
DEFCONFIG_LIST: 'defconfig_list';
```

```
TRISTATE    : 'tristate' ;
BOOL        : 'bool' ;
BOOLEAN             : 'boolean';
INT         : 'int';
STRING              : 'string';
HEX         : 'hex';
ALLNOCONFIG_Y: 'allnoconfig_y';
MODULES: 'modules';
VISIBLE: 'visible';

Reserved_words    : CONFIG | MENUCONFIG
                  | DEPENDS | ON
                  | DEFAULT | RANGE | OPTIONAL
                  | PROMPT | OPTION | SELECT
                  | IF | ENDIF | ENV
                  | DEFCONFIG_LIST
                  | CHOICE | ENDCHOICE
                  | COMMENT
                  | MENU | ENDMENU
                  | SOURCE
                  | MAINMENU
                  | TRISTATE | BOOL | BOOLEAN | INT | STRING | HEX
          ;

SYMBOL
    :
      '-' ('0'..'9')+
    | ('A'..'Z'|'a'..'z'|'0'..'9'|'_'|'/'|'.')+
//    | ('0' ('x'|'X'))
//        |'0' ('x' | 'X') ('0'..'9' | 'a'..'f' | 'A'..'F')+
    ;
WORD_QUOTE
    : '\'' (ESC|','|':'|.)*? '\''
    | '"' (ESC|','|':'|.)*? '"'
    ;


fragment
ESC
    : '\\' ('"'|'\'')
    ;
SLASH
    : '\\' ;
```

```
DASH: '-'      ;

DASHES
    : '-'
    ;

fragment
HELP_WS : (' '|'\t')*;

words : (SYMBOL | Reserved_words)+
          ;


NEWLINE    : ('\r'? '\n')+ ;

WS : (('\t'|' ')+|'#'.*?'\n') -> skip ; // skip spaces
```