Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

COMPRENDER LOS REQUISITOS

Sem - 5

Ing. Ena Mirella Cacho Chávez

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Logro de la unidad

Al concluir la unidad, el estudiante el estudiante describe los procesos de la ingeniería del software y su aplicación en casos de estudio.

Logro de la sesión

Al finalizar la sesión, el estudiante estará en condiciones de:

- Identificar los requisitos de un negocio
- Identificar partes interesadas.
- Definir un caso de uso
- Definir actor.

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Tabla de Contenidos

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

1. EVALUACIÓN DE MATERIAL ASINCRÓNICO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

1.1. Actividad asincrónica

- Antes de la sesión sincrónica, el estudiante debe revisar: Comprender los requisitos (Pressman & Maxim, 2020).

1.2. Evaluación de material asincrónico

- Estimado estudiante,

Luego de revisar el material brindado, responder a la interrogante:

¿Qué es un caso de uso?

1.2. Evaluación de material asincrónico

- Un **caso de uso** cuenta una **historia** estilizada sobre cómo un usuario final (que desempeña uno de varios papeles posibles) **interactúa con el sistema** en un conjunto específico de circunstancias.
- La historia puede ser un texto narrativo (una historia de usuario), un esquema de tareas o interacciones, una descripción basada en plantillas o una representación diagramática. Independientemente de su forma, un caso de uso **describe el software o el sistema desde el punto de vista del usuario final.**

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. DESARROLLO DE CONTENIDOS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

2.1. Ingeniería de requisitos

Visualizar un corto video que lo puedes encontrar en “Material Complementario”

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Es tu peor pesadilla

(Material complementario)

conker

REFUERZO MI APRENDIZAJE

Ingresa al enlace para resolver 5 preguntas referente a lo leído

https://www.joinconker.com/fyuqba

2.1. Ingeniería de requisitos

- Diseñar y **crear programas** informáticos es todo un **reto, creativo y divertido**. De hecho, la creación de software es tan atractiva que muchos desarrolladores **quieren lanzarse** a **ello antes de tener** una **idea clara** de lo que se necesita.

2.1. Ingeniería de requisitos

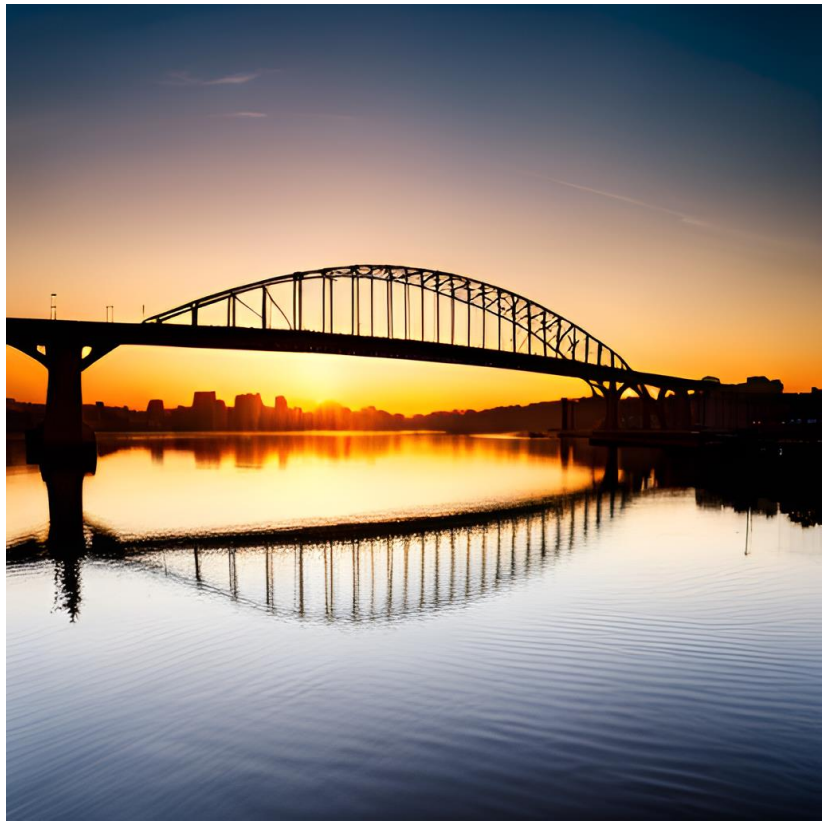
- Argumentan que las **cosas se aclararán** a medida que **construyan,** que las partes interesadas en el proyecto sólo podrán comprender las necesidades tras examinar las **primeras iteraciones del software**, que **las cosas cambian** tan rápidamente que cualquier intento de **comprender los requisitos** en detalle es una **pérdida de tiempo**, que lo **fundamental es producir un programa** que funcione y que todo **lo demás es secundario**.
- Lo que hace seductores a estos argumentos es que contienen elementos de verdad. Pero cada argumento es erróneo y puede conducir al fracaso de un proyecto de software.

2.1. Ingeniería de requisitos

- Ingeniería de requisitos
  - Término que designa el amplio espectro de tareas y técnicas que conducen a la comprensión de los requisitos.
  - **Desde el punto de vista del proceso de software**, la ingeniería de requisitos es una importante acción de ingeniería de software que comienza durante la **actividad de comunicación** y continúa en la actividad de **modelado.**
  - La ingeniería de requisitos establece una **base sólida** para el **diseño y la construcción**. Sin ella, es muy probable que el software resultante no satisfaga las necesidades del cliente.
  - Debe **adaptarse** a las **necesidades** del **proceso, el proyecto, el producto y las personas** que realizan el trabajo.
  - Es importante darse cuenta de que cada una de estas tareas se realiza de **forma iterativa** a medida que el **equipo del proyecto y las partes interesadas** siguen **compartiendo información sobre sus respectivas preocupaciones.**

2.1. Ingeniería de requisitos

- La ingeniería de requisitos tiende un **puente entre el diseño y la construcción**.
- Pero ¿dónde se origina el puente? Se podría argumentar que **comienza** con los **interesados en el proyecto** (por ejemplo, directores, clientes y usuarios finales), **donde:**
  - Se **definen las necesidades empresariales**
  - Se describen los **escenarios** de usuario
  - Se delinean las **funciones y características** y
  - Se identifican las **limitaciones** del proyecto.
- Otros sugieren que empiece con una **definición** más amplia del **sistema**, donde el **software** no es más que un **componente de un sistema más amplio.**



2.1. Ingeniería de requisitos

- Pero sea cual sea el punto de partida, el **viaje a través del puente** nos lleva muy **por encima del proyecto**, **permitiéndonos examinar:**
  - El **contexto** del trabajo de software que se va a realizar
  - Las **necesidades** específicas que el diseño y la construcción deben abordar
  - Las **prioridades** que guían el orden en el que se va a completar el trabajo; y
  - La **información, funciones y comportamientos** que tendrán un profundo impacto en el diseño resultante.

2.1. Ingeniería de requisitos

- La ingeniería de requisitos abarca **siete tareas** con límites a veces confusos:
  1. Inicio
  2. Obtención
  3. Elaboración
  4. Negociación
  5. Especificación
  6. Validación y
  7. Gestión.
- Es importante señalar que algunas de estas tareas se realizan **en paralelo** y que todas se **adaptan a las necesidades del proyecto**. Es de esperar que se haga un poco de **diseño durante el trabajo de requisitos** y un poco de trabajo de **requisitos durante el diseño.**

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

1. Inicio

En general, la mayoría de los proyectos comienzan con una **necesidad empresarial** identificada o cuando se descubre un **nuevo mercado** o **servicio** potencial.

Al inicio del proyecto, se establece una **comprensión básica** del **problema**, las **personas que desean una solución** y la **naturaleza de la solución** deseada.

La **comunicación** entre todas las **partes interesadas** y **el equipo** de software debe establecerse durante esta tarea para iniciar una colaboración eficaz.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. Consulta

Parece bastante sencillo: preguntar al cliente, a los usuarios y a otras personas **cuáles son los objetivos del sistema** o producto, qué se quiere conseguir, **cómo encaja el sistema** o producto en las necesidades de la empresa y, por último, **cómo se va a utilizar el sistema** o producto e n el día a día. Pero no es sencillo, **es muy difícil.**

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. Consulta

- Una parte importante de la elicitación es comprender los **objetivos empresariales**.
- **Un objetivo** es una **meta a largo plazo** que debe **alcanzar un sistema** o producto.
- Los objetivos pueden ser:
  - Funcionales
  - No funcionales (por ejemplo, fiabilidad, seguridad, usabilidad).
- Los objetivos suelen ser una **buena forma** de **explicar los requisitos** a las partes interesadas y, una vez establecidos, pueden utilizarse para **gestionar los conflictos** entre ellas.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. Consulta

- Los objetivos deben especificarse **con precisión** y servir de **base** para la elaboración, verificación y validación de los requisitos, la gestión de conflictos, la negociación, la explicación y la evolución.
- Tu trabajo consiste en **implicar a las partes interesadas** y animarlas a que **compartan sus objetivos** con sinceridad.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. Consulta

- Una vez **captados los objetivos**, se establece un mecanismo de **priorización** y se crea una **justificación de diseño** para una arquitectura potencial (que cumpla los objetivos de las partes interesadas).
- La **agilidad** es un aspecto **importante** de la ingeniería de requisitos. El objetivo de la obtención es transferir las ideas de las partes interesadas al equipo de software sin problemas ni demoras. Es muy **probable** que sigan **surgiendo nuevos requisitos** a medida que se produce el **desarrollo iterativo** del producto.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **3. Elaboración**
  - La tarea de elaboración se centra en desarrollar un **modelo de requisitos** refinado que identifique varios aspectos de la **función**, el **comportamiento** y la **información** del software.
  - La elaboración se basa en la **creación y el perfeccionamiento** de **escenarios** de usuario obtenidos durante la elicitación.
  - Estos **escenarios** describen **cómo** los **usuarios finales** (y otros actores) **interactuarán** con el **sistema**. Cada escenario de usuario se analiza para extraer **clases de análisis,** entidades de dominio visibles para el usuario final.
  - Se definen los **atributos** de cada clase de análisis y se **identifican los servicios** que requiere cada clase.
  - Se **identifican las relaciones** y la **colaboración entre clases**.
  - Elaborar es bueno, pero hay que saber cuándo parar. **La clave** está en **describir el problema** de forma que se establezca una **base firme para el diseño** y luego seguir adelante. No hay que obsesionarse con detalles innecesarios.

2.1. Ingeniería de requisitos

- **4. Negociación**
- No es raro que los **clientes y usuarios pidan más** de lo que se puede conseguir, dados los limitados recursos de la empresa.
- También es relativamente habitual que distintos clientes o usuarios **propongan** requisitos contradictorios, argumentando que su versión es "esencial para nuestras necesidades especiales."
- Estos conflictos deben conciliarse mediante un proceso de negociación.

2.1. Ingeniería de requisitos

- **4. Negociación**
- **Se pide a los clientes**, usuarios y otras partes interesadas que **clasifiquen los requisitos** y discutan los conflictos por orden de **prioridad.**
- En una negociación eficaz **no debe haber vencedores ni vencidos.** Ambas partes ganan, porque se consolida un "acuerdo" con el que ambos pueden vivir.
- Debe utilizar un **enfoque iterativo** que priorice los requisitos**, evalúe su coste y riesgo** y **aborde los conflictos** internos.
- De este modo, los **requisitos se eliminan**, se **combinan** o se **modifican** para que todas las partes queden satisfechas.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **5. Especificación**
- Una especificación puede ser un **documento escrito**, un **conjunto de modelos** gráficos, un **modelo matemático formal**, una **colección de escenarios** de uso, un **prototipo** o cualquier combinación de ellos.
- **Algunos sugieren** que se desarrolle y utilice una "**plantilla estándar**“ para una especificación, argumentando que así se consiguen requisitos presentados de forma coherente y, por tanto, más comprensible.
- Sin embargo, **a veces** es necesario **ser flexible** a la hora de desarrollar una especificación. La formalidad y el formato de una especificación varían en función del tamaño y la complejidad del software que se va a construir.
- **Para sistemas grandes**, un **documento escrito**, que combine descripciones en lenguaje natural y modelos gráficos, puede ser el mejor enfoque. **(Plantilla de especificación de requisitos en carpeta material complementario).**
- Sin embargo, los **escenarios de uso** pueden ser todo lo que se necesita para productos o **sistemas más pequeños** que residen en entornos técnicos bien comprendidos.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Validación**
- La **calidad** de los productos elaborados durante la ingeniería de requisitos se evalúa en una fase de validación.
- Una de las principales preocupaciones durante la validación de requisitos es la **coherencia.**
- Utilice el **modelo de análisis** para asegurarse de que los requisitos se han establecido de forma coherente.
- La validación de requisitos examina la especificación para garantizar que todos los requisitos de software se han establecido **sin ambigüedades**; que se han detectado y **corregido** las **incoherencias, omisiones y errores**; y que los productos de trabajo se **ajustan a las normas** establecidas para el proceso, el proyecto y el producto.

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Validación**
- El principal mecanismo de validación de requisitos es la **revisión técnica**.
- El **equipo de revisión** que **valida** los **requisitos** incluye a **ingenieros de software, clientes, usuarios y otras partes interesadas** que **examinan** la especificación **en busca de errores** en contenido o interpretación, **áreas que requieren aclaraciones**, **falta de información**, incoherencias (un problema importante cuando se diseñan productos o sistemas de gran tamaño), requisitos contradictorios o requisitos **poco realistas** (inalcanzables).

2.1. Ingeniería de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Validación**
- Para ilustrar algunos de los problemas que surgen durante la validación de requisitos, consideremos **dos requisitos** aparentemente **inocuos**:
  - El software debe ser fácil de usar.
  - La probabilidad de éxito de una intrusión no autorizada en una base de datos debe ser inferior a 0,0001.
- El **primer requisito** es **demasiado vago** para que los desarrolladores puedan probarlo o evaluarlo. ¿Qué significa exactamente "fácil de usar"? Para validarlo, hay que cuantificarlo o **calificarlo de alguna manera**.
- El **segundo requisito** tiene un **elemento cuantitativo** ("menos de 0,0001"), pero las **pruebas de intrusión serán difíciles** y **llevarán mucho tiempo**. ¿Está justificado este nivel de seguridad para la aplicación? ¿Pueden otros requisitos complementarios asociados a la seguridad (por ejemplo, protección por contraseña, handshaking especializado) sustituir al requisito cuantitativo señalado?

2.1. Ingeniería de requisitos

- **7. Gestión de requisitos**
- Los **requisitos** de los sistemas informáticos **cambian**, y el deseo de cambiarlos persiste **durante toda la vida del sistema.**
- La gestión de requisitos
  - Es un **conjunto de actividades** que ayudan al equipo del proyecto **a identificar, controlar y realizar un seguimiento de los requisitos** y los **cambios** en los requisitos en cualquier momento a medida que avanza el proyecto.
  - Muchas de estas actividades son idénticas a las **técnicas de gestión de la configuración** del software (SCM).

2.1. Ingeniería de requisitos

- Lista de comprobación para la validación de requisitos
  - (Ver en material complementario)

2.2. Estableciendo el trabajo preliminar

- En un **entorno ideal**,
  - Las **partes interesadas** y los **ingenieros** de software trabajan juntos en el mismo equipo.
  - En tales casos, la **ingeniería de requisitos** es simplemente una cuestión de mantener **conversaciones mezquinas** con colegas que son **miembros conocidos del equipo.**
- Pero la **realidad** suele ser muy distinta.
  - Los **clientes o usuarios** finales pueden residir en ciudades o países diferentes, tener una **vaga idea de lo que se necesita**, tener **opiniones contradictorias** sobre el sistema que se va a construir, tener **conocimientos técnicos limitados** y disponer de **poco tiempo para interactuar** con el ingeniero de requisitos.
  - **Ninguna de estas cosas es deseable**, pero todas **son comunes** , y a menudo se ve obligado a trabajar dentro de las limitaciones impuestas por esta situación.

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

1. Identificación de las partes interesadas
2. Reconocer múltiples puntos de vista
3. Hacia la colaboración
4. Formular las primeras preguntas
5. Requisitos no funcionales
6. Trazabilidad

2.2. Estableciendo el trabajo preliminar

- **1. Identificación de las partes interesadas**
- Una **parte interesada**
  - “Cualquiera que se beneficie de forma directa o indirecta del sistema que se está desarrollando".
  - Ya hemos identificado a los sospechosos habituales:
    - Directores de operaciones empresariales
    - directores de producto
    - responsables de marketing
    - clientes internos y externos
    - usuarios finales
    - Consultores
    - ingenieros de producto
    - ingenieros de software
    - ingenieros de soporte y mantenimiento, y otros.

2.2. Estableciendo el trabajo preliminar

- **1. Identificación de las partes interesadas**
- Cada parte interesada tiene:
  - Una **visión diferente del sistema**
  - Obtiene **distintos beneficios** cuando el sistema se desarrolla con éxito y
  - Está abierta a **distintos riesgos** si el esfuerzo de desarrollo fracasa.

- Al **principio**, debe crear una **lista de personas** que **contribuirán** a medida que se obtengan los **requisitos.**
- La **lista** inicial **crecerá** a medida que **se contacte con las partes interesadas,** ya que se preguntará a cada una de ellas: **"¿Con quién más crees que debería hablar?".**

2.2. Estableciendo el trabajo preliminar

- **2. Reconocer múltiples puntos de vista**
- Dado que existen muchas partes interesadas diferentes, los requisitos del sistema se estudiarán desde muchos puntos de vista distintos.
- Por ejemplo,
  - El grupo de **marketing** está interesado en **características que entusiasmen al mercado potencial** y **faciliten la venta** del nuevo sistema.
  - A los **directores** de empresa les interesa un conjunto de **funciones** que pueda construirse **dentro del presupuesto** y que esté listo para **cumplir las ventanas de mercado definidas**.
  - Los **usuarios finales** quieren **funciones** que les resulten **familiares y fáciles** de aprender y utilizar.
  - A los **ingenieros de software** pueden interesarles **funciones** que sean **invisibles** para los **interesados no técnicos**, pero que permitan una infraestructura que soporte funciones y características más comercializables.

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **2. Reconocer múltiples puntos de vista**
- Los **ingenieros de soporte** pueden centrarse en la capacidad de **mantenimiento** del software.
- Cada uno de estos grupos (y otros) aportará información al proceso de ingeniería de requisitos.
- A medida que se **recopila información** desde **múltiples puntos de vista**, los requisitos que surjan pueden ser incoherentes o **entrar en conflicto** entre sí.
- Deberá **clasificar toda la información** de las partes interesadas (incluidos los requisitos incoherentes y conflictivos) de forma que los **responsables de la toma de decisiones puedan elegir** un conjunto de requisitos internamente coherente para el sistema.

2.2. Estableciendo el trabajo preliminar

- **2. Reconocer múltiples puntos de vista**
- Hay varios **factores que pueden dificultar** la obtención de requisitos para un software que satisfaga a sus usuarios:
  - Los objetivos del proyecto no están claros
  - Las prioridades de las partes interesadas difieren
  - La gente tiene suposiciones tácitas
  - Las partes interesadas interpretan los significados de forma diferente y
  - Los requisitos se establecen de una forma que dificulta su verificación.
- El objetivo de una ingeniería de requisitos eficaz es **eliminar o, al menos, reducir estos problemas.**

2.2. Estableciendo el trabajo preliminar

- **3. Hacia la colaboración**
- Si en un proyecto de software participan **cinco partes interesadas**, es posible que haya **cinco (o más) opiniones diferentes** sobre el conjunto adecuado de requisitos.
- Los **clientes** (y otras partes interesadas) deben colaborar **entre sí** (evitando batallas territoriales mezquinas) y con los **profesionales de la ingeniería de software** si se quiere obtener un sistema de éxito.

2.2. Estableciendo el trabajo preliminar

- **3. Hacia la colaboración**
- Pero, ¿cómo se consigue esta colaboración?
- El trabajo de un **ingeniero de requisitos** consiste en:
  - **Identificar** las **áreas comunes** (es decir, los requisitos en los que todas las partes interesadas están de acuerdo) y
  - Las **áreas de conflicto** o incoherencia (es decir, los requisitos que desea una parte interesada pero que entran en conflicto con las necesidades de otra parte interesada).
- Por supuesto, es esta **última categoría** la que plantea un reto. La colaboración no significa necesariamente que los requisitos sean "definidos por el comité". En muchos casos, las partes interesadas **colaboran aportando su punto de vista** sobre los requisitos, pero un "**campeón del proyecto**" fuerte (por ejemplo, un director comercial o un tecnólogo de alto nivel) **puede tomar la decisión final** sobre qué requisitos son los más importantes.

2.2. Estableciendo el trabajo preliminar

- Utilizar el "Póquer de planificación"
- **(Ver en material complementario)**

REFUERZO MI APRENDIZAJE

Ingresa al enlace para resolver 5 preguntas referente a lo leído

https://www.joinconker.com/kajagn

conker

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **4. Formular las primeras preguntas**
- Las preguntas formuladas al inicio del proyecto deben estar "libres de contexto". El primer grupo de preguntas libres de contexto **se centra** en el **cliente** y otras **partes interesadas**, así como en los **objetivos** y **beneficios** generales del proyecto.
- Por ejemplo, se podría preguntar
  - ¿Quién está detrás de la petición de esta obra?
  - ¿Quién utilizará la solución?
  - ¿Cuál será el beneficio económico de una solución exitosa?
  - ¿Hay otra fuente para la solución que necesita?
- Estas preguntas **ayudan a identificar** a todas las **partes interesadas** en el software que se va a crear. Además, las preguntas identifican los **beneficios cuantificables** de una implantación satisfactoria y las **posibles alternativas** al desarrollo de **software a medida.**

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **4. Formular las primeras preguntas**
- Las siguientes preguntas le permite **comprender mejor el problema** y permite al cliente expresar sus **percepciones** sobre una **solución**:
  - ¿Cómo caracterizaría el "buen" resultado que generaría una solución satisfactoria?
  - ¿Qué problemas abordará esta solución?
  - ¿Puede mostrarme (o describirme) el entorno empresarial en el que se utilizará la solución?
  - ¿Afectarán los problemas o limitaciones especiales de rendimiento al planteamiento de la solución?

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **4. Formular las primeras preguntas**
- El último grupo de preguntas **se centra** en la **eficacia** de la propia actividad de **comunicación**.
- Gause y Weinberg las denominan "**metapreguntas**" y proponen la siguiente lista (abreviada):
  - ¿Es usted la persona adecuada para responder a estas preguntas? ¿Son "oficiales" sus respuestas?
  - ¿Son mis preguntas pertinentes para el problema que tiene?
  - ¿Hago demasiadas preguntas?
  - ¿Alguien más puede aportar información adicional?
  - ¿Debería preguntarte algo más?

- Estas preguntas (y otras) ayudarán a "**romper el hielo**" e iniciar la comunicación, esencial para el éxito de la encuesta.
- Pero el formato de reunión de preguntas y respuestas no es un enfoque que haya tenido un éxito abrumador. De hecho, la **sesión de preguntas y respuestas** debería utilizarse sólo en el **primer encuentro** y, a continuación, **sustituirse** por un **formato de obtención de requisitos** que combine elementos de resolución de problemas, negociación y especificación.

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **5. Requisitos no funcionales**
- Un requisito no funcional (NFR) puede describirse como un **atributo de calidad**, un **atributo de rendimiento**, un **atributo de seguridad** o una **restricción general** de un sistema.
- A menudo no son fáciles de articular para las partes interesadas. Chung sugiere que se pone un énfasis desproporcionado en la funcionalidad del software, pero que éste puede no ser útil o utilizable sin las características no funcionales necesarias.
- Es posible definir un enfoque en dos fases que puede ayudar a un equipo de software y a otras partes interesadas a identificar requisitos no funcionales.
  - Durante la **primera fase**, se establece un conjunto de **directrices de ingeniería** de software para el sistema que se va a construir. Estas incluyen directrices sobre **buenas prácticas**, pero también abordan el **estilo arquitectónico** y el uso de **patrones de diseño**.

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **5. Requisitos no funcionales**
- A continuación, se elabora una lista de NFR (por ejemplo, requisitos de **usabilidad**, **comprobabilidad**, **seguridad** o **mantenimiento**). **Una tabla** sencilla lista los **NFR** como etiquetas de **columna** y las **directrices de ingeniería** de software como etiquetas de **fila**. Una matriz de relaciones compara cada directriz con todas las demás, ayudando al equipo a evaluar si cada par de directrices es complementario, se solapa, está en conflicto o es independiente.
- En la **segunda fase**, el equipo **prioriza cada requisito no funcional** creando un conjunto homogéneo de requisitos no funcionales mediante una serie de reglas de decisión que establecen qué directrices aplicar y cuáles rechazar.

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Trazabilidad**
- La trazabilidad es un término de ingeniería de software que hace referencia a los **vínculos documentados** entre los **productos de trabajo** de ingeniería de software (por ejemplo, requisitos y casos de prueba).
- Una **matriz de trazabilidad** permite a un ingeniero de requisitos **representar la relación entre los requisitos y otros productos** de trabajo de ingeniería de software.
- **Filas** de la trazabilidad se etiquetan con **nombres de requisitos**, y las **columnas** pueden etiquetarse con el **nombre de un producto** de trabajo de ingeniería de software (por ejemplo, un elemento de diseño o un caso de prueba).

2.2. Estableciendo el trabajo preliminar

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Trazabilidad**
- Una **celda** de la matriz **se marca** para indicar la presencia de un **vínculo entre ambos**.
- Las matrices de trazabilidad pueden:
  - Servir de apoyo a diversas actividades de desarrollo de ingeniería.
  - Proporcionar continuidad a los desarrolladores cuando un proyecto pasa de una fase a otra, independientemente del modelo de proceso que se utilice.
  - Utilizarse a menudo para garantizar que los productos del trabajo de ingeniería han tenido en cuenta todos los requisitos.
  - A medida que crece el número de requisitos y de productos de trabajo, resulta cada vez más difícil mantener actualizada la matriz de trazabilidad. No obstante, es importante crear algún medio para seguir el impacto y la evolución de los requisitos del producto.

2.3. Recopilación de requisitos

- La recopilación de requisitos **combina** elementos de **resolución de problemas, elaboración, negociación y especificación**.
- Para fomentar un enfoque colaborativo y orientado al equipo en la recopilación de requisitos, las **partes interesadas trabajan juntas** para **identificar el problema**, proponer elementos de la **solución, negociar diferentes enfoques** y especificar un conjunto preliminar de **requisitos de la solución.**

2.3. Recopilación de requisitos

1. Recopilación colaborativa de requisitos
2. Escenarios de uso
3. Productos de la consulta

2.3. Recopilación de requisitos

- **1. Recopilación colaborativa de requisitos**
- Se han propuesto **muchos enfoques diferentes** para la recopilación colaborativa de requisitos. Cada uno utiliza un escenario ligeramente diferente, pero todos aplican alguna variación de las siguientes directrices básicas:
- Se celebran reuniones (reales o virtuales) a las que asisten tanto ingenieros de software como otras partes interesadas.
- Se establecen normas de preparación y participación.
- Se sugiere un orden del día lo suficientemente formal como para cubrir todos los puntos importantes, pero lo suficientemente informal como para fomentar la libre circulación de ideas.
- · Un "facilitador" (puede ser un cliente, un desarrollador o una persona ajena a la empresa) controla la reunión.
- Se utiliza un "mecanismo de definición" (pueden ser hojas de trabajo, rotafolios o adhesivos murales o un tablón de anuncios electrónico, una sala de chat o un foro virtual).

2.3. Recopilación de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **1. Recopilación colaborativa de requisitos**
- El objetivo es **identificar el problema**, proponer **elementos de la solución**, **negociar** distintos enfoques y especificar un conjunto **preliminar de requisitos** de la solución.
- Durante la **fase inicial**
  - **S**e genera una "**solicitud de producto**" de una o dos páginas.
  - Se selecciona un lugar, una hora y una fecha para la reunión
  - Se elige un moderador y
  - Se invita a participar a los miembros del equipo de software y a otras partes interesadas.

- Si un **sistema** o producto va a **servir a muchos usuarios**, hay que asegurarse de que los **requisitos** se obtienen **de una muestra** representativa de usuarios.
- Si sólo **un usuario** define todos los requisitos, el **riesgo** de aceptación es **alto** (lo que significa que puede haber otras partes interesadas que no acepten el producto). La solicitud de producto se distribuye a todos los asistentes antes de la fecha de la reunión.

2.3. Recopilación de requisitos

- A modo de **ejemplo**, consideremos un extracto de una **solicitud** de producto escrita por una **persona de marketing** que participa en el proyecto **SafeHome.**
- Esta persona escribe lo siguiente sobre la **función de seguridad doméstica** que va a formar parte de Safe Home:
  - Nuestros estudios indican que el mercado de los **sistemas de gestión doméstica** crece a un ritmo del **40% anual**.
  - La primera función de SafeHome que sacamos al mercado debería ser la de **seguridad doméstica.**
  - La mayoría de la gente está familiarizada con los "**sistemas de alarma**", así que sería fácil de vender.

2.3. Recopilación de requisitos

- También podríamos considerar la posibilidad de utilizar el **control por voz** del sistema mediante alguna tecnología como **Alexa.**
- La función de **seguridad del hogar** protegería contra y/o reconocería una variedad de "situaciones" indeseables como **entrada ilegal**, **incendio, inundación, niveles de monóxido de carbono** y otras.
- Utilizará nuestros **sensores inalámbricos** para detectar cada situación
- Podrá ser programada por el propietario y contactará automáticamente con una agencia de vigilancia y con el **teléfono móvil del propietario** cuando se detecte una situación.

2.3. Recopilación de requisitos

- En realidad, **otras personas contribuirían** a esta descripción durante la reunión de requisitos y se dispondría de mucha más información. Pero incluso con información adicional, existe **ambigüedad**, es probable que haya **omisiones** y pueden producirse **errores**. Por ahora, bastará con la "descripción funcional" anterior.

2.3. Recopilación de requisitos

- Durante la revisión de la solicitud de producto en los días previos a la reunión, se pide a **cada asistente** que elabore una **lista de objetos** que forman parte del **entorno que rodea al sistema**, otros **objetos que debe producir** el sistema y **objetos que utiliza el sistema** para realizar sus funciones.
- Además, se pide a **cada asistente** que haga otra **lista de servicios** (procesos o funciones) **que manipulan** o interactúan **con los objetos**.
- Por último, también se elaboran **listas de restricciones** (por ejemplo, coste, tamaño, normas empresariales) y **criterios de rendimiento** (por ejemplo, velocidad, precisión, seguridad).
- Se informa a los asistentes de que no se espera que las **listas** sean exhaustivas, sino que reflejen la **percepción que cada uno tiene** del sistema.

2.3. Recopilación de requisitos

- Los objetos descritos para **SafeHome** podrían incluir:
  - El panel de control
  - Detectores de humo
  - Sensores de puertas y ventanas
  - Detectores de movimiento
  - Una alarma
  - Un evento (se ha activado un sensor)
  - Una pantalla
  - Una tableta
  - Números de teléfono
  - Una llamada telefónica, etc.

2.3. Recopilación de requisitos

- La lista de servicios podría incluir:
  - La configuración del sistema
  - La activación de la alarma
  - La supervisión de los sensores
  - La marcación del teléfono mediante un router inalámbrico
  - La programación del panel de control y la lectura de la pantalla (nótese que los servicios actúan **sobre objetos**).
- De forma similar, cada asistente elaborará **listas de restricciones** (por ejemplo:
  - El sistema debe reconocer cuándo los sensores no están en funcionamiento, debe ser fácil de usar, debe interactuar directamente con una línea telefónica estándar) y criterios de rendimiento (por ejemplo, un evento del sensor debe reconocerse en 1 segundo y debe implementarse un esquema de prioridad de eventos).

2.3. Recopilación de requisitos

- Las **listas de objetos** pueden fijarse a las paredes de la sala con grandes hojas de papel, *pegarse a las paredes* con hojas adhesivas o escribirse en un tablón.
- Alternativamente, las listas pueden haberse publicado en un foro de grupo o en un sitio web interno o planteado en un entorno de red social para su revisión antes de la reunión.
- **Lo ideal** es que **cada entrada** de la lista pueda **manipularse por separado**, de modo que las listas puedan **combinarse**, puedan **borrarse** entradas y puedan hacerse **adiciones**. En esta fase, la crítica y el debate están estrictamente prohibidos. **Evite el impulso de rechazar la idea de un cliente** por "**demasiado costosa**" o "poco práctica".
- Se trata de **negociar** una **lista aceptable** para todos. Para ello , hay que mantener la mente abierta. Una vez presentadas las listas individuales en un área temática, el grupo crea una lista combinada eliminando las entradas redundantes, añadiendo cualquier idea nueva que surja durante la discusión, pero sin borrar nada.

2.3. Recopilación de requisitos

- Una vez **creadas las listas combinadas** de todos los temas, **se inicia el debate**, coordinado por el moderador.
- La lista combinada se acorta, alarga o reformula para reflejar adecuadamente el producto o sistema que se va a desarrollar.
- El **objetivo** es elaborar una **lista consensuada** de objetos, servicios, limitaciones y prestaciones del sistema que se va a construir.
- En muchos casos, un objeto o servicio descrito en una lista requerirá más explicaciones. Para ello, las partes interesadas elaboran especificaciones para las entradas de las listas o crean un caso de uso relacionado con el objeto o servicio.

2.3. Recopilación de requisitos

- Por ejemplo, las miniespecíficaciones para el objeto SafeHome Panel de control podría ser:
  - El panel de control es una unidad montada en la pared que mide aproximadamente 230 × 130 mm.
  - El panel de control tiene conectividad inalámbrica con sensores y una tableta.
  - La interacción con el usuario se produce a través de un teclado que contiene 12 teclas.
  - Una pantalla OLED en color de 75 × 75 mm proporciona información al usuario.
  - El software proporciona indicaciones interactivas, eco y funciones similares.

2.3. Recopilación de requisitos

- Las **miniespecificaciones se presentan a todas las partes interesadas** para su debate. Se realizan adiciones, supresiones y ampliaciones. En algunos casos, el desarrollo de las miniespecificaciones **descubrirá nuevos objetos, servicios, restricciones o requisitos de rendimiento** que se añadirán a las listas originales.
- Durante todas las discusiones, el equipo puede plantear alguna cuestión que no pueda resolverse durante la reunión. Se mantiene una lista de cuestiones para poder actuar sobre ellas más adelante.

2.3. Recopilación de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Ejemplo de caso práctico

- Realización de una reunión de recopilación de requisitos
- **(Visualizar en material complementario)**

REFUERZO MI APRENDIZAJE

Ingresa al enlace para resolver 5 preguntas referente a lo leído

https://www.joinconker.com/ztbrqv

conker

2.3. Recopilación de requisitos

- **2. Escenarios de uso**
- A medida que se reúnen los requisitos, **empieza a materializarse** una **visión** general de las **funciones y características del sistema**. Sin embargo, es **difícil pasar a actividades de ingeniería de software** más técnicas **hasta que no se entienda** cómo **utilizarán las funciones** los distintos tipos de usuarios finales.
- Para lograrlo, los desarrolladores y usuarios pueden crear un conjunto de **escenarios** que **identifiquen un hilo de uso** para el sistema que se va a construir.
- Los **escenarios**, a menudo denominados **casos de uso**, proporcionan una descripción de **cómo se utilizará el sistema**.

2.3. Recopilación de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Desarrollo de un escenario de usuario preliminar

(Visualizar en material complementario)

REFUERZO MI APRENDIZAJE

Ingresa al enlace para resolver 5 preguntas referente a lo leído

https://www.joinconker.com/jufvyp

conker

2.3. Recopilación de requisitos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **3. Productos de la consulta**
- Los productos de trabajo que se obtienen durante la obtención de requisitos varían en función del tamaño del sistema o producto que se vaya a construir.
- En el caso de sistemas grandes, los productos de trabajo pueden incluir:
  1. Una declaración de necesidad y viabilidad
  2. Una declaración delimitada del alcance del sistema o producto
  3. Una lista de clientes, usuarios y otras partes interesadas que hayan participado en la obtención de requisitos
  4. Una descripción del entorno técnico del sistema
  5. Una lista de requisitos (preferiblemente organizados por funciones) y las restricciones de dominio que se aplican a cada uno; y
  6. Un conjunto de escenarios de uso que permitan comprender el uso del sistema o producto en diferentes condiciones de funcionamiento.
- Cada uno de estos productos de trabajo es revisado por todas las personas que han participado en la obtención de requisitos.

2.4. Desarrollo de casos de uso

- Un **caso de uso**
  - Cuenta una **historia estilizada** sobre cómo un **usuario** final (que desempeña uno de varios papeles posibles) **interactúa con el sistema** en un conjunto específico de circunstancias.
  - **La historia** puede ser.
    - Un **texto narrativo** (una historia de usuario)
    - Un **esquema de tareas** o interacciones
    - Una **descripción basada en plantillas** o
    - Una **representación diagramática**.
  - Independientemente de su forma, un **caso de uso describe el software** o el sistema desde el **punto de vista del usuario final.**

2.4. Desarrollo de casos de uso

- El **primer paso** para **redactar un caso de uso**
  - Es **definir** el conjunto de "**actores**" que intervendrán en la historia.
  - Los **actores** son las distintas **personas** (o **dispositivos**) que **utilizan el sistema** o producto en el contexto de la función y el comportamiento que se van a describir.
  - Los **actores** representan los **papeles** que desempeñan las **personas (o dispositivos**) en el funcionamiento del sistema.
  - Definido de manera algo más formal,
    - Un actor es **todo aquello** que **se comunica con el sistema** o producto y que es **externo al propio sistema**.
    - Cada **actor** tiene **uno o varios objetivos** cuando utiliza el sistema.

2.4. Desarrollo de casos de uso

- Es importante señalar que un **actor** y un **usuario final no son** necesariamente lo **mismo**.
- Un **usuario típico** puede **desempeñar varios papeles** diferentes al utilizar un sistema
- Mientras que un **actor** representa **una clase** de **entidades externas** (a menudo, pero **no siempre, personas**) que desempeñan **un solo papel** en el contexto del caso de uso.
- A modo de **ejemplo**, consideremos un **usuario** que interactúa con el programa que permite experimentar con la configuración de los sensores de alarma en un edificio virtual.

2.4. Desarrollo de casos de uso

- Tras una cuidadosa revisión de los requisitos, el **software** para el ordenador de control **requiere cuatro modos** (roles) diferentes para la interacción: modo de colocación, modo de prueba, modo de monitorización y modo de resolución de problemas.
- Por lo tanto, pueden definirse cuatro actores:
  - Editor
  - Probador
  - Monitor y
  - Solucionador de problemas.
- En **algunos casos**, el **usuario** puede **desempeñar todos los papeles**. **En otros**, **diferentes personas** pueden desempeñar el papel de cada actor .

2.4. Desarrollo de casos de uso

- Dado que la **obtención de requisitos** es una actividad **evolutiva**, **no se identifican todos los actores durante la primera iteración**.
- Es posible identificar **actores primarios** durante la primera iteración y **actores secundarios** a medida que se aprende más sobre el sistema.
  - Los **actores primarios** interactúan para lograr la función requerida del sistema y obtener el beneficio previsto del sistema. **Trabajan directa y frecuentemente con el software**.
  - Los **actores secundarios** apoyan al sistema para que los actores primarios puedan hacer su trabajo.

2.4. Desarrollo de casos de uso

- Una vez **identificados los actores**, se pueden **desarrollar los casos de uso**. Preguntas a las que debe responder un caso de uso:
  - 1. ¿Quién es el actor principal, el actor o actores secundarios?
  - 2. ¿Cuáles son los objetivos del actor?
  - 3. ¿Qué condiciones previas deben existir antes de que comience la historia?
  - 4. ¿Qué tareas o funciones principales desempeña el actor?
  - 5. ¿Qué excepciones podrían considerarse a medida que se describe la historia?
  - 6. ¿Qué variaciones son posibles en la interacción del actor?
  - 7. ¿Qué información del sistema adquirirá, producirá o modificará el actor?
  - 8. ¿Tendrá el actor que informar al sistema de los cambios en el entorno exterior?
  - 9. ¿Qué información desea el actor del sistema?
  - 10. ¿Desea el actor ser informado de cambios inesperados?

2.4. Desarrollo de casos de uso

- Recordando los requisitos básicos **de SafeHome**, definimos cuatro actores:
  - El propietario de la vivienda (un usuario)
  - El gestor de la instalación (probablemente la misma persona que el propietario de la vivienda, pero desempeñando un papel diferente)
  - Los sensores (dispositivos conectados al sistema) y
  - El subsistema de supervisión y respuesta (la estación central que supervisa la función de seguridad doméstica de safehome).
- En este ejemplo, sólo tenemos en cuenta al **propietario de la vivienda**. El actor propietario de la vivienda **interactúa** con la función de seguridad doméstica de **diferentes maneras**, utilizando el panel de control de la alarma, una tableta o un teléfono móvil.

2.4. Desarrollo de casos de uso

El dueño de la casa:

1. Introduce una contraseña para permitir el resto de interacciones
2. Consultas sobre el estado de una zona de seguridad
3. Consulta el estado de un sensor
4. Pulsa el botón del pánico en caso de emergencia
5. Activa y desactiva el sistema de seguridad

2.4. Desarrollo de casos de uso

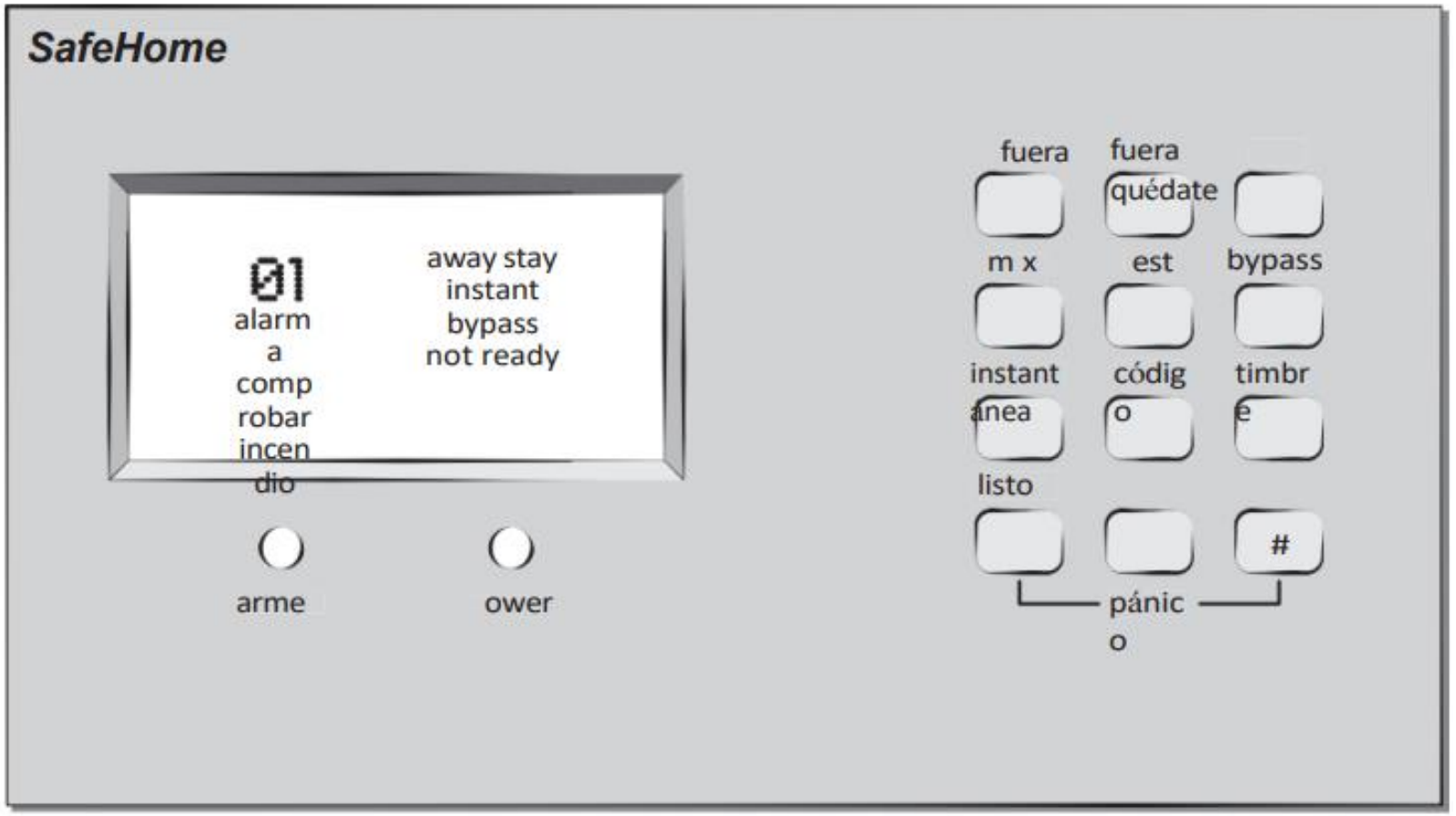
- Considerando la situación en la que el **propietario de la vivienda utiliza el panel de control**, el caso de uso básico para la activación del sistema es el siguiente:
- 1. El propietario **observa el panel** de control SafeHome para determinar si el sistema está listo para la entrada de datos.
  - Si el sistema no está listo, aparece un mensaje de no listo y el propietario debe cerrar físicamente las ventanas o puertas para que desaparezca el mensaje de "no listo". (Un mensaje de no preparado implica que un sensor está abierto, es decir, que una puerta o ventana está abierta).

2.4. Desarrollo de casos de uso

- 2. El propietario utiliza el teclado para **introducir una contraseña** de cuatro dígitos. La contraseña se compara con la contraseña válida almacenada en el sistema.
  - Si la **contraseña es incorrecta**, la central emite un pitido y se reinicia para recibir más datos.
  - Si la **contraseña es correcta**, la central espera a que se realicen más acciones.
- 3. El propietario selecciona y teclea "**Stay" o "Ausente**" para **activar el sistema.**
  - **Stay** activa sólo los **sensores perimetrales** (los sensores interiores de detección de movimiento se desactivan).
  - **Ausente** activa **todos los sensores**.
- 4. Cuando se produce la activación, el propietario puede observar una **luz roja de alarma.**

2.4. Desarrollo de casos de uso

SafeHome
panel de control



2.4. Desarrollo de casos de uso

- El **caso de uso básico** presenta una **historia** de usuario de **alto nivel** que describe la **interacción entre el actor y el sistema.**
- En muchos casos, los casos de uso se elaboran aún más para proporcionar muchos más detalles sobre la interacción.
- Por ejemplo, Cockburn sugiere una **plantilla** para descripciones detalladas de casos de uso (Ver en material complementario)

2.4. Desarrollo de casos de uso

- Desarrollo de un diagrama de casos de uso de alto nivel

(Ver en material complementario)

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

3. TRABAJO COLABORATIVO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Trabajo colaborativo

- *En grupos de 5 integrantes, investigar sobre una empresa y las actividades que se pueden digitalizar o automatizar.*

Receso

- *Para que todos los participantes pueden despejarse y regresen a la sesión con mayor predisposición para continuar, se brinda un pequeño receso de (10 minutos)*

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

4. EVALUACIÓN DE LA SESIÓN

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Evaluación de la sesión

- *Estimado estudiante,*
- *Ud. a sido evaluado en el transcurso de la sesión*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

5. CONCLUSIONES

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

5.1. Conclusiones

- *La ingeniería de requisitos, término que designa el amplio espectro de tareas y técnicas que conducen a la comprensión de los requisitos.*
- *Desde el punto de vista del proceso de software, la ingeniería de requisitos es una importante acción de ingeniería de software que comienza durante la actividad de comunicación y continúa en la actividad de modelado.*
- *La ingeniería de requisitos abarca siete tareas con límites a veces confusos: Inicio, obtención, elaboración, negociación, especificación, validación y gestión.*
- *Para el trabajo preliminar, se siguen las siguientes actividades: identificación de las partes interesadas, reconocer múltiples puntos de vista, hacia la colaboración, formular las primeras preguntas, requisitos no funcionales y trazabilidad*
-

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

6. AVISO O ANUNCIO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

6.1. Aviso o Anuncios

- *Para la siguiente sesión, debes revisar "Modelos de proceso"*

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

7. REFERENCIAS BIBLIOGRÁFICAS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Referencias Bibliográficas

- *Pressman, R., & Maxim, B. (2020). Software engineering: A practitioner's approach (Ninth edition). McGraw-Hill Education.*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

8. MATERIAL COMPLEMENTARIO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Material complementario

- *Software e Ingeniería de software (Pressman & Maxim, 2020)*

GRACIAS

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Ingeniería de Sistemas
Universidad Nacional de Cajamarca