



**FACULTAD DE QUIMICA E INGENIERIA DEL ROSARIO**  
**PONTIFICIA UNIVERSIDAD CATÓLICA ARGENTINA**

**TESINA DE GRADO**

Propuesta de Metodología y Documentación  
en el Desarrollo de Software.

**Autor:**

A.S. Yanina León

**Directora de Tesina:**

Mg. Cristina Bender

**Carrera:**

Licenciatura en Sistemas  
y Computación

**Cátedra:**

Seminario de Sistemas

**Año 2018**

## INTRODUCCIÓN

La presente Tesina tiene como objetivo proponer una forma de trabajo diferente en las etapas de desarrollo de software, rescatando algunos aspectos de las metodologías más antiguas de Análisis y Diseño de Sistemas y de las Metodologías Ágiles que se están usando hoy en día en muchas Empresas de la Industria del Software.

En primer lugar se presenta un fundamento por el cual se elabora esta tesina con el fin de proponer una Metodología de Análisis y Diseño diferente y basada en Metodologías Ágiles pero agregando documentación que he creado para la etapa previa a la codificación y posterior a la generación de código del software. Esta propuesta surge de la necesidad de innovar sobre las Metodologías ya existentes debido a ciertas desventajas que a mi criterio, fueron surgiendo en su aplicación.

Seguido a la Fundamentación se presenta un Marco Teórico donde se hace referencia a las distintas Metodologías de Análisis y Diseño de Sistemas que desde los inicios de la Ingeniería de Software, distintos autores fueron planteando herramientas, las cuales quedaron expresadas en libros y publicaciones web, y que surgieron de sus propias experiencias que han logrado a lo largo su trayectoria laboral.

La base del aprendizaje del análisis y diseño de software fue la Metodología de Análisis y Diseño Estructurado. Sobre esta Metodología distintos autores hicieron sus aportes innovando sobre las herramientas ya conocidas y adaptándolas a sus formas de trabajo y a las necesidades que surgían en las distintas etapas del desarrollo de software a lo largo del tiempo.

En el anexo se puede ver un caso práctico documentado con las metodologías de análisis y diseño estructurado y con notación UML.

Luego presento mi propuesta, donde se describen los pasos para documentar cada etapa del desarrollo en dos situaciones diferentes que podemos enfrentarnos al momento de brindar soluciones informáticas a otras empresas, que son Implementar un Sistema Nuevo o Implementar Cambios en un Sistema Productivo.

Finalmente en la conclusión de mi propuesta presento un caso práctico que aplica mi propuesta de documentación de análisis y diseño.

## ÍNDICE

1. Fundamentación .....	4
2. Marco teórico .....	9
2.1. Modelo de análisis y diseño estructurado .....	9
2.2. Metodologías orientadas a procesos.....	10
2.3. Lenguaje unificado de modelado.....	17
2.4. Metodologías ágiles.....	25
2.4.1. Scrum .....	26
2.4.2. Kanban .....	26
2.4.3. Xp o programación extrema .....	27
3. Conclusión sobre las metodologías presentadas.....	28
4. Propuesta .....	31
4.1. Metodología de ingeniería de software para un sistema en producción.....	31
4.2. Metodología de ingeniería de software para un sistema nuevo a implementar.....	35
4.2.1. Relevamiento .....	36
4.2.2. Definición de procesos prioritarios .....	37
4.2.3. Asignación de casos de uso a programadores.....	40
4.2.4. Programación, generación de código y documentación técnica. ....	40
4.2.5. Pruebas en servidor de desarrollo. ....	41
4.2.6. Pasaje a servidor de testing y ejecución de las pruebas.....	41
4.2.7. Pasaje a servidor de producción.....	42
5. Conclusión .....	43
6. Ejemplo de documentación propuesta. ....	45
6.1. Documento de casos de uso basados en transacciones (relevamiento, análisis y diseño).....	45
6.2. Documento de diagramas de actividades para los procesos de negocio (relevamiento).....	47
6.3. Casos de pruebas (pruebas en servidor de desarrollo y de testing).....	48
6.4. Documentación técnica de procesos (tarea posterior a la etapa de programación). ....	49
7. Evaluación de la propuesta .....	52
8. Anexo.....	53
8.1. Caso práctico documentado con metodología de análisis y diseño estructurado .....	53
8.2. Caso práctico documentado con notación uml .....	67
9. Bibliografía .....	75

## 1. Fundamentación

Vivimos en una sociedad de información global emergente, con una economía que depende cada vez más de la creación, la administración y la distribución de la información.

Muchas Empresas y Organizaciones tienen éxitos en sus objetivos por la implantación y uso de los Sistemas de Información. Por este motivo, constituyen un campo esencial de estudio en administración y gerencia de empresas. El manejo de información generada desde los Sistemas Informáticos difiere en forma significativa del manejo de datos producidos manualmente.

Los sistemas de información son la mejor opción para las empresas que desean ir al ritmo de los negocios de la actualidad ya que hoy se necesita rapidez y seguridad al momento de verificar datos reales.

A través de ellos podemos además de obtener resultados rápidos y confiables, organizar la información que requiere la empresa y tener acceso a ella en cualquier momento, al mismo tiempo que reduce el espacio físico que ocupan los archivos y documentos escritos.

Existen diferentes metodologías de Desarrollo de Sistemas, que se fueron aplicando en años, donde se abordaba un Análisis y Diseño del Sistema, previo a la codificación e implementación.

En las distintas metodologías de Desarrollo de Sistemas, se busca de forma eficiente y rápida la elaboración del software. Debemos conocer las distintas metodologías para saber implementar la más adecuada a la hora de realizar un proyecto de software, conocer ante todo, sus ventajas y desventajas, y lo más importante cabe mencionar que no sólo podemos utilizar una metodología, podemos hacer una combinación de dos o más dependiendo del administrador de proyecto y sus habilidades para implementarlas, haciendo que sea aún mejor la calidad en el desarrollo y el ordenamiento de las tareas a realizarse.

Para dar solución a ciertos requerimientos de sistemas de una Organización, los profesionales de sistemas utilizan diferentes metodologías de Desarrollo de Software, desde las Metodologías Tradicionales a Metodologías Ágiles como la más reciente.

Desde sus comienzos estas metodologías han ido evolucionando conforme a la evolución de las tecnologías de software y a las necesidades del mercado, lo cual fue necesario cambiar los paradigmas de trabajo en los ámbitos de las organizaciones que brindan soluciones informáticas a otras empresas.

Dentro de cada Metodología de Desarrollo de Sistemas la etapa de Análisis y Diseño, también fue cambiando.

El **Análisis y Diseño Estructurado** y el **Lenguaje de Modelo Unificado** más conocido como UML proponen crear amplia documentación, variedad de gráficos y herramientas que permiten hacer frente al análisis y diseño previo a la programación del software.

El **Análisis y Diseño Estructurado** requiere del desarrollo de una serie de etapas para representar el modelo esencial del sistema, donde llevan mucho tiempo de elaboración y documentación.

En la metodología tradicional existen varios enfoques del Ciclo de Vida de un Proyecto Estructurado, como la Metodología en Cascada, Método de Prototipos, Modelo en Espiral y Modelo Incremental, en donde la etapa de análisis y diseño era una fase fundamental que debía ser bien documentada, para que el área de programación genere el código en base a esta documentación.

Las desventajas presentes en la Metodología Tradicional, es que se debe completar una etapa antes de comenzar la siguiente etapa con su correspondiente documentación, o se hacen revisiones de cada etapa del análisis y diseño lo cual genera demoras para avanzar hacia la etapa siguiente, que consiste en la programación del módulo en estudio y generar el entregable al cliente.

Como es una Metodología de Análisis orientada a procesos, si hay que modificar una parte del proceso o todo el proceso, implica un cambio masivo en todo el análisis y su correspondiente documentación.

Es inseguro para el cliente porque no ve plasmado su requerimiento hasta el momento en que recibe el entregable, generando en ocasiones, disconformidad de los usuarios del sistema o conflictos entre las personas involucradas en el proyecto de desarrollo, por no cumplir con las necesidades inicialmente especificadas.

En los primeros pasos en la Ingeniería de sistemas, se relevaban todos los requerimientos al inicio del proyecto y el resultado no era el esperado en relación a su funcionalidad, inclusive los presupuestos elaborados que surgían de las estimaciones de los requerimientos no alcanzaban a cubrir los costos reales de desarrollo e implementación por no captar de entrada todos los requerimientos necesarios para cubrir las reglas de negocio.

En 1994 la Empresa Standish Group<sup>1</sup>, con el fin de obtener un informe sobre el resultado que obtenían las Empresas de la Industria de Software en relación a la Implementación de sus proyectos de Sistemas, obtuvo un resultado no muy favorable.

- El 31% de los proyectos de software fueron cancelados antes de tiempo (2480 proyectos).
- En las grandes compañías, sólo el 9% de los proyectos de software fue entregado a tiempo y dentro del costo presupuestado y el 16% satisfizo estos requerimientos en las compañías pequeñas.<sup>2</sup>

1 Standish Group International, Inc. o Standish Group es una firma internacional independiente de asesoramiento en investigación de TI fundada en 1985, conocida por sus informes sobre proyectos de implementación de sistemas de información en el sector público y privado.

2 <https://www.slideshare.net/SergioRios/unidad-13-analisis-de-requerimientos> (Slide 6) - Referencia en la página <http://www.pmoinformatica.com/2016/08/tecnicas-analisis-requerimientos.html>

Luego en 1995 Standish Group, realizó una encuesta a los participantes de los proyectos para saber el motivo por el cual los proyectos fracasaban y las respuestas fueron diversas:<sup>3</sup>

- Requerimientos con definiciones incompletas (13,1%)
- Falta de compromiso del usuario (12,4%)
- Falta de recursos (10,6%)
- Expectativas no realistas (9,9%)
- Falta de Soporte Ejecutivo (9,3%)
- Requerimientos y Especificaciones cambiantes (8,7%)
- Falta de planeamiento (8,1%)
- Fin de la necesidad del sistema (7,5%)

Hay proyectos de sistemas que son muy grandes, abarcan mucha funcionalidad, y trabajar con metodologías que siguen una planificación muy estructurada y con la elaboración de documentos muy extensos para la definición de los requerimientos, puede provocar que nunca se concrete su implementación.

En el **Anexo 8.1** se muestra un **Caso Práctico Documentado con Metodología de Análisis y Diseño Estructurado**, donde podemos ver la extensa documentación que se debe desarrollar para las diferentes etapas del proyecto y una breve lista de requerimientos que fue definida por los usuarios en la etapa de relevamiento.

Otras de las formas mencionadas anteriormente para documentar el análisis y diseño de un sistema, es utilizando notación **UML, Lenguaje de Modelo Unificado**.

**UML**, no es un método o metodología, es una notación usada para expresar un diseño de software que fue propuesta por Booch, Rumbaugh y Jacobson.<sup>4</sup>

Es un lenguaje gráfico que permite visualizar, especificar, construir y documentar los artefactos de los sistemas de objetos distribuidos.

UML es uno de los lenguajes de modelado más utilizados por las Empresas de la Industria de Software.

Consiste en modelar el sistema analizando distintos aspectos del mismo, ya sea aspectos estáticos para modelar la estructura que componen el sistema, como dinámicos o de comportamiento (procesos) con la alternativa de utilizar una gran variedad de Diagramas.

Una de las desventajas que presenta UML, al igual que la Metodología de Análisis y Diseño Estructurado, es el tiempo que lleva mantener actualizados los diagramas que representan el diseño del sistema.

Para representar el diseño y la funcionalidad de una pantalla de un sistema, requiere de muchos diagramas excesivamente extensos, lo cual no resulta para nada práctico si buscamos la agilidad que se requiere hoy en día en estas tareas de desarrollo de software.

<sup>3</sup> [https://www.standishgroup.com/sample\\_research\\_files/chaos\\_report\\_1994.pdf](https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf)  
<http://www.pmoinformatica.com/2016/08/tecnicas-analisis-requerimientos.html>

<sup>4</sup> El Lenguaje Unificado de Modelado: Manual de Referencia. Pearson Educación (1999)

Requiere de mucho tiempo de capacitación a los recursos involucrados en el proyecto de desarrollo de software, tanto para analistas como programadores. Esto obliga no sólo a que los analistas de sistemas conozcan la notación UML, sino también que los programadores comprendan su notación para volcar esa lógica en el momento de llevar a cabo la tarea de diseño y programación.

Otra gran desventaja es que UML es una notación empleada sólo para la programación orientada a objetos, no es factible su uso para todos los lenguajes de programación.

En el **Anexo 8.2**, se muestra un **Caso Práctico Documentado con Notación UML**.

Hoy en día la mayoría de las Empresas de la Industria del Software, implementan como marco de trabajo las **Metodologías Ágiles** que dan un gran aporte a los continuos cambios que hay que enfrentar a nivel de negocio.

Entre las más utilizadas están **SCRUM, KANBAN y XP** (Programación Extrema).

En las Metodologías Ágiles la documentación tiene un nivel de detalle más equilibrado y limitado y se priorizan los encuentros cara a cara entre las personas involucradas en el proyecto como herramienta de comunicación.

Si bien tiene sus ventajas, ya que existe mayor agilidad en las entregas de producto que se realizan al cliente, una de las grandes desventajas es que toda la planificación y las definiciones de software dependen de la implicancia de los participantes del proyecto.

Al no existir documentación en detalle, ya que se basa en la comunicación verbal mediante reuniones frecuentes, las definiciones que se realizan sobre un proyecto de software quedan en la cabeza de los participantes involucrados en el proyecto y se corre el riesgo de perder esta información si algún participante o varios abandona el proyecto.

También, puede ser problemático cuando no hay espíritu colaborativo entre los participantes o falta de empatía que dificulte la comunicación.

Es por esto que en esta Tesina, se propone una Metodología y Documentación diferente para el Desarrollo de Software, donde se busca rescatar las ventajas de las metodologías ya conocidas y que se vienen utilizando en el mercado. Además se propone innovar en algunos aspectos para adaptarnos a las Metodologías Ágiles que surgieron recientemente sin dejar de lado la tarea de documentación y a su vez que la misma no sea tediosa a la hora de llevarla a cabo.

La propuesta en sí es aplicar en el desarrollo de software Metodologías Ágiles como SCRUM o Programación Extrema, donde el desarrollo es ágil, iterativo e incremental, y utilizar una herramienta de software (ejemplo Mantis Bug Tracker <sup>5</sup> o Kanban) que permita hacer un seguimiento de los requerimientos que realiza el cliente con el fin de dar solución a las necesidades que se presentan en el negocio día a día.

---

<sup>5</sup> <https://www.mantisbt.org/>

Esta evolución del Proyecto de Software generada por cada iteración, que incluye una planificación, análisis de requerimientos prioritarios, diseño, codificación y pruebas, debe estar documentada en cada etapa.

Se propone una forma diferente de especificar los casos de uso a la forma utilizada en la notación UML, esta opción tiene un enfoque más técnico porque como en las metodologías ágiles el usuario forma parte del proyecto y tiene una implicancia más importante con mayor conocimiento del producto que se va a implementar, podemos interactuar con él utilizando un lenguaje más técnico y no tan exclusivo a la interacción con el sistema como se presenta en el Lenguaje Visual UML.

Además se presenta una alternativa de documentación técnica de procesos que son codificados en el área de desarrollo en una Empresa de Ingeniería de Software, sin utilizar diagramas extensos que son difíciles de mantener o leer y sólo se pueden utilizar con lenguajes orientados a objetos.

La organización global de esta Tesina consiste en hacer una breve exposición de conceptos relacionados al análisis y diseño de sistemas que se fueron utilizando a lo largo de su historia por grandes empresas de la Industria del Software y que fueron presentando distintos autores en libros y publicaciones. Esto se puede ver en el siguiente ítem en el Marco Teórico y una conclusión sobre las ventajas y desventajas de cada metodología.

Luego les presento mi propuesta cómo una forma diferente de documentar el Análisis y Diseño de Sistemas con dos alternativas de trabajo como metodología de desarrollo.

Finalmente les dejo una Conclusión sobre mi propuesta.



## 2. Marco Teórico

### 2.1. Modelo de Análisis y Diseño Estructurado

En los inicios de la Industria del Software, el Modelo en Cascada, se tomaba de base para realizar un Análisis Estructurado, donde la propuesta era similar a una línea de ensamble de producción, en este caso para desarrollar software.

El Modelo en Cascada, proponía finalizar una etapa antes de comenzar la siguiente. Este modelo tenía muchas desventajas dado que no era posible iniciar la programación si no estaban analizados y diseñados todos los requerimientos.

Una vez que se concluía con la Etapa de Diseño y el mismo era validado por el cliente, se continuaba con la Etapa de Programación, y luego iniciada esta etapa, no admitían agregar ningún cambio.

Es justamente una de las grandes desventajas que presentaba este modelo.

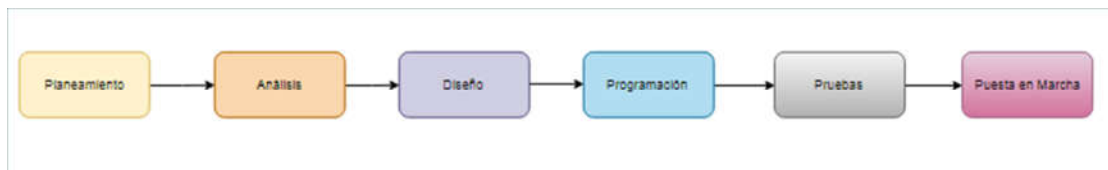


Figura 1: Etapas de la Metodología Tradicional – Modelo en Cascada<sup>6</sup>

Así surgen los modelos iterativos, donde antes de finalizar una etapa se podía volver a la etapa anterior para ajustar ciertos requerimientos ante ciertos cambios o agregar nueva funcionalidad requerida por el usuario después de ver los entregables.

El modelo iterativo incremental, proponía ir analizando, desarrollando y probando el sistema en forma incremental, lo cual se iniciaba con uno o dos requerimientos generando un mini-ejecutable o prototipo y luego se iban agregando más requerimientos siguiendo siempre la etapa de análisis, diseño, codificación y testing.

Por cada iteración se agregaba nueva funcionalidad al sistema, y permitía hacer breves presentaciones al usuario, donde permitía mostrarle en forma temprana el sistema en funcionamiento.

Por cada entrega del producto que se hacía al usuario, surgían ajustes que realizar en el sistema, donde tanto el analista como el programador, debían actualizar la documentación de análisis y el código.

<sup>6</sup> Análisis Estructurado Moderno. Edward Yourdon



## 2.2. Metodologías orientadas a procesos

Las metodologías orientadas a procesos comprenden una fase de análisis estructurado y los métodos de DeMarco, Gane y Sarson, Yourdon, son algunos de los actores que permiten lograr esto<sup>7</sup>

Se basan en la creación de un documento con una especificación estructurada de los requerimientos realizando diferentes modelos lógicos y físicos.

La metodología orientada a procesos utiliza diferentes herramientas:

- **Diagrama de flujo de datos:** Estos son diagramas que representan los procesos de datos que deben llevar a cabo un sistema a distintos niveles de abstracción y los datos que hay entre las funciones. Se basan en la utilización de un método descendente de descomposición funcional para definir los requisitos del sistema. **El DFD** consiste en un modelo gráfico explotado en niveles de procesos del sistema que parten de un diagrama de contexto.
- **Diccionario de datos:** Es el conjunto de las definiciones de todos los datos que aparecen en el diagrama de flujos de datos.
- **Especificaciones de procesos:** como se obtienen las salidas del proceso a partir de sus entradas.

**Metodología de Demarco,** en primera instancia se realiza el estudio del sistema físico actual con DFD, luego se elabora el modelo lógico actual con DFD y por último se llega al nuevo modelo lógico tomando en cuenta las necesidades del cliente proponiendo Modelos Físicos alternativos.

Se realiza una valoración de costo/beneficio de las alternativas propuestas y se presenta una documentación.

**Metodología de Gane y Sarson:** es una metodología que tuvo su mayor auge en los años '80, es muy parecida a la metodología propuesta por Demarco.

Consiste, en primer lugar crear el modelo lógico actual elaborando un Diagrama de Flujo de Datos.

Luego se crea un modelo lógico realizando especificaciones de procesos y el modelo de datos con los almacenes de datos del sistema (en 3° Forma Normal). Esta es precisamente la diferencia que tiene con la Metodología Demarco.

**Metodología de Yourdon:** con el fin de Analizar y Diseñar el sistema de acuerdo a los requerimientos que hace el usuario, Yourdon propone crear el Modelo Esencial del sistema.

---

<sup>7</sup> Análisis Estructurado y Especificación del Sistema publicada en 1979 por DeMarco.  
Análisis Estructurado de Sistemas de Gane & Sarson.  
Análisis y Diseño Estructurado de E. Yourdon.

El **Modelo Esencial**, es un modelo lógico que especifica los requerimientos necesarios para satisfacer las necesidades y expectativas que tiene el usuario sobre el sistema, pero no describe ni menciona la forma en que se realizará su implementación.

Cuando se elabora el Modelo Esencial, se supone que disponemos de tecnología perfecta y que tendremos fácil acceso a la misma y a costo cero.

Crear el Modelo esencial, no es una tarea sencilla. Entre las dificultades que se pueden presentar, enumeramos algunas:

- Crear una secuencia arbitraria de flujos de datos. Esta secuencia es necesaria que siga el orden que queda determinado por el ingreso de datos, el procesamiento de los mismos y su salida o respuesta al estímulo externo.
- Tener datos redundantes entre los almacenes de datos.
- Procesos con funcionalidades similares o repetidas dentro de otros procesos.
- Trabajar con usuarios inexpertos, que no tienen claridad sobre los requerimientos que solicitan.
- Flujos de datos que se envían a burbujas de procesos y que no dan ninguna respuesta, no se utilizan los datos para generar ningún reporte o salida.

El modelo esencial consiste en armar el **Modelo Ambiental** y el **Modelo de Comportamiento**.

- **Modelo Ambiental**

Todo sistema es parte de un sistema mayor o interactúa con otros subsistemas. Es por eso que en esta etapa se definen las fronteras del sistema a analizar y nos enfocamos en el ambiente con el cual va a interactuar.

El modelo exterior a nuestro sistema es lo que nos interesa, para luego modelar el interior de nuestro sistema que es lo que conocemos como Modelo de Comportamiento.

La frontera entre el sistema y el ambiente externo con el cual interactúa no es fácil de detectar, inclusive muchos usuarios que participan del proyecto, no pueden definir esta línea, lo que comúnmente se denomina zona gris. Es una zona arbitraria que queda sujeta a negociación entre los que participan del proyecto.

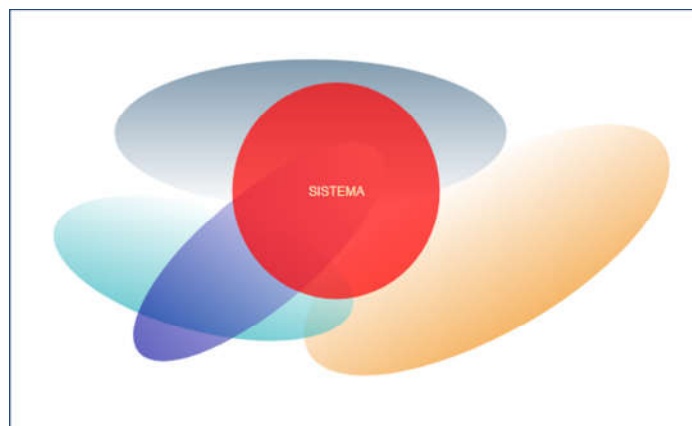


Figura 2: Frontera entre el Sistema y el Ambiente Externo

### ❖ ***Declaración de Propósitos***

Lo primero que se define en el Modelo Ambiental es la Declaración de Propósitos. Consiste en una Declaración textual, breve y concisa del objetivo del sistema.

### ❖ ***Lista de Eventos***

Otras de las herramientas que forman parte de este modelo es la Lista de Eventos. Se listan en una tabla los eventos de entrada y las respuestas del sistema a estos eventos. Previo a esta lista de eventos, se hace una narrativa del sistema, donde se expresa por cada evento cómo interactúan las entidades externas con el sistema. Esta narrativa surge de preguntas y encuestas que se realizaron en el relevamiento que se hizo al cliente que desea implementar el nuevo sistema.

#### ***Ejemplo de Lista de Eventos:***

1. Encargado del Sector Almacenes ingresa un pedido de materiales.
2. Encargado del Sector Compras ingresa un pedido de Cotización.
3. Diariamente se emite listado de proveedores con saldo a favor.
4. Cuando el stock llega a su punto de pedido, generar los pedidos de producto en forma automática para reponer el stock.

En el ejemplo se muestran tres tipos de eventos:

El evento 1 y 2 corresponde a eventos de flujos de datos. En este tipo de eventos se ingresan datos al sistema, que luego serán procesados para dar una salida.

El evento 3, es un evento de Flujo Temporal. El sistema debe estar preparado para emitir diariamente un Listado de Proveedores, el cual se busca que el sistema lo dispare en forma automática de acuerdo a la frecuencia indicada.

El evento 4, se busca que el sistema realice una acción determinada, y esta acción se dispara por un control interno que realiza el sistema. Al detectar que un producto llegó a un nivel de stock que es igual a su punto de pedido (control interno), el sistema dispara una acción que es generar los pedidos de stock al sector Compras. Este evento corresponde a un Flujo de Control.

### ❖ ***Diagrama de Contexto***

Es un caso especial del Diagrama de flujo de datos, donde con una única burbuja se representa todo el sistema que responde a estímulos o eventos externos de entidades o a eventos temporales.

Las personas o entidades con los que se comunica el sistema son los terminadores y se representan con rectángulos.

La información que el sistema necesita guardar para responder a los eventos, se guarda en los almacenes de datos y estos almacenes se representan con líneas paralelas.

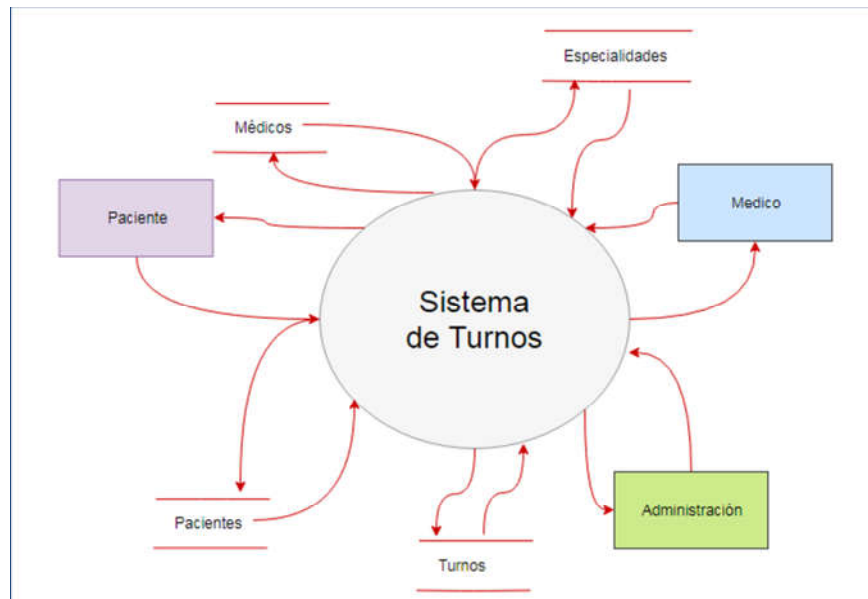


Figura 3: Diagrama de Contexto

Este tipo de diagrama resulta práctico y es útil cuando se representa un sistema de pocos eventos y pocos almacenes de datos. Pero presenta complicaciones y resulta confuso cuando hay que representar sistemas grandes, que es el caso de los sistemas reales.

Luego se hace un análisis más exhaustivo y se analiza el funcionamiento interno del sistema, esto se logra armando el **Modelo de Comportamiento**.

- **Modelo de Comportamiento**

Uno de los diagramas principales del Modelo de Comportamiento es el **Diagrama de Flujo de Datos** (DFD).

Los **Diagramas de Flujo de Datos**, son diagramas que representan los *procesos* funcionales del sistema, y muestra como fluyen los datos que ingresan al sistema y se transforman en flujos de salida. Los datos que ingresan, se guardan en almacenes o "tanques de almacenamiento de datos" y se obtienen flujos de salida como respuesta a las entidades externas.



El DFD es una red de procesos, los cuales se comunican entre ellos con flujos de datos y almacenes de datos.

En principio, se crea el DFD de nivel cero. El DFD de nivel cero se explota en distintos niveles, para tener una definición más detallada de cada proceso.

La construcción de los DFD de Sistemas Reales son complejos por la gran cantidad de funciones que incluyen, es por eso que se crean niveles de procesos, donde cada proceso del DFD de nivel cero, se subdivide en varios procesos para reducir la complejidad y poder trabajar más en detalle cada funcionalidad del sistema.

No existe un límite de jerarquía de niveles. Esto quedará determinado por la necesidad de dar definiciones más detalladas a cada proceso o funcionalidad.

Los procesos se representan mediante círculos y deben estar numerados, la numeración no indican una secuencia de ejecución, es simplemente para identificarlos y luego dividirlos en subprocesos.

Se deben evitar procesos que reciban entradas de flujo de datos, y que no den respuestas o que no realicen actualizaciones de datos en almacenes. En este caso es un proceso que no tiene sentido su representación en el DFD.

Los *flujos de datos* se representan con flechas que van desde las entidades a los procesos, o de los procesos a los almacenes. Representan la información que se mueve dentro del sistema. Deben etiquetarse con un nombre representativo acorde a los datos que mueve.

Puede haber Flujos Temporales o Flujos de Datos o Flujos de Control como mencionamos con anterioridad. Estos flujos de control se utilizan en sistemas de tiempo real, donde la función es generar la acción de otro proceso.

Los *almacenes de datos* se representan con líneas paralelas. Son colecciones de datos que se almacenan con cada ingreso de datos que se da en el sistema. Se requiere guardar estos datos para obtener las salidas requeridas por las entidades externas. El conjunto de almacenes es el comienzo de la futura base de datos del sistema.

Cada vez que un flujo de datos ingresa a un almacén se interpreta que está actualizando datos, lo podemos entender como un alta o una baja o una modificación en una tabla de la base de datos.

Cuando un flujo de datos sale de un almacén de datos está obteniendo datos, está recuperando datos para generar una salida. Estas salidas las podemos ver como salidas de datos por pantalla o salidas de datos en reportes o listados.

Cuando se modela el DFD, en forma paralela se arma el *Diagrama de Entidad-Relación*. Esto permite hacer una equivalencia entre ambos diagramas.

Otra de las herramientas utilizadas en Análisis Estructurado que se emplea en el Modelo de Comportamiento es el *Diccionario de Datos*.

Por cada evento se especifican los datos que ingresan al sistema, los datos que salen del sistema y los datos que se guardan en los almacenes de datos. El Diccionario de Datos es

más simple de realizar cuando se hace en forma conjunta con el DFD, el DER y las especificaciones de procesos, de esta forma se logran balancear las herramientas de modelado y permite hacerlo siguiendo la secuencia de cada evento.

Luego se realizan las **Especificaciones de Proceso**, que definen mediante un lenguaje pseudocódigo un proceso o funcionalidad del sistema donde describe como se transforman las entradas del sistema en salidas para dar respuesta a los eventos.

Es útil tanto para el analista como para el programador, porque les permite tener una misma visión sobre la funcionalidad de un proceso del sistema.

Seguido se elabora un **Diagrama de Entidad-Relación** que describe en forma abstracta la forma que se almacenan los datos del sistema. Se representan las estructuras de datos y como se relacionan.

Este diagrama tiene una notación que fue discutida por varios autores, en cuanto a las reglas de armado.

Los componentes de este diagrama son:

- Tipos de Objetos: representan una colección de objetos del mundo real, el cual cada objeto en forma individual, puede describirse con uno o más datos, como se hace en el Diccionario de Datos y puede ser identificado en forma unívoca.
- Relaciones: representan como se asocian los datos de un objeto con otro objeto.
- Indicadores asociativos de Tipo de Objeto: es un tipo de objeto que funciona como objeto y como relación. Es una relación entre dos objetos, de la cual se desea guardar datos.
- Indicadores de Subtipo/Supertipo: Permite clasificar los objetos en categorías y subcategorías.

Ejemplo:

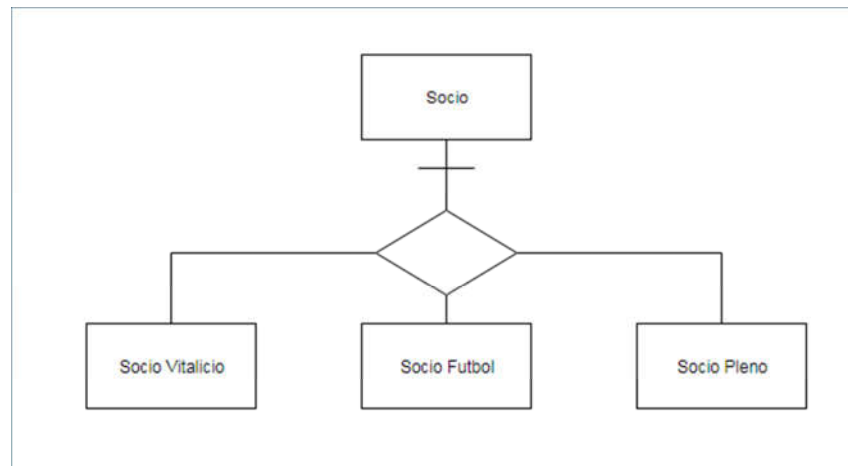


Figura 4: Diagrama de Entidad-Relación indicando relación de Supertipo/subtipo.

### • **Modelo de Implantación de Usuario**

En el Modelo de Implantación de usuario, se define qué procesos serán automatizados y la Interfaz Humana, es decir el diseño de las entradas y salidas de datos.

El usuario podría optar por automatizar algunos procesos, y otros procesos dejarlos fuera del proyecto de implantación o incorporarlos más adelante después de la puesta en producción.

La determinación de la Interfaz humana, implica definir junto al usuario, la elección de dispositivos o medios de Entrada y Salida de datos.

Las entradas de datos al sistema pueden ser por diferentes medios:

- en una pantalla mediante un teclado
- escáner de código de barra
- escáner de imágenes
- escáner óptico
- Pizarras táctiles
- Entrada de Voz
- Lector de huellas digitales
- Llaves o Tarjetas electrónicas
- Importación de archivos planos o excel, donde también se requiere de la programación de una pantalla para poder realizarlo.
- Procesos batch que permiten tomar información de una base de datos externa.
- Web services, es una de las últimas tecnologías que se utiliza hoy en día. Primero comenzó a usarse el protocolo SOAP, que usa el lenguaje XML, lo cual en el archivo XML permite intercambiar todo tipo de datos entre dos aplicaciones web.

Otra tipo de web services es API REST, que utiliza el protocolo HTTP (por puerto 80) para la transferencia de datos usando javascript como lenguaje, el cual es un formato de texto ligero para intercambiar datos entre equipos remotos.

Existen otros protocolos además de HTTP para la transferencia de datos, como son FTP (para transferencia de archivos) y SMTP (para servicio de correo)

El Sistema que estamos automatizando consume el web service, obteniendo datos que serán procesados y darán una respuesta. Esta respuesta puede ser una interfaz a otro sistema, una salida por pantalla u otro dispositivo o simplemente se hará una actualización en la base de datos.

Las Salidas de datos del Sistema, pueden ser:

- Salidas Impresas (Reportes, Listados, Comprobantes)
- Salidas por Pantalla (Imágenes, Gráficos, Grillas de Datos, Reportes o Listados)
- Salidas de voz (audio).
- Interfaz hacia otro sistema (ejemplo archivos planos)
- Web Service



La Metodología de Análisis y Diseño Estructurado fue muy utilizada entre los años 60 y los años 80, época de mayor auge, en donde la programación era estructurada y se usaban lenguajes de programación de alto nivel.

Luego aparecieron las herramientas CASE, que permitían al programador hacer un diseño de software asistido, usando notación UML.

### 2.3. Lenguaje Unificado de Modelado.

El Lenguaje Unificado de Modelado, comienza a gestarse a mediados de los años 90 y a partir del año 1997 la OMG (Object Management Group) lo adopta como un lenguaje de modelado estándar para diseñar tecnología orientada a objetos.

UML en la actualidad es uno de los lenguajes de modelado más utilizados por Empresas de la Industria del Software de todo el mundo.

La OMG es un consorcio de Empresas que tienen como objetivo promover la tecnología orientada a objetos como son algunas de las siguientes:

- UML, Lenguaje Unificado de Modelado, que permite una representación gráfica del Sistema en estudio, utilizando diagramas de diversos propósitos.
- XMI o XML, es una especificación que permite intercambiar metadatos,
- CORBA, permite la integración y el trabajo conjunto de diferentes procesos codificados en distintos lenguajes de programación y distribuidos en varios equipos.
- BPMN (Modelo de Notación de Procesos de Negocio), es una notación a mi criterio de las más prácticas para realizar un relevamiento de negocio y que permite al usuario entender la notación sin tener conocimiento previo del lenguaje de modelado. Simplemente muestra un Workflow de negocio definido por Gerentes y Administrativos de la Empresa que se está relevando. No es suficiente para mostrar niveles de detalle de procesos.

Existen muchas herramientas de programación de software que ofrecen un entorno de desarrollo utilizando notación UML.

Algunas de ellas son:

- ARGOUML, es una aplicación que permite crear diagramas UML, está escrita en JAVA y permite generación de Código: Java, PHP, Python, C++ y Csharp (C#).<sup>8</sup>
- Otra aplicación que permite realizar diagramas UML es BOUML, utilizada para especificar y generar código en JAVA, C++, PHP, IDL, MySQL, Python.<sup>9</sup>

<sup>8</sup> <https://es.wikipedia.org/wiki/ArgoUML>

<sup>9</sup> <https://es.wikipedia.org/wiki/BOUML>



- JDeveloper es un ambiente de desarrollo integrado que fue creado por la Empresa Oracle Corporation que permite generar lenguaje Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML y otros.<sup>10</sup>
- Telelogic Rhapsody es una herramienta utilizada para el desarrollo y el testing de software. Se crean los modelos basados en lenguaje UML, y permite una integración con la documentación y la programación.<sup>11</sup>
- Rational Rose es otra herramienta de diseño orientada a objetos, creada por IBM, donde a partir de la creación de modelos usando diagramas UML, permite generar código Java y C++ en forma automática.<sup>12</sup>

Hasta aquí mencionamos las diferentes tecnologías y algunas de las herramientas de programación que se usan en la Ingeniería de Software, algunas con mayor o menor ventaja sobre otras, que se basan en notación UML.

UML, como se cita anteriormente es un lenguaje que permite modelar el sistema con diagramas.

El objetivo o funciones principales son:

- Visualizar: expresar de forma gráfica un sistema y que pueda ser interpretado de la misma manera por otra persona.
- Especificar las características de un sistema antes de realizar la codificación.
- Construir Diseños de sistemas a partir de modelos.
- Documentar: Los diagramas que se van creando en cada modelo sirven como documentación del sistema desarrollado.

Los diagramas que se utilizan en UML, muestran una perspectiva diferente del sistema, con lo cual todos son importantes en su aplicación.

Para definir y especificar un sistema, UML utiliza vistas, donde cada una de estas vistas incluye diferentes diagramas.

- Vistas de Casos de Uso. Es la base del desarrollo funcional del sistema dado que describe el comportamiento. Los diagramas que incluye son Diagramas de Casos de Uso y Diagramas de Actividad.
- Vista Lógica. Se especifica el diseño del Sistema, utilizando para definir la Estructura de datos; los Diagramas de Clase y Diagramas de Objetos. Luego para mostrar el comportamiento; Diagramas de Estado, Diagrama de Secuencia, Diagrama de Colaboración y Diagrama de Actividad.
- Vista de Componentes: Esta vista permite modelar como se organizan los módulos de programas y archivos que formarán parte del sistema. Se utiliza sólo el Diagrama de Componentes.

<sup>10</sup> <https://es.wikipedia.org/wiki/JDeveloper>

<sup>11</sup> [https://es.wikipedia.org/wiki/Telelogic\\_Rhapsody](https://es.wikipedia.org/wiki/Telelogic_Rhapsody)

<sup>12</sup> [https://www.ibm.com/support/knowledgecenter/es/SS4JE2\\_7.5.5/com.ibm.xttools.sample.rose.model.doc/topics/sample\\_rose\\_intro.html](https://www.ibm.com/support/knowledgecenter/es/SS4JE2_7.5.5/com.ibm.xttools.sample.rose.model.doc/topics/sample_rose_intro.html)

- Vista de Implantación. Es donde se definen los nodos y dispositivos periféricos, es una descripción de la Arquitectura Física del Sistema a implementar. En esta vista se usa el Diagrama de Implantación.
- Vista de Concurrencia. Se muestran aspectos estáticos como asignación de los componentes a la arquitectura física, y los aspectos dinámicos de su interacción. Se representan los Diagramas de Componentes e Implantación para la arquitectura física y para la descripción dinámica los Diagramas de Estado, Diagrama de Secuencia, Diagrama de Colaboración y Diagrama de Actividad.

La versión **UML 1.0** modela con los siguientes diagramas:<sup>13 14</sup>

1. Diagrama de casos de uso.
2. Diagrama de clases.
3. Diagrama de objetos.
4. Diagrama de secuencia.
5. Diagrama de colaboración.
6. Diagrama de estados.
7. Diagrama de actividades.
8. Diagrama de componentes.
9. Diagrama de despliegue.

Luego en la versión **UML 2.0**, se agregan nuevos diagramas:<sup>15</sup>

1. Diagrama de clases.
2. Diagrama de objetos.
3. Diagrama de componentes.
4. Diagrama de estructura compuesta.
5. Diagrama de casos de uso.
6. Diagrama de secuencia.
7. Diagrama de comunicación.
8. Diagrama de estados.
9. Diagrama de actividades.
10. Diagrama de despliegue.
11. Diagrama de paquetes.
12. Diagrama de tiempos.
13. Diagrama de visión global de interacciones.

Los mismos creadores de la Notación UML, crearon el **Proceso Unificado de Desarrollo** donde es un Proceso de Ingeniería de Software con las siguientes características:

<sup>13</sup> <https://www.omg.org/spec/UML>

<sup>14</sup> El Lenguaje Unificado de Modelado. Manual de Referencia. (Booch, Rumbaugh, Jacobson)

<sup>15</sup> [https://es.wikipedia.org/wiki/Lenguaje\\_unificado\\_de\\_modelado](https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado)

- **Iterativo e Incremental:** El desarrollo se divide en cuatro etapas: Inicio, Elaboración, Construcción y Transición. Dentro de cada una de estas etapas, se hacen iteraciones con el objetivo de agregar mayor funcionalidad o agregar optimizaciones al producto final.
- **Dirigido por Casos de Uso:** Por cada iteración que se realiza en las etapas mencionadas se agrega un nuevo caso de uso, el cual será diseñado, codificado, implementado y probado.
- **Centrado en la Arquitectura,** existen diferentes alternativas de modelos y vistas que se utilizan para definir la arquitectura de software de un sistema.

A continuación se describe cada diagrama:

#### ❖ Diagramas de Casos de Uso

Un Diagrama de Casos de Uso describe cómo interactúa el actor o los actores con el sistema. Un actor es un rol de varios usuarios que utilizan el sistema. Un caso de uso es una funcionalidad dentro del sistema. Un caso de uso puede ser una especialización de un caso de uso, también puede ser que un caso de uso sea una extensión de otro caso de uso. Es decir, existen relaciones de asociación y extensión entre los casos de uso.

Los diagramas de casos de uso modelan la vista de casos de uso estática de un sistema. Estos diagramas son especialmente importantes en el modelado y organización del comportamiento de un sistema.

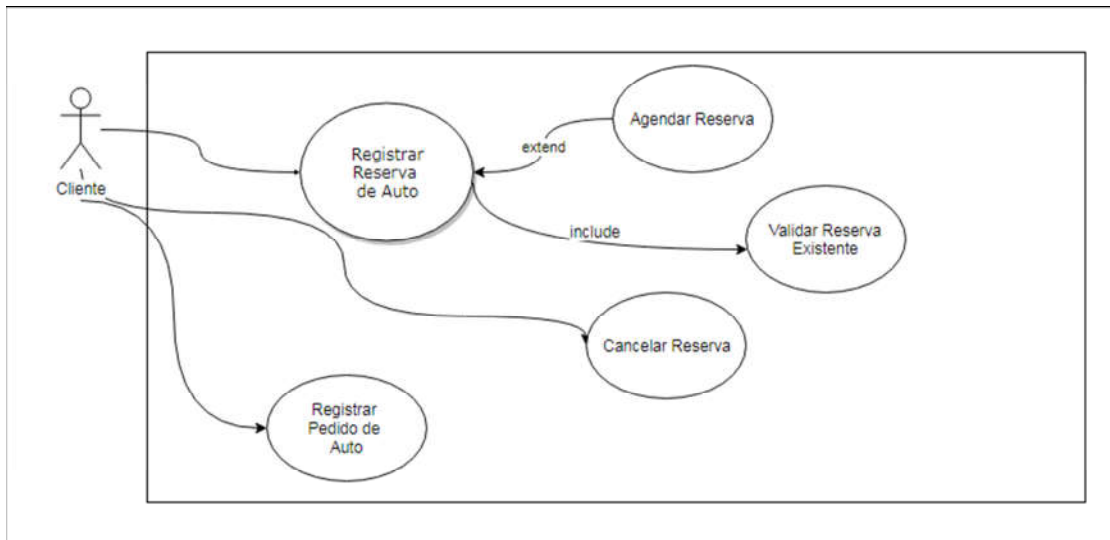


Figura 5: Diagrama de caso de uso con sus relaciones.

El Diagrama de casos de uso sólo muestra “qué” debe hacer el sistema, pero no dice “cómo” debe hacerlo.

Para esto se crean escenarios por cada caso de uso, donde se describe una secuencia de acciones que realizan el actor y el sistema, inclusive caminos alternativos que ejecuta el sistema para obtener información de salida requerida por el actor.

Caso de Uso: Registrar Reserva de Auto		Fecha: 20/10/18
Estado: Análisis		Tipo: Primario
Actor: Cliente		
Descripción: Cliente solicita la reserva de un auto		
Precondición: El usuario debe identificarse y autenticarse.		
Post-Condición: La reserva del auto queda agendada.		
Actor		Sistema
1- Ejecuta aplicación del Sistema	Muestra una pantalla que permite el ingreso de usuario y contraseña.	
2- Ingresa usuario y contraseña.	El sistema valida datos ingresados y da acceso al menú principal del sistema.	
3- Ingresa al Menú de Reservas	Despliega Menú y muestra opciones para ingresar día y horario de reserva.	
4- Selecciona día desde-día hasta, horario.	Muestra datos completos de los autos disponibles.	
5. Selecciona un auto y Confirma.	Llama a CU: Agendar Reserva.	
Caminos Alternativos		
2	Los datos ingresados por el usuario son incorrectos. El sistema muestra mensaje, "Usuario y Contraseña incorrecto"	
4	Visualiza un mensaje de error "Para estas fechas y horarios no hay autos disponibles"	
Importancia: Alta		
Comentarios: Ninguno		
Rendimiento Esperado: Optimo		

## ❖ Diagramas de Secuencia

Por cada caso de uso se crea un diagrama de secuencia para mostrar el comportamiento que existe entre los objetos del sistema, cada vez que el actor hace una petición al sistema.

Es una de las opciones de diagrama de interacción que muestra los objetos y sus interacciones en el tiempo por medio de mensajes que se representan con flechas que van desde la línea de vida origen hasta la línea de vida destino de cada objeto.

En forma vertical se representa la línea de vida de cada objeto. Los diagramas de secuencia son útiles para mostrar el orden secuencial en que se ejecutan los mensajes.

El UML soporta diferentes estereotipos, que son formas de extender o alterar el significado, la presentación y la sintaxis de un elemento del diagrama<sup>16</sup>

Ejemplo: para representar el Objeto Tareas, se utilizó el siguiente estereotipo.

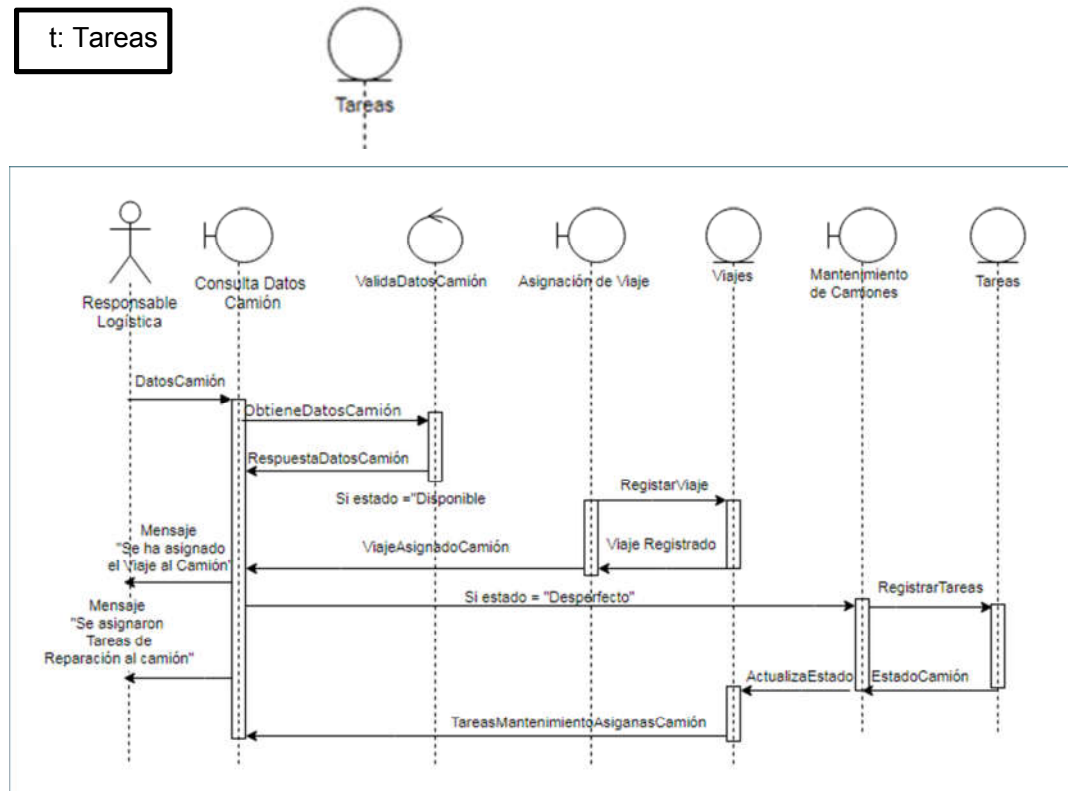


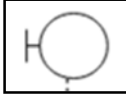
Figura 6: Diagrama de Secuencia.

En el diagrama se puede ver que el actor solicita datos al sistema, por medio de la Interfaz, el sistema internamente realiza controles, actualiza y obtiene datos almacenados en Tareas y Viajes (objetos entidad) y da respuesta al requerimiento del usuario.

<sup>16</sup> <http://www.sparxsystems.com.ar/EAUserGuide/index.html?receive.htm>. Especificación de OMG de la superestructura UML, v2.1.1



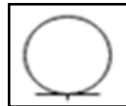
Cada objeto tiene una forma diferente de representación de acuerdo a lo que realiza en el sistema.



**Objeto Interfaz**, que permite interactuar al actor con el sistema. El actor envía datos por medio de un mensaje al Sistema usando la Interfaz de entrada.



**Objeto Control**, que realiza las validaciones y controles sobre los datos recibidos en el mensaje.



**Objeto Entidad**: representa los datos de almacenamiento.

Los mensajes que se envían los objetos, se pueden representar en dos tipos de diagramas de interacción, el diagrama de secuencia que destaca la secuencia de los mensajes en el tiempo y el diagrama de colaboración que enfatiza las relaciones entre objetos y los mensajes que se envían.

Los elementos claves para armar un diagrama de secuencia son los escenarios de los casos de uso, los objetos, los mensajes y los controles o validaciones que realice el sistema.

Es útil sólo cuando se representan casos de uso menores.

### ❖ Diagrama de Colaboración

Es un diagrama de interacción que muestra cómo se organizan los mensajes que intercambian los objetos. Es importante en este diagrama enumerar la secuencia de envío de mensajes, es decir mostrar el orden, cómo se organiza el envío de datos entre los objetos que describen la funcionalidad del caso de uso.

Se puede representar un caso de uso con varios diagramas de colaboración, utilizando paquetes para delimitar la funcionalidad.

### ❖ Diagrama de Actividades

Un diagrama de actividades muestra las actividades que realizan el actor que hace uso del caso de uso y las actividades que realiza el sistema para responder a la petición del usuario, las actividades siguen un curso controlado por flujos de control.

En cada columna se anotan las actividades que realizan el actor y el sistema. Se representa un nodo inicial y un nodo final, las actividades se unen con conectores para representar el flujo de tareas. Pueden intervenir elementos de bifurcación de tareas o bucles.

### ❖ Diagrama de Transición de Estados

Se utilizan para modelar comportamientos dinámicos del sistema. Así como el Diagrama de actividades muestra el flujo de control entre actividades, el diagrama de transición de estados, muestra cómo cambia de estado un objeto del sistema ante cada evento a lo largo de su vida.

### ❖ Diagramas de Clases

Es uno de los diagramas más importantes de UML porque representa la vista de diseño estática del sistema donde se muestran las clases, las interfaces, operaciones y las relaciones entre clases. Es imprescindible que esté completa su representación porque se utiliza también en Ingeniería Inversa.

Es donde se representan los datos que se requiere guardar de cada objeto del sistema. Por cada objeto se definen las operaciones o colaboraciones que realiza.

El Diagrama de Casos de Uso es una herramienta fundamental para armar este diagrama, dado que describe las funciones que realiza el sistema, y describe los datos que se envían entre el actor y el sistema.

### ❖ Diagrama de objetos

Es un diagrama de estructura que modela aspectos estáticos del sistema, donde se representan los objetos intervinientes y sus relaciones.

Un diagrama de objetos es una instancia de un diagrama de clases, muestra una imagen del estado del sistema en un momento determinado. Es la foto del diagrama de clases en un tiempo determinado.

### ❖ Diagrama de componentes

Los diagramas de componentes se utilizan para representar la estructura estática del sistema pero en relación a su implementación. Los elementos que se incluyen en el diagrama son la composición interna de una clase, las interfaces externas y los puertos.

### ❖ Diagrama de despliegue

Son diagramas que se utilizan para modelar y documentar como van a estar distribuidos los nodos del servidor donde se va a ejecutar el sistema.



## 2.4. Metodologías Ágiles

A diferencia de las Metodologías presentadas anteriormente donde se enfocaban en la documentación del análisis y diseño de sistemas previo a la programación, las metodologías ágiles restan tiempo dedicado a la planificación y documentación del análisis y diseño, poniendo toda la atención en una gestión de proyecto ágil basada en la comunicación permanente y eficaz con el usuario e introduciendo actualizaciones en forma continua y periódica en los sistemas.

Estos cambios se inician con los requerimientos del usuario donde tiene una participación más activa en el proceso de actualización del sistema.

El objetivo de esta Metodología es la dinámica del proceso y la respuesta rápida a las necesidades que el cliente plantea. Es justamente una de sus ventajas si el equipo de trabajo está bien logrado.

Es importante que exista una buena interacción entre los participantes del equipo, que todos estén mentalizados en la dinámica del proceso y tengan un conjunto de valores compatibles, un conjunto de valores basados en la confianza y el respeto mutuo.

Se busca trabajar con energía y motivación, con reuniones diarias de pocos minutos donde se plantean los requerimientos del usuario, se define el alcance y se planifica la próxima entrega o prototipo estableciendo un ciclo de entrega que le aporte valor al cliente, lo cual permite dar respuesta al cambio dando la atención permanente que necesita.

Cada cambio que solicita el cliente debe ser bien aceptado, porque el cambio que necesita es una ventaja competitiva en su trabajo.

Las entregas deben ser a corto plazo, en lo posible semanales, esto permite que el usuario vea plasmado en el sistema los cambios que está solicitando y lo motiva a seguir en el proceso de cambio.

Algunos métodos ágiles de desarrollo de software son: <sup>17</sup>

- Adaptive Software Development (ASD)
- Agile Unified Process
- Crystal Clear
- Feature Driven Development (FDD)
- Lean Software Development (LSD) : Lean startup
- Kanban (desarrollo)
- Open Unified Process (OpenUP)
- Programación Extrema (XP)
- Método de desarrollo de sistemas dinámicos (DSDM)
- Scrum
- G300
- 6D-BUM
- PMI Agile

<sup>17</sup> [https://es.wikipedia.org/wiki/Desarrollo\\_%C3%A1gil\\_de\\_software](https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software)

Para comprender como funcionan estas metodologías nos vamos a centrar en las tres metodologías ágiles más utilizadas en las empresas de Ingeniería de Software.

#### 2.4.1. SCRUM <sup>18</sup>

Existen conceptos específicos en esta metodología que se irán mencionando los más relevantes para comprender como se organizan en un proyecto de software.

Existe un **Scrum Master**, que tiene una visión global del proyecto, es quien representa el equipo de trabajo, y responsable de que el equipo funcione y tenga la dinámica que se requiere para cumplir con los **Sprints**.

Los **Sprints**, son los ciclos de entrega de producto que corresponde al tiempo de desarrollo de los programadores, estos plazos se establecen de acuerdo a diferentes variables como ser recursos disponibles o nivel de exigencia de los requerimientos prioritarios, pero preferiblemente se busca que sean de duración fija. Se establecen Sprints de 1 a 4 semanas como máximo para

no perder la dinámica que caracteriza a esta metodología. Cada Sprint tiene como resultado un entregable que se realiza al cliente.

El cliente o **Product Owner**, es quien prioriza los requerimientos que se van a incluir en el próximo sprint.

El **Team Member**, responsable de desarrollo del producto.

SCRUM es de gran aporte cuando no se está cumpliendo con lo que solicita el cliente, cuando los costos de desarrollo son altos, cuando la competencia del equipo es baja o en casos de equipos de trabajos con perspectivas dispares.

SCRUM es una metodología de trabajo colaborativo, donde se basa en el trabajo en equipo y se busca obtener resultados a corto plazo y en forma permanente.

#### 2.4.2. KANBAN <sup>19</sup>

No es una metodología de Análisis y Diseño, es una metodología complementaria a SCRUM, porque se basa en optimizar los flujos de trabajo, donde se definen y clasifican las tareas pendientes de realizar, las tareas en curso y tareas terminadas.

El usuario solicita un requerimiento para realizar un cambio en el sistema, y el analista funcional quien está en contacto permanente con el cliente ingresa un ticket en una herramienta como KANBAN. En el mercado se lo conoce como KANBAN Tools o Tablero Kanban.

<sup>18</sup> <https://proyectosagiles.org/como-funciona-scrum/>

<sup>19</sup> <https://kanbantool.com/es/metodologia-kanban>

Existen muchas herramientas que permiten hacer un seguimiento de tickets que son asignados a un responsable de realizarla. Cada responsable ingresa a la herramienta, donde visualiza las tareas que tiene asignadas y comienzan a desarrollarlas.

Tienen la posibilidad de agregar observaciones o comentarios para comunicar el estado de la tarea a los demás integrantes del grupo de trabajo, se pueden adjuntar especificaciones relacionadas a los requerimientos del cliente y determinar en qué grado de avance se encuentran.

En la actualidad hay muchas herramientas online que permiten llevar este seguimiento de un proyecto de Ingeniería de Software, en donde accede el cliente para hacer sus requerimientos sobre el sistema y son visibles por todos los integrantes del proyecto a cargo del desarrollo e implementación.

Es una forma de distribuir y asignar tareas, pero no define un estándar de cómo se lleva a cabo esa tarea. Si hay documentación quedará distribuida por tickets o requerimientos, y puede ser que no exista una documentación sobre aspectos funcionales o estructurales del sistema.

#### **2.4.3. XP o Programación Extrema**

Es una metodología de trabajo que pone énfasis en requerimientos de pocas horas de desarrollo, se hacen entregas al cliente y se espera una aceptación de parte de ellos. El objetivo es mejorar la calidad del producto, es una metodología complementaria a SCRUM donde se busca mayor productividad.

El objetivo de esta metodología es enseñar las mejores prácticas de Ingeniería de Software en el desarrollo de proyectos basado en un proceso iterativo e incremental.

Se arman grupos de pocos integrantes donde el objetivo es aumentar la productividad, cumplir con requerimientos del cliente y certificar los estándares de calidad del producto con pruebas continuas.

Es una metodología basada en la realización de casos de prueba antes de desarrollar el código. La tarea de programación se recomienda que sea compartida, permite hacer revisiones y el desarrollo de una lógica de programación simple que sea igualmente interpretada por los responsables del código, se analiza en forma grupal la resolución del mismo mejorando la calidad del producto final.

El cliente o el usuario tienen mayor participación en las pruebas y tareas de desarrollo.

Al ser una metodología basada en prueba y error se hacen optimizaciones en forma permanente, garantizando no introducir nuevos errores y mantener el sistema estable.

### 3. Conclusión sobre las Metodologías Presentadas

Las Metodologías Ágiles definen una forma de trabajo, si bien tiene sus ventajas porque organiza las entregas en plazos cortos, le da dinámica al trabajo de desarrollo de software y mejora la calidad del producto con optimizaciones en forma continua, es una metodología que no menciona diagramas para documentar en las etapas de análisis y diseño, no es una notación como UML y no tiene un estándar establecido para documentar.

Es muy arriesgado trabajar a largo plazo sin documentación, porque los integrantes de los proyectos cambian permanentemente, y los sistemas perduran en el tiempo con la necesidad de ir adaptándolos a los cambios requeridos por el cliente.

Al no existir continuidad de los integrantes de un proyecto y no existir documentación que describa los procesos que se desarrollaron en un sistema, esa información a futuro no estará disponible lo que implica que los próximos responsables de ese mismo producto tengan que realizar lecturas de código realizadas por otro programador para interpretar la lógica del proceso y llevar a cabo la tarea de reingeniería de software que implicará más horas para adaptarlo a los cambios de negocio o correcciones de errores que se originen.

La desventaja de la programación extrema es que se debe contar con profesionales con experiencia. Cuando el equipo de trabajo recién se inicia o no tiene experiencia, no es posible cumplir con entregas al cliente en el corto plazo, porque para llevar a cabo estos cambios cuando no se tiene conocimiento de su funcionalidad, o de la lógica de los procesos y de las estructuras sobre las que están programados, se convierte en una tarea que insume más tiempo con un alto costo en horas de análisis y programación.

Para implementar este tipo de metodologías en una empresa, se requiere personal de sistemas con ciertas habilidades comunicacionales y orientado a la atención al cliente, dado que la base de la optimización del producto y el éxito del proyecto en las metodologías ágiles es la comunicación fluida con el usuario.

El perfil del usuario también debe ser de las mismas características, que sepa expresar y transmitir cuáles son las necesidades del cambio que requiere del sistema para realizar las mejoras y hacer un feedback lo antes posible después de cada nuevo ejecutable que recibe. Esto agiliza el proceso de iteración entre las tareas de análisis, programación y testing con la finalidad de implementar la mejora en el sistema en el servidor de producción lo antes posible.

Esta metodología requiere de cierta exigencia hacia los programadores lo cual no beneficia a una empresa si el personal no se adapta a la forma de trabajo. Lo ideal es que realice tareas diversas para modificar el ritmo de trabajo, llevándolo a tareas más livianas en cuanto a esfuerzo, para incorporarlo nuevamente a la programación extrema.

Además de las habilidades comunicacionales entre los integrantes del equipo debe existir también un ritmo de trabajo similar en forma individual. No es beneficioso si un integrante trabaja a destiempo no cumpliendo con plazos cortos de entrega de ejecutables, pues se pierde la esencia de esta metodología, desmotivando a los demás integrantes del grupo.

En el caso de las Metodologías Tradicionales, como el Análisis y Diseño Estructurado es una forma de trabajo que se ha dejado de usar hace tiempo, dado que requiere un alto costo en las tareas de documentar el análisis y diseño y además no es flexible a los cambios, porque ejecuta una etapa por única vez y no permite volver a etapas anteriores como es el caso del Modelo en Cascada.

Luego en el Modelo de Prototipos, se introduce una mejora de trabajo al permitir volver a etapas anteriores para analizar y rediseñar, pero requiere muchas horas de mantenimiento de la documentación, por la gran cantidad de diagramas propuestos en cada modelo.

Es difícil separar el concepto de Metodología de desarrollo con el concepto de documentación de análisis y diseño. Si bien la propuesta de esta tesis es proponer una forma de documentar el análisis y diseño, no podemos dejar de mencionar que son tareas dentro una metodología de desarrollo de software y que requiere adaptarla a las metodologías ágiles utilizadas hoy en día, que buscan las empresas para mejorar la productividad.

La documentación no debe ser una tarea que implique mucho tiempo con extensos diagramas difícil de mantener.

Hay documentación de procesos que debe existir expresada en pseudocódigo o lenguaje natural.

Muchos programadores tienen como hábito expresar en lenguaje natural una breve descripción de la funcionalidad, antes de iniciar las líneas de código para desarrollar un proceso, lo cual puede ser beneficioso para interpretar un código de programación realizado por otro integrante del equipo, pero esto queda librado a la forma de trabajo de cada programador y no se podría utilizar como un estándar de documentación, además insume más horas de programación y mayor esfuerzo para el programador. Esto facilita la lectura e interpretación del mismo a otras personas que acceden al código para reprogramar, a los analistas para realizar las tareas de análisis y definiciones e incluso al mismo autor del código que a futuro le es difícil interpretar que programó en ese momento.

Dado que la lógica de programación se basa en la lógica de razonamiento, no todos los programadores resuelven un mismo requerimiento de la misma forma, se pueden ver algoritmos diferentes para dar solución a un mismo caso de uso del sistema.

La lógica de razonamiento es un conjunto de actividades mentales que permiten desarrollar una idea. Si las tareas de análisis, programación y testing de un requerimiento no se realizan en equipo en forma conjunta y se desarrollan como tareas individuales, insume mucho tiempo interpretar la lógica de otro integrante del equipo sin tener un documento que describa el proceso del requerimiento.

Existen muchos lenguajes de programación que generan en forma automática los diagramas y la documentación, pero no en lenguaje natural.

La programación neurolingüística o PNL, junto a la inteligencia artificial han logrado captar lenguaje natural a medida que se expresa, este lenguaje es procesado por un software y luego se vuelca en un computador. Es una actividad informática que automatiza cualquiera de las formas de la comunicación verbal, ya sea transmitida por telefonía o cualquier otro equipo preparado para esto.

El proceso inverso, es decir expresar lenguaje de código en lenguaje natural, sería una alternativa de documentación, aunque puede generar ambigüedades en la interpretación.

Es por eso que muchos lenguajes de programación generan en forma automática diagramas para representar la lógica de programación con estándares establecidos, pero no todos los lenguajes de programación trabajan de la misma forma.

El avance de la tecnología y la educación en el área de la psicología y la inteligencia artificial, permitieron logros importantes en la captura de pensamientos o actividad cerebral y volcarla en imágenes, o la captura de voz y volcarla a lenguaje natural o la lectura de labios y volcarla en lenguaje natural.

Es posible hoy en día automatizar una conversación entre analistas y el usuario que trabajan en la definición de un requerimiento, generando en lenguaje natural un documento con las definiciones del requerimiento. Pero se encuentra pendiente expresar en lenguaje natural, el código generado por un programador como solución a un requerimiento.

Rescato algunas ventajas de cada metodología que voy a tomar como propuesta para presentar en esta tesina.

## 4. Propuesta

Antes de definir una propuesta de documentación para el análisis y diseño de sistemas, vamos a definir dos Metodologías de trabajo en la Ingeniería de Software.

Si estamos frente a un Sistema que ya existe, es decir, se encuentra implementado en el cliente y en el cual se introducen procesos nuevos en forma permanente de acuerdo a las nuevas reglas de negocio o se realizan optimizaciones para mejorar el uso, se requiere una Metodología de Ingeniería de Software diferente frente al caso de Implementar un nuevo Sistema que no se encuentra productivo.

### 4.1. Metodología de Ingeniería de Software para un Sistema en Producción

Basándonos en las Metodologías Agiles, en especial en XP (Programación Extrema), se propone introducir nuevos cambios al sistema a partir de Casos de Prueba por Proceso.

En el caso de prueba, se debe mostrar con pantallas, los datos previos (parámetros) que deben estar en el sistema para que el proceso funcione en forma óptima, la secuencia de pasos que realiza el usuario hasta lograr los resultados obtenidos y definir los resultados esperados. Cuando los resultados esperados difieren de los resultados obtenidos, se define el requerimiento con el cambio a realizar en el proceso. Es simplemente mostrar la secuencia de pantallas que ejecuta un usuario del sistema, para mostrar su funcionamiento y describir el resultado que se obtiene, ya sea con error o resultado exitoso. Si el resultado obtenido no es el resultado esperado, se debe expresar que se deseaba conseguir.

Este caso de prueba, puede ser creado por el analista funcional o por el mismo usuario y debe ser validado por el usuario.

Luego este requerimiento se ingresa en una herramienta de seguimiento de tickets o requerimientos, donde está a la vista de todos los integrantes del proyecto.

Al requerimiento se le asigna una estimación de horas aproximadas, en donde se espera la aceptación del usuario.

Cuando el usuario acepta el requerimiento con su respectiva cotización, se asigna la tarea de programación a un responsable del área de desarrollo de software.

El programador inicia la tarea de programación, realiza los cambios en los procesos y finaliza la tarea en la herramienta de seguimiento de tickets y se deja asentado la cantidad de horas reales que llevó realizar el requerimiento, lo que se traduce en el costo que deberá enfrentar el cliente para implementar el requerimiento en el Servidor de Producción.

Luego se realizan las pruebas en un servidor local (de la Empresa que desarrolla el software), si las pruebas no arrojaron los resultados esperados se revisan los procesos nuevamente.

Si las pruebas fueron satisfactorias, se implementan en el servidor de prueba del cliente en donde el usuario y los analistas funcionales realizan las pruebas en forma conjunta hasta aprobar su implementación en el servidor de producción.

Se elabora una documentación técnica especificando los cambios realizados en el proceso que será entregado al cliente para comprender la lógica del mismo, este documento lo ayudará a realizar las pruebas correspondientes y validar el requerimiento solicitado.

Hasta aquí no se menciona nada nuevo en relación a la forma de trabajo que tienen la mayoría de las Empresas de Ingeniería de Software. Quizás no se siga en forma exacta los pasos mencionados, pero es la forma en general que utilizan para introducir nuevos cambios en el sistema que se encuentra implementado y que se encuentra en análisis.

En esta metodología se propone la **documentación de los casos de prueba** para generar el requerimiento y la **documentación técnica del proceso modificado**, que más adelante se describe la forma de realizarlo.

Para agilizar las tareas de análisis, programación, documentación y testing algunas empresas dividen los procesos del sistema y se asignan a un grupo de responsables.

Cada vez que surge un cambio en un proceso determinado ya sea para cualquier cliente, siempre son los mismos responsables de generar el cambio en ese proceso.

Esto es muy común cuando se trata de sistemas que abarcan mucha funcionalidad y se requiere conocimiento especializado en ciertas áreas de negocio.

A continuación se describe el contenido que se propone en la **Documentación Técnica para la implementación de un proceso**:

**(1) Parámetros previos a la ejecución del proceso.**

- a. Los parámetros previos, se refieren a datos que deben estar previamente cargados en una tabla, para que el proceso pueda leerlos y tenga una ejecución normal.
- b. Parámetros Guiados. Es habitual utilizar guías de procesos. Estas guías de procesos, son datos cargados en una tabla. Si estos datos no se cargan el proceso funciona normalmente obteniendo el resultado esperado, pero si esta guía de proceso se carga con ciertos datos, se puede lograr que el proceso tenga un funcionamiento diferente, inclusive hasta llamar a pantallas diferentes. Es muy común utilizar esto cuando un mismo sistema se encuentra implementado en varios clientes y se pretende que tenga comportamientos diferentes.



**Ejemplo de Guía de Proceso o Tabla de Parámetros Previos:**

Para que el Proceso “MantenimientoCamion” funcione en forma correcta se debe tener cargada la siguiente tabla de parámetros, en donde el Cliente1 solicita que el Mantenimiento tenga presupuestos que se actualicen con moneda dólar. Para tal fin al Cliente1, se le debe parametrizar el Valor MantenimientoCamionD, esto indica que el proceso MantenimientoCamion, llamará al proceso MantenimientoCamionD, para lograr una actualización de presupuesto en moneda dólar.

Si el día de mañana, el Cliente1 cambia su regla de negocio, y requiere que los presupuestos de Mantenimiento de Camiones se actualicen con el índice de Inflación, en el campo Valor, para el Cliente1, se indica MantenimientoCamionI.

El proceso debe estar preparado para leer la Funcion\_ID: 1234, y esta parametrización debe estar indicada en un documento técnico, para mostrar cómo se comportará el sistema.

Si la tabla de Parámetros no tiene cargada la Función\_ID:1234, el proceso funcionará en forma correcta pero solo se podrá realizar actualizaciones de presupuesto en forma manual de acuerdo a lo que ingrese el operador en los precios de Insumos de camiones.

Funcion_ID	Proceso_Id	Par_Id	Valor	Descripción1
1234	MantenimientoCamion	cliente1	MantenimientoCamionD	Módulo Mantenimiento de Camiones para cliente 1 con presupuesto que actualiza con moneda Dólar
1234	MantenimientoCamion	cliente2	MantenimientoCamionI	Módulo Mantenimiento de Camiones para cliente 2 con actualización por Índice de Inflación

**Ejemplo de Tabla de Parámetros o Guía de Proceso****(2) Punto de menú desde donde se ejecuta.**

Se debe indicar la ruta para llegar a ejecutar la funcionalidad que solicitó el usuario. Si el punto de menú no existe porque el objeto es nuevo, se debe indicar nombre del objeto para agregarlo al punto de menú.

Ejemplo: Si el usuario solicitó un Listado de Camiones, se debe indicar el punto de menú para obtenerlo y el nombre del objeto que se debe agregar al menú como una opción más para que el usuario pueda acceder. Supongamos que el nombre del objeto es RCamionCC.

**Punto de menú para acceder al Reporte “Listado Camión, Costo, Consumo”**

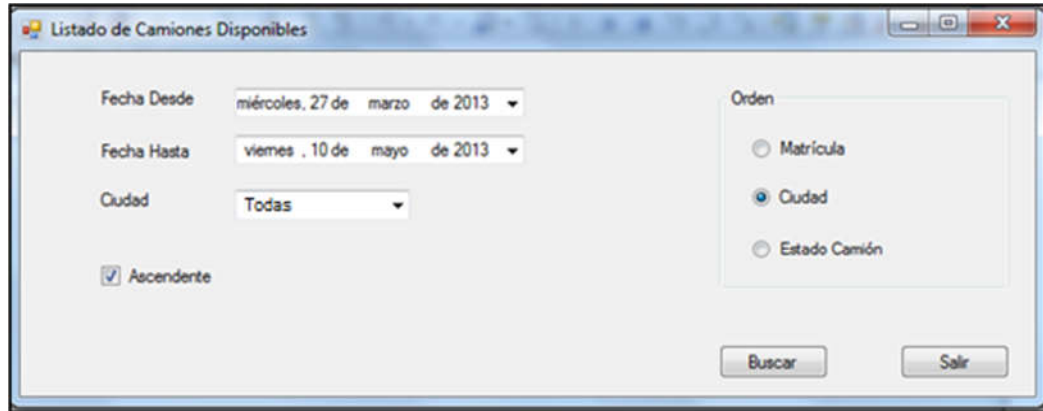
Camiones→Consumo Camiones→ Listados→Listado Camión, Costo, Consumo

**Objeto a agregar al menú:**

RCamionCC

### (3) Pasos a tener en cuenta en la ejecución.

Se puede mostrar una secuencia de pantallas o datos a ingresar en las pantallas o mostrar botones que se deben presionar para la ejecución de los procesos. O mostrar el caso de prueba que realizó el analista funcional para validar el correcto funcionamiento.



Listado de Camiones Disponibles			
Fecha Desde	27/03/2013		
Fecha Hasta	10/03/2013		
Ciudad	Todas		
Matricula	Camionero	Consumo	Costos de Mantenimiento
AA 123 KR	Carlos Perez	32.987	3.600
AB 543 YL	Ramiro López	34.112	5.690
AD 145 HY	Mauro R.	7500	20556

### (4) Lógica de procesamiento.

La lógica de procesamiento es una breve descripción de las tablas que recorre el proceso, que datos obtiene y validaciones que realiza.

### (5) Descripción del resultado (reportes de salida, estado o modificaciones que deben quedar en las tablas de la base de datos)

Muchas veces un proceso realiza actualizaciones en la base de datos y no se visualiza por pantalla y tampoco muestra ningún reporte que advierta de este cambio.

Es por eso que se requiere que se exprese la lógica de procesamiento y se detalle que datos actualiza, en que tablas y bajo qué condiciones.

#### (6) Listado de tablas involucradas.

En caso de omitir los pasos 4 y 5, se puede detallar un listado de tablas involucradas en el proceso y aclarar si obtiene datos o si actualiza y que campos.

La tarea de documentar los ítems 2, 3, 5 y 6 es una tarea que puede realizar el analista funcional mientras realiza las pruebas, con la colaboración del programador del proceso.

El ítem 1 y 4, es una tarea que debe realizar el programador mientras realiza el proceso de programación, agregando comentarios en las líneas de código, lo que facilitaría posteriormente crear con lenguaje natural la lógica del proceso.

### 4.2. Metodología de Ingeniería de Software para un Sistema nuevo a Implementar

En principio se define una forma de trabajo propuesta para la Ingeniería de Software de un Sistema nuevo a implementar, que actualmente muchas Empresas de software la utilizan hoy en día.

Esta forma de trabajo para la construcción de software, la vamos a dividir en etapas:

- **Relevamiento**, donde la documentación de esta etapa es la **Lista de Requerimientos** y **Diagramas de Actividades** para definir procesos de Negocio (Workflow). BPMN ó Modelo de Notación de Procesos de Negocio. Responsables de esta tarea, analistas funcionales y usuarios.
- **Definición de Procesos Prioritarios**, donde la documentación en esta etapa es la **Especificación de Casos de Uso**. Responsables de esta tarea, analistas funcionales.
- **Asignación de Casos de Uso a Programadores**. Para hacer un seguimiento del requerimiento se puede utilizar alguna herramienta online donde todos los integrantes del proyecto tengan acceso para ver el avance de las tareas. Ejemplo Kanban Tools o Mantis Bug Tracker, son propuestas diferentes.
- **Programación**, generación de código y documentación técnica que describa el proceso. Esta documentación técnica muchas veces puede ser solicitada por el cliente, para comprender la lógica de los procesos. Es una tarea que puede realizar el programador a medida que genera el código. Algunos clientes lo exigen para cumplir ciertas normativas (ejemplo Entidades Bancarias).
- **Pruebas en Servidor de Desarrollo**. Una vez finalizada la actividad de programación, se realizan las pruebas en un servidor local antes de enviarlo al servidor del cliente

para verificar que no existan errores de ejecución y que cumpla con las definiciones del cliente. Responsables de esta tarea, analistas funcionales y programadores.

- **Pasaje a Servidor de Testing y Pruebas.** Cuando se realiza el pasaje al servidor de Testing del cliente, se puede enviar como documentación un caso de prueba y en caso que lo soliciten la documentación técnica. Luego se comienzan las pruebas, lo cual es una tarea compartida entre el analista funcional y el usuario. Se espera un feedback lo antes posible del cliente para realizar la puesta en producción del requerimiento.
- **Pasaje a Servidor de Producción.** Se comunica al cliente ya sea por mail o por medio de la herramienta online donde se hace el seguimiento de tickets, avisando que se ha realizado una nueva actualización en el servidor de producción, detallando los requerimientos resueltos.

#### 4.2.1. Relevamiento

En la etapa de relevamiento, los profesionales de sistemas realizan reuniones en forma permanente con los usuarios de distintos sectores de la Empresa donde se va a implementar el software.

En estas reuniones se busca definir y analizar los procesos prioritarios que se requieren para la salida a producción del sistema. Algunos de estos procesos ya existen en el Core que se ofrece como solución y otros procesos deben ser desarrollados especialmente para cada cliente para cumplir con sus necesidades de negocio.

Para determinar los procesos que se van a incluir en el sistema, el usuario define qué información necesita ingresar al sistema, reglas de negocio y circuitos administrativos, en base a estos datos, se propone una lista de requerimientos o casos de uso.

Esto nos permite ver las opciones de menú que necesita el usuario para hacer uso del sistema.

Ejemplo de **Lista de Requerimientos** tomados del Sistema de Transporte de Cargas.

- Administración ingresa datos del cliente.
- Administración ingresa datos de camión.
- Administración ingresa datos de camionero.
- Administración ingresa datos de viajes frecuentes.
- Responsable de Logística solicita informe de estado de cada camión.
- Responsable de Logística genera orden de mantenimiento del camión.
- Cliente realiza pedido de un viaje.
- Diariamente, generar un informe de pedidos.
- Responsable de Logística solicita listado de camiones disponibles.

- Responsable de Logística asigna el pedido de viaje de acuerdo al destino.
- Administración envía al responsable de logística la verificación de estado de camión y camionero.
- Responsable de Logística contrata servicios de camiones externos a la empresa.
- Camionero entrega remito.
- Camionero realiza rendición de gastos de viajes.
- Dpto. Contaduría solicita valorización de los viajes (tarifario) para facturar al cliente.
- Dpto. Contaduría solicita emisión de facturas electrónicas.
- Dpto. Contaduría solicita listado de facturas impagas.
- Dpto. Contaduría solicita listado de proveedores.
- Dirección General solicita costo por camión y por insumo.
- Mensualmente, generar Informe de pedidos pendientes.
- Dpto. Logística solicita informe de consumos por cada camión.

Si cada proceso o caso de uso involucra varios sectores de la empresa, se puede mostrar cuales son las actividades que lleva a cabo cada sector y quienes son los usuarios responsables.

Para definir los circuitos de negocio o administrativos, se crean **diagramas de actividades** que describen la secuencia de las tareas que se desarrollan, en donde se pueden ver también los sectores intervinientes. En este caso, no innovaría sobre lo propuesto por UML en relación a su notación.

#### 4.2.2. Definición de Procesos Prioritarios

Se crean **documentos especificando los casos de uso** que necesita el usuario como prioridad para hacer uso del sistema, donde en lugar de usar la notación que propone UML, se propone una documentación diferente del caso de uso de una forma más práctica y a mi criterio más fácil de mantener a futuro.

Inclusive al ser una documentación sin notación especial como la notación UML que requiere de experiencia en su aplicación, esta propuesta de documentación se puede entregar al cliente para su validación en donde no tendrá dificultades para su comprensión.

Mi propuesta de Documentación de Caso de Uso, sería la siguiente:

Se definen los objetos o transacciones que van a definir la estructura de datos, utilizando una nomenclatura estándar para especificar los atributos.

Se analizan los diferentes datos que se ingresan al sistema, diseño de la interfaz de entrada de datos, formato de los tipos de datos, validaciones sobre los datos que se ingresan y validaciones en eventos transaccionales (Confirmar, Salir, Grabar, etc.)

Por ejemplo para el requerimiento “Alta de un cliente”, el caso de uso sería de la siguiente forma:

<b>Caso de Uso: Alta de un Cliente</b>			<b>Fecha: 20/10/18</b>
<b>Estado:</b> Análisis			Tipo: Primario
<b>Actor:</b> Responsable de Administración			
<b>Descripción:</b> Responsable de Administración Ingresar Datos del cliente.			
<b>Parámetros Previos a la ejecución:</b> Deben existir datos en las tablas Localidades, Códigos Postales, Categoría IVA y Tipos de Documento.			
<b>Resultado Esperado:</b> Debe quedar registrada el alta del nuevo cliente con un código de cliente generado en forma automática por el sistema.			
<b>Descripción Interfaz de Entrada, Proceso, Validaciones e Interfaz de Salida</b>			
<p><b>Desde el punto de menú Clientes, debe existir una pantalla “Mantenimiento de Clientes”,</b> que permita dar de alta un nuevo cliente, modificarlo y eliminarlo.</p> <p>Los datos a Ingresar en la Transacción “Alta de cliente” son los siguientes:</p>			
Campo	Tipo de dato	Descripción	Validaciones a nivel de campo
ClienteCod*	Numeric(6)	Código de Cliente	Se genera en forma automática, es único y secuencial.
ClienteNom	Character(60)	Nombre de Cliente	Ingreso Obligatorio
ClienteApe	Character(60)	Apellido de Cliente	Ingreso Obligatorio
ClienteTipDoc	Character(5)	Tipo de Documento del Cliente	Ingreso Obligatorio. Obtener de tabla Tipos de códigos de documentos. En caso de ingresar un dato que no existe en la tabla Tipos de códigos de Documentos, mostrar mensaje de error.
ClienteDoc	Numeric(8)	Documento del cliente	Ingreso Obligatorio
ClienteFchNac	Date()	Fecha Nacimiento de Cliente	
ClienteCatIva	Numeric(2)	Categoría IVA	Obtener Código de tabla Categoría de IVA. Ingreso Obligatorio. En caso de ingresar un dato que no existe en la tabla Categoría de IVA, mostrar mensaje de error.
ClienteDom	Character(60)	Domicilio del Cliente	
ClienteCodPos	Numeric(8)	Código Postal del Cliente	Obtener Código Postal de Tabla Códigos Postales. Validar que exista en la tabla el dato ingresado.
ClienteLocal	Character(60)	Localidad del Cliente	Se completa después de ingresar Código Postal
ClienteTel	Character(14)	Teléfono de Cliente	
ClienteTelOficina	Character(14)	Teléfono de Cliente en la Oficina	
ClienteCodVia	Numeric(8)	Código de Viajes Frecuentes del Cliente	Ingreso Opcional. Obtener el código de Viaje de la Tabla Viajes. Debe permitir ingresar más de un viaje.

### Validaciones en Evento Confirmación.

**En la Confirmación de la Transacción**, el sistema debe validar que no se haya ingresado anteriormente el número de documento. Si el documento ya existe, el sistema debe mostrar un mensaje indicando el código de cliente correspondiente.  
Esta validación podría hacerse también en el momento en que se carga el documento.

### Tablas que Graba el Proceso.

Después de ingresar y Confirmar los datos, quedarán grabadas las tablas Clientes y ViajesClientes.

Tabla CLIENTES		
Campo	Tipo de dato	Descripción
ClienteCod*	Numeric(6)	Código de Cliente
ClienteNom	Character(60)	Nombre de Cliente
ClienteApe	Character(60)	Apellido de Cliente
ClienteTipDoc	Character(5)	Tipo de Documento del Cliente
ClienteDoc	Numeric(8)	Documento del cliente
ClienteFchNac	Date()	Fecha Nacimiento de Cliente
ClienteCatIva	Numeric(2)	Categoría IVA
ClienteDom	Character(60)	Domicilio del Cliente
ClienteCodPos	Numeric(8)	Código Postal del Cliente
ClienteLocal	Character(60)	Localidad del Cliente
ClienteTel	Character(14)	Teléfono de Cliente
ClienteTelOfic	Character(14)	Teléfono de Cliente en la Oficina

Tabla VIAJESCLIENTES		
ClienteCod*	Numeric(6)	Código de Cliente
ClienteCodVia	Numeric(8)	Código de Viajes Frecuentes del Cliente

Como se puede ver el proceso graba dos tablas. Como en el último campo ClienteCodVia, estamos indicando que se pueden ingresar más de un viaje para un mismo cliente, requiere aplicar Normalización llevándolo a 1° Forma Normal.

Hay lenguajes de programación hoy en día, que al ingresar una transacción, hace la normalización de tablas en forma automática, como es el caso de Genexus. Con lo cual no sería necesario indicar cuales son las tablas que graba el proceso ya que queda implícito en la transacción.

Tomé como Nomenclatura Estándar, la nomenclatura GIK para la notación de los campos de la transacción que fue propuesta en la Comunidad de Genexus.<sup>20</sup>

<sup>20</sup> <https://wiki.genexus.com/commwiki/servlet/wiki?1872.GIK>,

### Ejemplo de Nomenclatura GIK

Objetos	Categorías	Calificador	Complemento
Cliente	Cod		
Cliente	Nom		
Cliente	Fch	Ini	
Cliente	Fch	Fin	
Cliente	Fch	Ing	Banco
Factura	Vta	Nro	
Factura	Cmp	Nro	

- **Objeto:** Es el nombre de la transacción a la que pertenece el atributo.
- **Categoría:** Es la categoría semántica del atributo.
- **Calificador:** Puede existir uno o dos calificadores
- **Complemento:** Texto libre

Si el caso de uso fuese “Generar un Reporte de Clientes”. Cuando se especifique el caso de uso, en el apartado **<Descripción Interfaz de Entrada, Proceso, Validaciones e Interfaz de Salida>**, se debe mostrar un Modelo de Pantalla de Entrada de datos, con campos que serán utilizados como filtros para obtener el reporte, se debe indicar los campos que se quieren mostrar en el reporte, definir el orden y agrupación en caso de ser necesario.

Hasta aquí hemos propuesto como documentar los requerimientos, una forma diferente de presentar los casos de uso.

#### 4.2.3. Asignación de Casos de Uso a Programadores.

Para hacer un seguimiento del requerimiento se puede utilizar alguna herramienta online donde todos los integrantes del proyecto tengan acceso para ver el avance de las tareas.

#### 4.2.4. Programación, generación de código y documentación técnica.

Esta documentación técnica puede ser solicitada por el cliente en cualquier momento, más aún si fueron procesos desarrollados con anterioridad en donde no tuvieron una participación activa en la definición del requerimiento. Es la misma documentación técnica que se propone en el punto 4.1

Es necesario que exista esta documentación para comprender como funcionan internamente los procesos, permite a los analistas comprender su lógica de programación y poder realizar las tareas de análisis. También permite al analista dar soporte al usuario cuando hace consultas sobre la funcionalidad del proceso. Esto hace que el usuario use el sistema, que es justamente el objetivo que busca una Empresa que desarrolla un Software, si el usuario no





comprende como es el funcionamiento de un módulo del sistema, puede que no se involucre y deje de usarlo o que lo use de forma incorrecta y genere errores o inconsistencias en los datos, lo cual es algo que debemos evitar.

Para comprender la lógica de los procesos, se puede describir en lenguaje natural un proceso. Si bien puede parecer una tarea pesada para el programador, podría ser más simple de lo que parece y más aún si es una tarea compartida con el analista funcional.

Esta técnica de documentación sería mayormente aplicable al lenguaje procedural, no tiene mucho sentido aplicarlo en eventos del sistema excepto que sea necesario, como confirmar una carga de datos en una transacción, ya que las validaciones en estos casos se muestran por pantalla.

En las pantallas donde hay procesos con mucha lógica de programación, se puede ir agregando notas descriptivas en el código, lo más resumida posible, indicando lo que hará el proceso que se está por programar. Anteponer a varias líneas de código un comentario. Indicar porque se accede a una tabla, si es para validar datos, indicar que se está validando. Estos comentarios se pueden agregar en cada acceso que se hace a una tabla, para obtener datos con ciertas condiciones o para validar datos con ciertas condiciones.

Agregar comentarios indicando “si se cumplen ciertos parámetros, se accede a una tabla para obtener determinados datos”

Es una tarea que puede realizar el programador a medida que genera el código.

Luego estos comentarios se pasan a un documento de Word, en donde se termina de describir la funcionalidad del proceso.

#### **4.2.5. Pruebas en Servidor de Desarrollo.**

Una vez finalizada la actividad de programación, el analista funcional comienza con las pruebas en el servidor local. Del resultado de las pruebas, se tomará la decisión de subir el ejecutable al servidor de prueba del cliente o continuar haciendo ajustes de programación.

#### **4.2.6. Pasaje a Servidor de Testing y ejecución de las Pruebas.**

Cuando se realiza el pasaje al servidor de Testing del cliente, se comienzan con las pruebas, esta vez con la participación del cliente en donde se trabajará en forma conjunta en ultimar detalles y definiciones en relación al funcionamiento del nuevo requerimiento. Se espera una respuesta lo antes posible del cliente para realizar la puesta en producción del requerimiento.



## 5. Conclusión

Cada metodología empleada en el desarrollo de software surgió de las rutinas de trabajo que los profesionales realizaban en sus Empresas. Los autores más conocidos que enseñan sobre Análisis y Diseño de Sistemas han expresado sus propias experiencias en libros y publicaciones sobre cada metodología que han utilizado.

A lo largo del tiempo estas metodologías fueron variando de acuerdo a las necesidades del mercado y a los cambios tecnológicos que lo seguirán haciendo, obligándonos a realizar un proceso de adaptación. Esas metodologías propuestas por los primeros autores fueron implementadas a largo del tiempo en distintas Empresas de la Industria del software que las han incorporado a sus rutinas de trabajo con sus variantes y a partir de estas prácticas fueron surgiendo otras metodologías.

Actualmente lo último que surgió como modo de trabajo en las empresas son las metodologías ágiles, de la cual rescato la forma en que se organiza el trabajo, la comunicación fluida con el cliente, su participación para definir los requerimientos prioritarios y los tiempos cortos de entrega de los ejecutables.

Es por eso que en mi propuesta propongo XP como propuesta para iniciar los requerimientos, donde el disparador son los casos de prueba que se realizan en un sistema productivo.

Si bien en ningún momento en la propuesta menciono SCRUM, hago notar en las etapas de desarrollo, la participación permanente del cliente y entregas por casos de uso o requerimientos prioritarios, que es justamente los sprint que propone SCRUM.

Para las definiciones de requerimientos presento una propuesta diferente para especificar los casos de uso, en relación a la notación que enseña UML. Esta propuesta es una opción más técnica, son **casos de uso basados en la ejecución de transacciones**, donde se describe puntos de menú del cual se va a ejecutar la transacción (pantalla o secuencia de pantallas), estructura de la transacción con los tipos de datos de cada campo, validaciones que se deben realizar campo por campo y validaciones que se deben realizar a nivel de eventos de la transacción.

La ventaja que tiene sobre la Especificación de Casos de Uso de UML es que tiene menos redacción, no es tan extensa su notación y al visualizar la transacción con los campos y los tipos de datos, permite tener una noción rápida del requerimiento, tanto para el usuario, como para el analista y para el programador. Puede ser que un caso de uso llame a otro caso de uso, en tal caso se puede agregar en que evento se produce la llamada a otro caso de uso y luego describir el mismo, es decir mostrar la relación entre casos de uso mediante un CALL().

Si la transacción involucra varios sectores dentro de la empresa, se puede anexar un Diagrama de actividades que describa cómo interactúan los sectores, mostrando las actividades que cada uno lleva a cabo.

La documentación del requerimiento debe existir, no debe ser una tarea ligera dentro de las metodologías ágiles pero tampoco debe ser extensa como se propone en UML.

Luego en mi propuesta agrego un diseño de documento técnico que se debe presentar una vez codificado un proceso. Las ventajas que tiene la creación de este documento, tal como se ha mencionado en varias ocasiones en esta tesina, es que permite comprender el comportamiento de un proceso, permite a los analistas hacer el análisis sin la necesidad de acudir a los programadores para la lectura del código, dar soporte al usuario en relación a la funcionalidad del sistema, realizar las pruebas y validar el resultado esperado, entre otras ventajas al momento de su implementación.

La desventaja que se puede presentar es que si no existe el hábito de realizarlo frecuentemente, puede insumir más horas de un recurso de las que estaban previstas, puede darse también que el documento no esté completo y no refleje todo lo que realiza el proceso que se ha programado.

Esta propuesta puede no ser de utilidad para todos los lenguajes de programación, dado que en la propuesta de **Casos de Uso basado en Transacciones** que presento está desarrollada en base a una transacción de Genexus <sup>21 22</sup>, en donde tiene incorporado la normalización de la base de datos que mantiene en forma óptima y que se actualiza justamente con estas transacciones o vistas o casos de uso del usuario.

Pero si en lugar de basarnos en una transacción, en el caso de uso describimos los datos que se ingresan en una pantalla, en forma de tabla con los campos como fue armado en mi propuesta y además se agregan Diagramas para describir la vista estática del sistema, podría usarse esta documentación para definir requerimientos que se codifiquen en otro lenguaje de programación.

Esta es mi propuesta, una propuesta de documentación simple con un enfoque más técnico, la cual se puede incorporar a las Metodologías Ágiles utilizadas hoy en día.

1. Documento de Casos de Uso basados en Transacciones (Relevamiento, Análisis y Diseño).
2. Documento de Diagramas de Actividades para los procesos de negocio (Relevamiento).
3. Casos de Pruebas
4. Documentación Técnica de Procesos

A continuación se desarrolla un ejemplo de cada documentación.

<sup>21</sup> <https://es.wikipedia.org/wiki/GeneXus>

<sup>22</sup> [https://es.wikipedia.org/wiki/ARTech\\_Conultores\\_SRL](https://es.wikipedia.org/wiki/ARTech_Conultores_SRL)

## 6. Ejemplo de Documentación Propuesta.

### 6.1. Documento de Casos de Uso basados en Transacciones (Relevamiento, Análisis y Diseño).

Es un documento que puede crearse en el momento de relevar o después de relevar y debería estar validado por el cliente antes de enviarlo al sector de desarrollo para que generen el código correspondiente.

<b>Caso de Uso: Alta de un Cliente</b>			<b>Fecha: 20/10/18</b>
<b>Estado:</b> Análisis			Tipo: Primario
<b>Actor:</b> Responsable de Administración			
<b>Descripción:</b> Responsable de Administración Ingresa Datos del cliente.			
<b>Parámetros Previos a la ejecución:</b> Deben existir datos en las tablas Localidades, Códigos Postales, Categoría IVA y Tipos de Documento.			
<b>Resultado Esperado:</b> Debe quedar registrada el alta del nuevo cliente con un código de cliente generado en forma automática por el sistema.			
<b>Descripción Interfaz de Entrada, Proceso, Validaciones e Interfaz de Salida</b>			
<p><b>Desde el punto de menú Clientes, debe existir una pantalla “Mantenimiento de Clientes”,</b> que permita dar de alta un nuevo cliente, modificarlo y eliminarlo.</p> <p>Los datos a Ingresar en la Transacción “Alta de cliente” son los siguientes:</p>			
Campo	Tipo de dato	Descripción	Validaciones a nivel de campo
ClienteCod*	Numeric(6)	Código de Cliente	Se genera en forma automática, es único y secuencial.
ClienteNom	Character(60)	Nombre de Cliente	Ingreso Obligatorio
ClienteApe	Character(60)	Apellido de Cliente	Ingreso Obligatorio
ClienteTipDoc	Character(5)	Tipo de Documento del Cliente	Ingreso Obligatorio. Obtener de tabla Tipos de códigos de documentos. En caso de ingresar un dato que no existe en la tabla Tipos de códigos de Documentos, mostrar mensaje de error.
ClienteDoc	Numeric(8)	Documento del cliente	Ingreso Obligatorio
ClienteFchNac	Date()	Fecha Nacimiento de Cliente	
ClienteCatIva	Numeric(2)	Categoría IVA	Obtener Código de tabla Categoría de IVA. Ingreso Obligatorio. En caso de ingresar un dato que no existe en la tabla Categoría de IVA, mostrar mensaje de error.
ClienteDom	Character(60)	Domicilio del Cliente	
ClienteCodPos	Numeric(8)	Código Postal del Cliente	Obtener Código Postal de Tabla Códigos Postales. Validar que exista en la tabla el dato ingresado.

ClienteLocal	Character(60)	Localidad del Cliente	Se completa después de ingresar Código Postal
ClienteTel	Character(14)	Teléfono de Cliente	
ClienteTelOficina	Character(14)	Teléfono de Cliente en la Oficina	
ClienteCodVia	Numeric(8)	Código de Viajes Frecuentes del Cliente	Ingreso Opcional. Obtener el código de Viaje de la Tabla Viajes. Debe permitir ingresar más de un viaje.

### Validaciones en Evento Confirmación.

**En la Confirmación de la Transacción**, el sistema debe validar que no se haya ingresado anteriormente el número de documento. Si el documento ya existe, el sistema debe mostrar un mensaje indicando el código de cliente correspondiente.  
Esta validación podría hacerse también en el momento en que se carga el documento.

### Tablas que Graba el Proceso.

Después de ingresar y Confirmar los datos, quedarán grabadas las tablas Clientes y ViajesClientes.

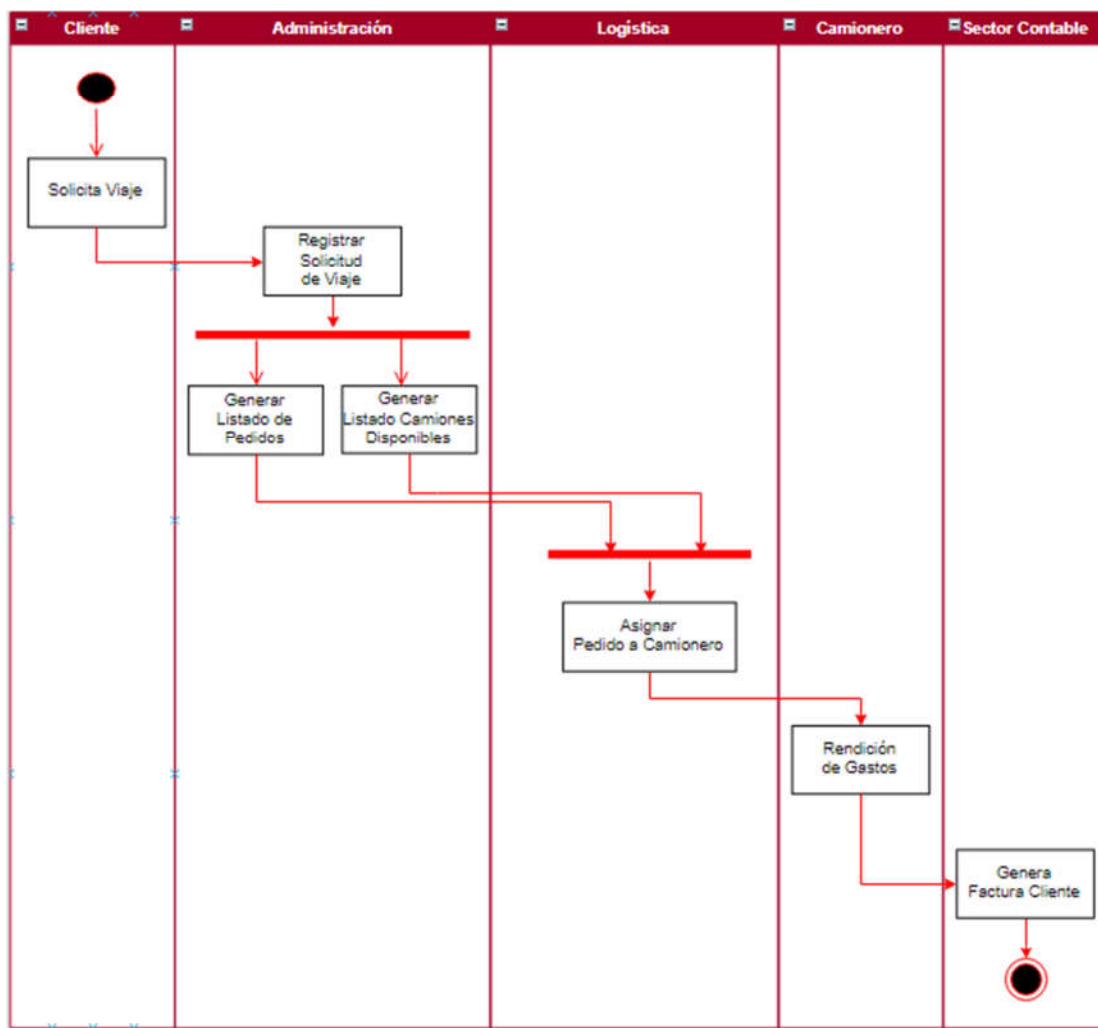
Tabla CLIENTES		
Campo	Tipo de dato	Descripción
ClienteCod*	Numeric(6)	Código de Cliente
ClienteNom	Character(60)	Nombre de Cliente
ClienteApe	Character(60)	Apellido de Cliente
ClienteTipDoc	Character(5)	Tipo de Documento del Cliente
ClienteDoc	Numeric(8)	Documento del cliente
ClienteFchNac	Date()	Fecha Nacimiento de Cliente
ClienteCatIva	Numeric(2)	Categoría IVA
ClienteDom	Character(60)	Domicilio del Cliente
ClienteCodPos	Numeric(8)	Código Postal del Cliente
ClienteLocal	Character(60)	Localidad del Cliente
ClienteTel	Character(14)	Teléfono de Cliente
ClienteTelOfic	Character(14)	Teléfono de Cliente en la Oficina

Tabla VIAJESCLIENTES		
ClienteCod*	Numeric(6)	Código de Cliente
ClienteCodVia	Numeric(8)	Código de Viajes Frecuentes del Cliente

## 6.2. Documento de Diagramas de Actividades para los procesos de negocio (Relevamiento).

A continuación se presenta el diagrama de actividades que involucra a varios sectores de la Empresa de Transporte desde que se solicita el viaje hasta que se genera la factura al cliente. Cada uno de estos sectores debe tener disponible en el Sistema a implementar un punto de menú que les permita realizar estas actividades.

Del relevamiento que se realice en la Empresa, se pueden obtener más de un diagrama de actividades que muestre el circuito que involucra el ingreso de datos al sistema con los sectores responsables y las salidas de datos que necesitan generar.



### 6.3. Casos de Pruebas (Pruebas en servidor de Desarrollo y de Testing)

Para los casos de prueba, la propuesta es simplemente usar las funcionalidades del sistema, de los nuevos procesos que se desarrollaron y mostrar como es el funcionamiento.

Generar casos de prueba con resultados exitosos, que permitan guardarlos como documentos que reflejen la funcionalidad.

Es importante especificar cuáles son los datos previos que deben estar guardados en las tablas del sistema para que el caso de prueba finalice de la forma esperada.

#### **Caso de Prueba: Asignar Viaje a un Pedido.**

Ingresar al Sistema desde el Navegador Web con la URL que se ve en pantalla en donde el operador deberá ingresar credenciales para autenticarse.

Se visualizará el menú principal del sistema.

Se debe ingresar en el Punto de Menú **Viajes** a la opción **Pedidos de Viajes**.

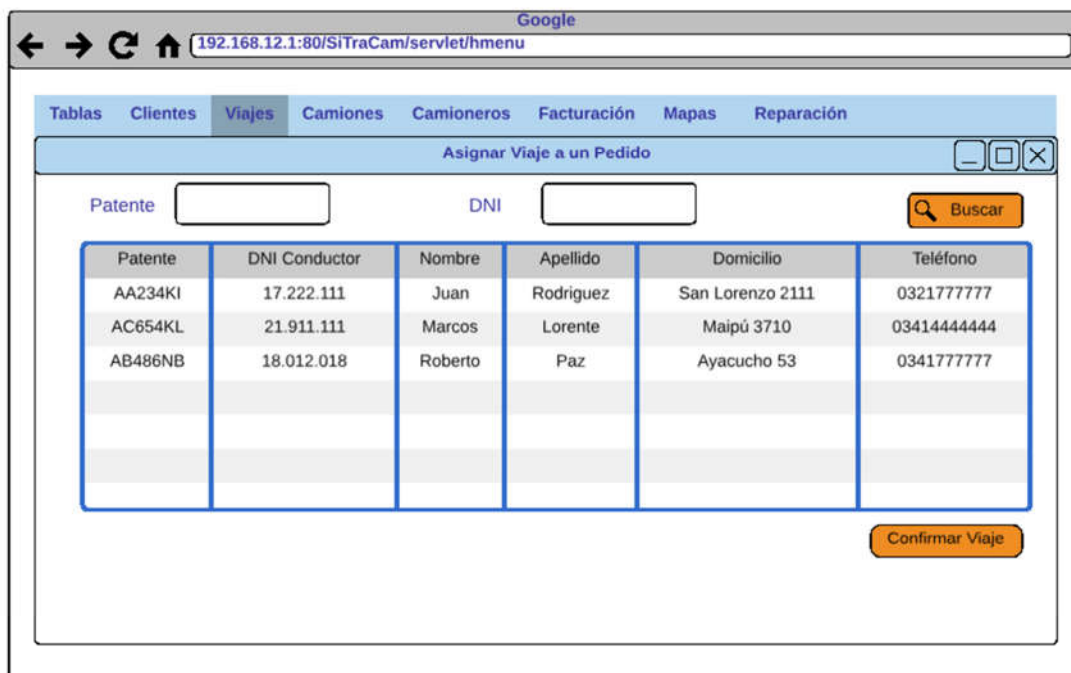
Se accede a la siguiente pantalla en donde se pueden buscar los pedidos con Estado Pendiente, es decir los pedidos que aún no tienen un viaje asignado.

Nro.Ped.	Fec.Pedido	Destino	Estado	CódCliente	Nombre	Apellido	Domicilio	Teléfono
2485	05/12/18	Necochea	Pendiente	7891	Alfonso	León	San Juan 2111	0321777777
2486	11/12/18	Viedma	Pendiente	9111	Mario	Paz	Maipú 3710	0341444444
2486	18/12/18	Corrientes	Pendiente	3411	Pedro	Antonelli	Ayacucho 53	0341777777

Se debe seleccionar el renglón correspondiente a un pedido de la grilla y al presionar Asignar Viaje, se abre la siguiente pantalla en donde podrá visualizar los camiones disponibles que realizan el viaje al destino solicitado. Se selecciona el renglón de la grilla correspondiente al camión que será asignado para realizar el viaje y se presiona "Confirmar Viaje".

El sistema mostrará un mensaje "El viaje ha sido asignado" dejando el pedido en estado "En Viaje".





Patente	DNI Conductor	Nombre	Apellido	Domicilio	Teléfono
AA234KI	17.222.111	Juan	Rodriguez	San Lorenzo 2111	0321777777
AC654KL	21.911.111	Marcos	Lorente	Maipú 3710	0341444444
AB486NB	18.012.018	Roberto	Paz	Ayacucho 53	0341777777

#### 6.4. Documentación Técnica de Procesos (Tarea posterior a la etapa de Programación).

A continuación se presenta una alternativa de documentación técnica para el caso de prueba planteado anteriormente.

- **Parámetros previos a la ejecución del proceso.**

Los datos de los camiones y camioneros deben estar cargados en forma completa antes de asignar el pedido de viaje, de lo contrario no podrá confirmar la asignación del mismo.

- **Punto de menú desde donde se ejecuta.**

Se debe ingresar en el punto de menú Viajes → Procesos → Pedidos de Viajes  
Objeto a agregar al menú: WPedViaje

- **Pasos a tener en cuenta en la ejecución.**

Ingresar al Sistema desde el Crhome a la URL que se ve en pantalla e ingresar credenciales para autenticarse.

Se visualizará el menú principal del sistema.

Se debe ingresar en Punto de Menú Viajes a la opción **Pedidos de Viajes**.

Se accede a la siguiente pantalla en donde se pueden buscar los pedidos con Estado Pendiente, es decir los pedidos que aún no tienen un viaje asignado.



En esta misma pantalla se podrá seleccionar el pedido y cancelarlo en caso que así lo solicite el cliente.

The screenshot shows a web browser window with the address bar displaying '192.168.12.1:80/SiTraCam/servlet/hmenu'. The application has a navigation menu with tabs: 'Tablas', 'Clientes', 'Viajes' (selected), 'Camiones', 'Camioneros', 'Facturación', 'Mapas', and 'Reparación'. The main section is titled 'Pedidos de Viajes'. It contains search filters: 'Nro Pedido Desde' and 'Nro Pedido Hasta' (text inputs), 'Estado' (a dropdown menu with 'Pendiente' selected), and 'Cod. Cliente' (text input). A 'Buscar' button is to the right. Below the filters is a table with the following data:

Nro. Ped.	Fec. Pedido	Destino	Estado	Cód. Cliente	Nombre	Apellido	Domicilio	Teléfono
2485	05/12/18	Necochea	Pendiente	7891	Alfonso	León	San Juan 2111	0321777777
2486	11/12/18	Viedma	Pendiente	9111	Mario	Paz	Maipú 3710	03414444444
2486	18/12/18	Corrientes	Pendiente	3411	Pedro	Antonelli	Ayacucho 53	0341777777

At the bottom right of the table area are two buttons: 'Cancelar Pedido' and 'Asignar Viaje'.

Se debe seleccionar el renglón correspondiente a un pedido de la grilla y al presionar Asignar Viaje, se abre la siguiente pantalla en donde podrá visualizar los camiones disponibles que realizan el viaje al destino solicitado.

The screenshot shows the 'Asignar Viaje a un Pedido' screen. It has the same navigation menu as the previous screen. The main section is titled 'Asignar Viaje a un Pedido'. It contains search filters: 'Patente' and 'DNI' (text inputs), and a 'Buscar' button. Below the filters is a table with the following data:

Patente	DNI Conductor	Nombre	Apellido	Domicilio	Teléfono
AA234KI	17.222.111	Juan	Rodriguez	San Lorenzo 2111	0321777777
AC654KL	21.911.111	Marcos	Lorente	Maipú 3710	03414444444
AB486NB	18.012.018	Roberto	Paz	Ayacucho 53	0341777777

A 'Confirmar Viaje' button is located at the bottom right of the table area.

Se selecciona el renglón de la grilla correspondiente al camión que será asignado para realizar el viaje y se presiona “Confirmar Viaje”.

El sistema mostrará un mensaje “El viaje ha sido asignado” dejando el pedido en estado “En Viaje”.

- **Lógica de procesamiento.**

Cuando se accede a la pantalla pedidos de viajes, el sistema muestra los pedidos que se encuentran en la tabla PedViaje de acuerdo a los parámetros ingresados en los filtros de la pantalla. Por defecto trae los pedidos que se encuentran en estado “Pendiente” (EstPedVie= P).

Al seleccionar un pedido y presionar “Asignar Viaje”, el sistema accede a la siguiente pantalla en donde se podrán visualizar los camiones disponibles para ese destino.

Para mostrar los datos de la grilla ingresa a la tabla DestCam (Destino Camion) donde el destino es igual al destino del pedido seleccionado.

Al seleccionar un renglón y presionar “Confirmar Viaje”, quedará el pedido en estado “En Viaje” y se grabará la tabla Viajes con el nro de pedido, el nro de patente.

- **Descripción del resultado (reportes de salida, estado o modificaciones que deben quedar en las tablas de la base de datos)**

Al confirmar el viaje el sistema mostrará un mensaje “El viaje ha sido asignado” dejando el pedido en estado “En Viaje”.

## 7. Evaluación de la Propuesta

Con esta propuesta se intenta simplificar las tareas de documentación que se usan en UML y que se estaba dejando de hacer en las Metodologías Ágiles.

En el caso del documento de casos de uso con un enfoque más técnico y orientado a los datos que se van a ingresar al sistema, es una alternativa que reduce notablemente la notación planteada en UML, es mucho más ágil y fácil de interpretar tanto por el usuario, como los analistas y desarrolladores.

El diagrama de actividades para modelar el negocio, BPMN (Modelo de Notación de Procesos de Negocio), es un diagrama que resulta muy útil cuando se realizan las reuniones con el cliente para definir los circuitos y revisar procesos que realiza cada sector. Es un diagrama que propone UML del cual considero que permite realizar un relevamiento rápido inclusive es un diagrama que podría armar el mismo usuario si es que está en conocimiento de esta herramienta.

Los casos de prueba siempre se ejecutan en el sistema, pero no siempre se documentan. Cuando los procesos son muy complejos y se actualizan muchos datos e intervienen muchas tablas, es beneficioso que estén documentados porque a futuro permite entender cómo funciona el sistema y en el caso que surja la necesidad de realizar cambios solicitados por el usuario, simplifica la tarea del analista.

No siempre es necesario estar creando documentación, hay procesos que son simples, son evidentes e intuitivo el uso del sistema y la documentación técnica propuesta sería redundante para el usuario y una carga horaria adicional para la Empresa a cargo del desarrollo y la documentación.

Se deja esta propuesta como una alternativa de documentación para las Metodologías Ágiles, y se invita a innovar sobre las mismas con el fin de mejorarla.

## 8. Anexo

### 8.1. Caso Práctico Documentado con Metodología de Análisis y Diseño Estructurado

Después de concretar varias entrevistas en una Empresa de Transporte de Cargas, se realizó la documentación del Análisis y Diseño de Sistemas tal como la Metodología lo indica:

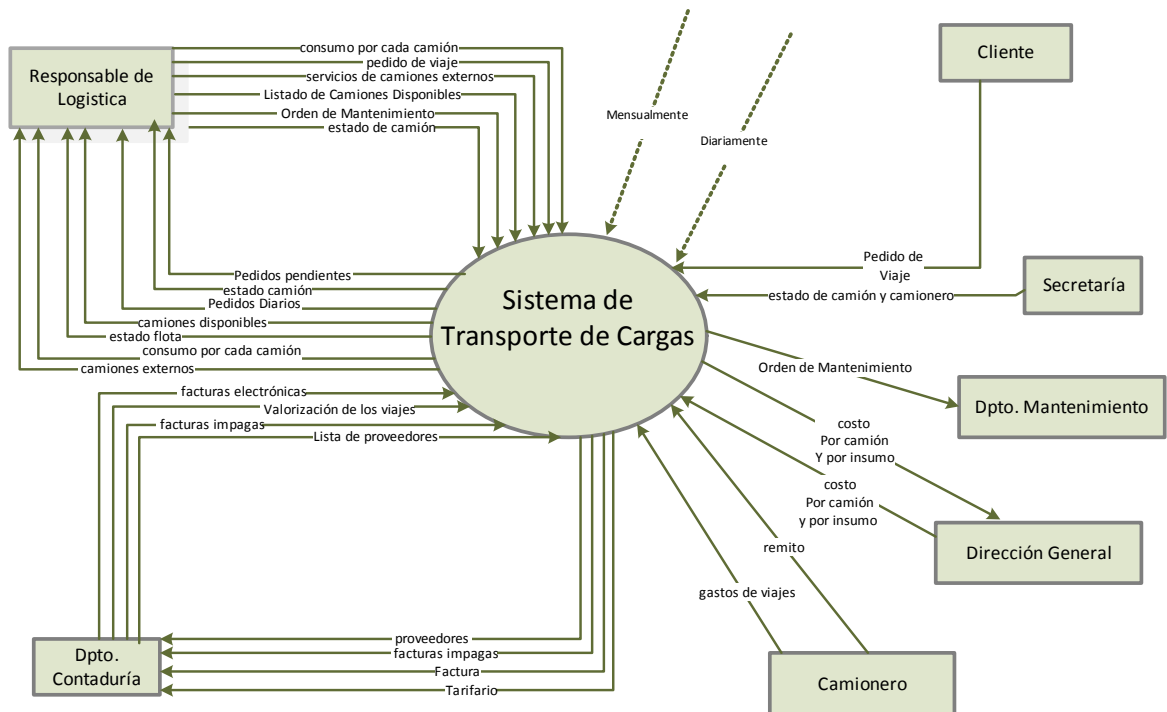
- ❖ El Modelo Ambiental, que consiste en definir:
  - Declaración de Propósitos
  - Diagrama de Contexto
  - Lista de Eventos
- ❖ Modelo de Comportamiento
  - Diccionario de Estructura de Datos
  - Diagrama de Flujo de datos Nivel 0
  - DFD Nivel 1
  - Diagrama Entidad Relación
  - Especificación de proceso
- ❖ Modelo de Implantación de Usuario
  - Determinación de la Frontera de Automatización
  - Determinación de la Interfaz Humana
  - Actividades de Apoyo Manual
  - Restricciones Operativas
- ❖ Modelos de Implantación del Sistema
  - Modelo de Procesadores
  - Modelo de Tareas
- ❖ Modelo de Implantación de Programas
  - Diagrama de Estructura

## ❖ Modelo Ambiental

### ▪ Declaración de Propósitos

El propósito del Sistema de Transporte de Cargas es brindar información detallada sobre los viajes realizados, la disponibilidad de equipos (camioneros y camiones), tiempo y forma de entrega como así también datos estadísticos que permitan al dueño optimizar la atención al cliente.

### ▪ Diagrama de Contexto



▪ **Lista de Eventos**

<b>Entrada</b>	<b>Salida</b>
Responsable de Logística consulta el estado de cada camión	Informe del estado del camión para el Responsable de Logística
Responsable de Logística genera orden de mantenimiento del camión	Orden de mantenimiento para Dpto. Mantenimiento
Cliente realiza pedido de un viaje	
Diariamente, generar un informe de pedidos	Informe de pedidos diarios a Responsable de Logística.
Responsable de Logística solicita listado de camiones disponibles	Informe de camiones disponibles para el Responsable de Logística
Responsable de Logística asigna el pedido de viaje de acuerdo al destino	
Secretaría envía la verificación de estado de camión y camionero	Informe sobre el estado de las flotas para el Responsable de Logística
Responsable de Logística contrata servicios de camiones externos a la empresa	Informe de camiones externos para el Responsable de Logística
Camionero entrega remito	
Camionero realiza rendición de gastos de viajes	
Dpto. Contaduría solicita valorización de los viajes para facturar al cliente	Tarifario de valorización de viajes a contaduría
Dpto. Contaduría solicita emisión de facturas electrónicas	Facturas correspondiente al Cliente
Dpto. Contaduría solicita listado de facturas impagas	Informe de facturas impagas
Dpto. Contaduría solicita listado de proveedores	Informe de proveedores al Dpto. Contaduría
Dirección General solicita costo por camión y por insumo	Informe detallado de los costos del camión con el insumo del mismo para la Dirección General
Mensualmente	Informe de pedidos pendientes
Dpto. Logística solicita informe de consumos por cada camión	Informe con los consumos por cada camión para Dpto. Logística

❖ **Modelo de Comportamiento**▪ **Diccionario de Estructura de Datos**

## ✓ Entidades

Empresa = cod\_Suc + CUIT\_emp + razón\_social\_emp + Domicilio (e)+ cond\_iva\_emp + nro\_IB\_emp

Proveedores = cod\_suc + cod\_prv + CUIT\_prv + razón\_social\_prv + Domicilio(e)+ cond\_iva\_prv +  
nro\_IB-prv + suc\_bco\_prv + cbu\_bco\_prv

Factura = cod\_suc + nro\_fact + letra + pto\_vta + fec\_fact + [CUIT\_prv|CUIT\_cli] + moneda +  
1{[cod\_concepto|cod\_viaje] + cantidad + precio}N + total

Camiones = cod\_suc + matricula +cod\_camionero+ nro\_motor + modelo + marca + 1{nro\_serie\_acop}2  
+ nro\_poliza + fec\_vto\_poliza + fec\_vto\_ctrl\_veh+estado

Camioneros = cod\_Suc + cod\_camionero + Camionero (e)+ fecha\_alta+ fecha\_baja

Mantenimiento =nro\_mant+ matricula + fec\_ing\_man + km\_cam + 1{codserv + duración + prioridad +  
1{insumo}N}N

Servicio= codserv+descserv

Pedidos = cod\_suc + cod\_pedido + cod\_cliente + fec\_viaje + tipo\_carga + cant\_carga + uni\_med +  
dom\_origen + dom\_destino + hora\_sal + hora\_ent + nro\_remito+estado

Clientes = cod\_cliente + CUIT\_cli + razón\_social\_cli + Domicilio (e)

Camioneros\_Ext = Camionero (e)

Viajes = cod\_suc + cod\_viaje + cod\_camionero + matricula + cod\_pedido + dinero\_transito+  
Km\_recorridos+ litros\_consum

Costo viaje = cod\_suc + cod\_viaje + 1{fec\_gasto + cod\_gasto + concepto + importe}N

Remito = cod\_suc + nro\_remi + fec\_remi + cod\_cliente + 1{tpo\_carga + cant\_carga + uni\_med}N

Tarifa = cod\_suc + cod\_tarifa + [km\_recorridos|cant\_carga] + valor\_tarifa

Ciudad = cod\_ciudad + desc\_ciudad + cod\_postal

Provincia = cod\_prov + desc\_prov

Pais = cod\_pais + desc\_pais

Sucursal=cod\_suc+ cod\_ciudad+cod\_rov

Estructura de datos

Domicilio(e)= calle + nro\_calle + piso + depto + cod\_ciudad + cod\_prov + cod\_pais + cod\_postal +  
teléfono + mail + Fax





Camionero (e) = CUIT\_cam + nombre + apellido + tpo\_dni + nro\_dni + Domicilio (e) + fec\_nac + licencia + clase + fec\_vto\_carnet + cond\_iva\_cam

Dom\_Origen = domicilio(e)

Dom\_Destino = domicilio(e)

✓ Tipo de datos

Tpo\_dni = DNI, LE, LC

Clase = A21, A22, A3, B1, B2, C, D1, D2, D31, D32, E1, E2, F, G1, G2

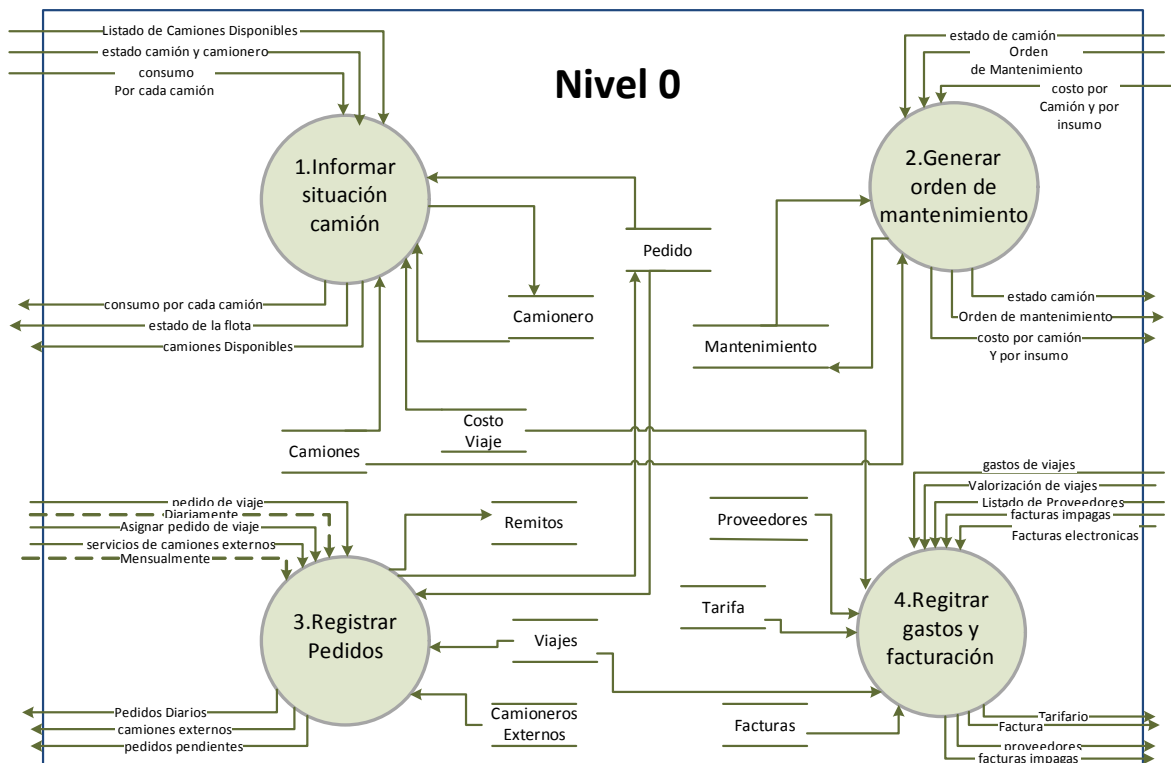
Tpo\_carga = bolsa, acoplado, cajas

Uni\_med = kg, litros, metros, cm

Cond\_iva\_Priv = [Exento|Resp. Insc.|Resp.No\_Insc|Cons.Final]

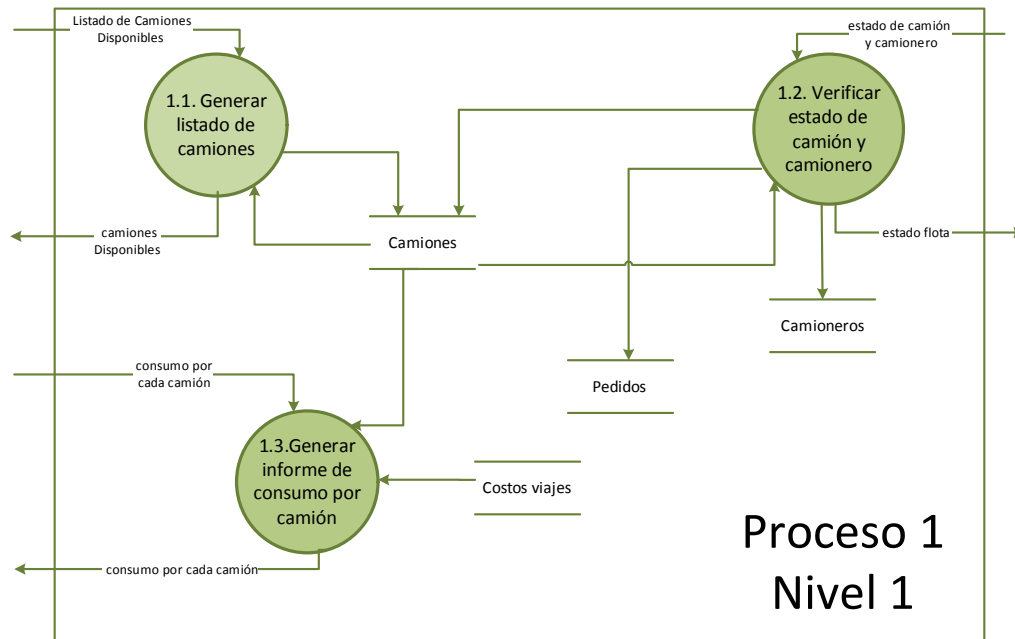
Cond\_iva\_emp = [Exento|Resp. Insc.|Resp.No\_Insc|Cons.Final]

#### Diagrama de Flujo de Datos Nivel 0

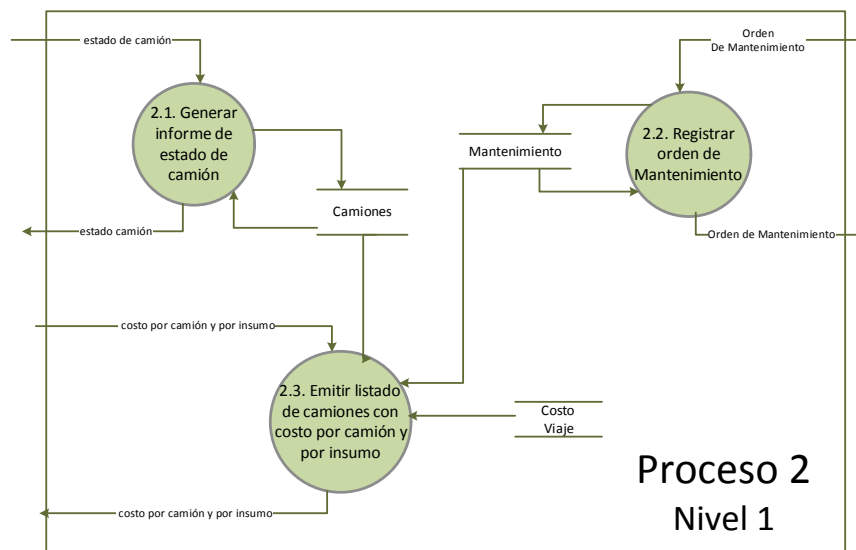




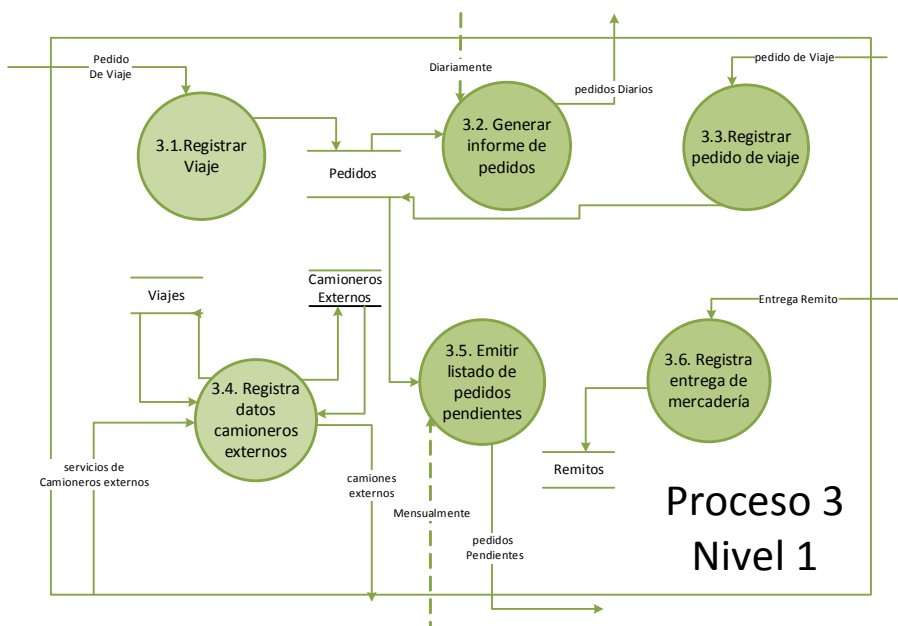
✓ DFD Nivel 1 Proceso 1



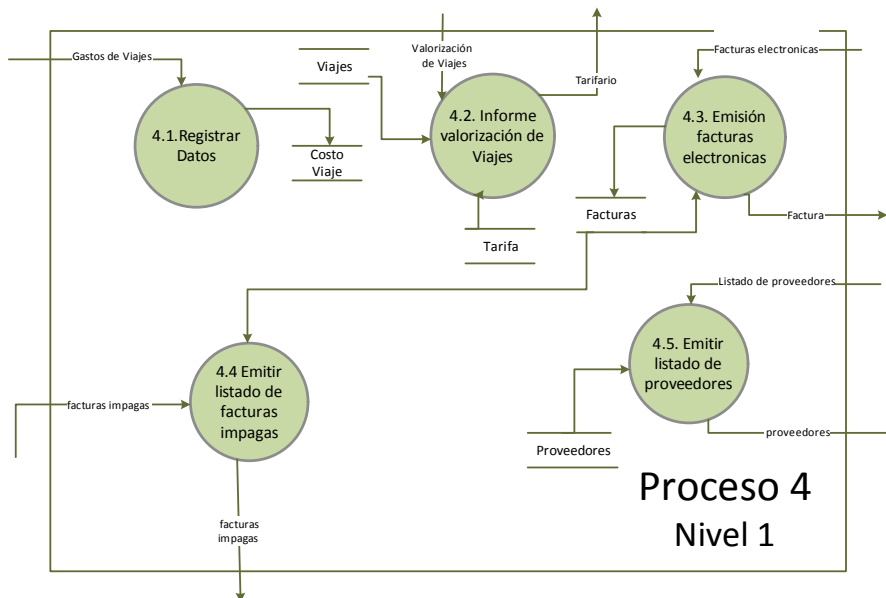
✓ DFD Nivel 1 Proceso 2



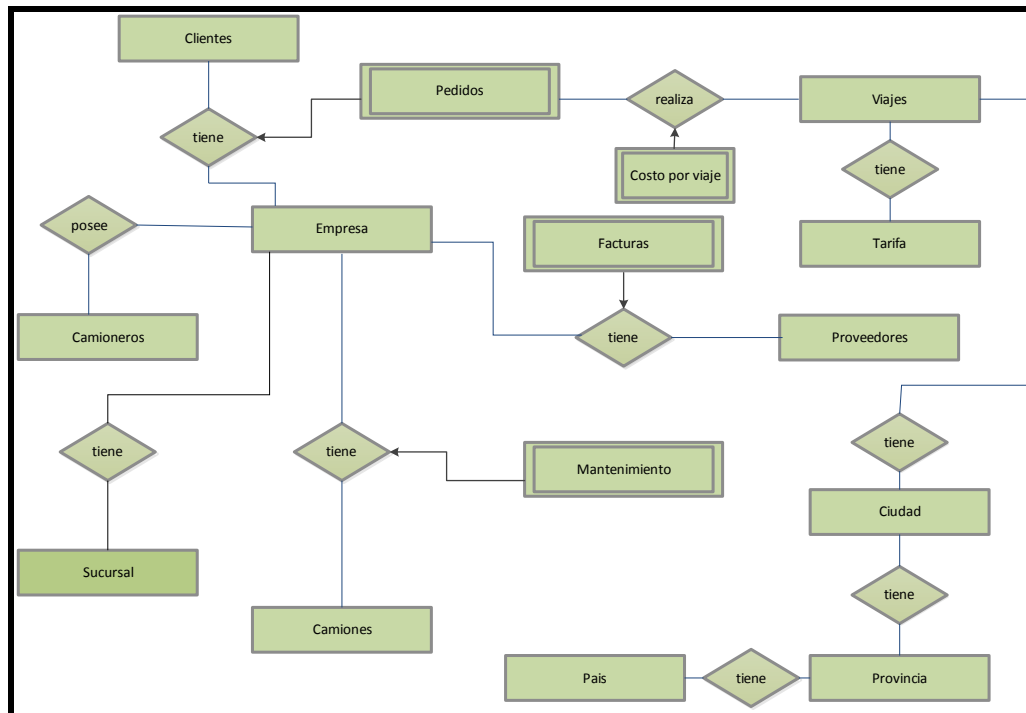
✓ DFD Nivel 1 Proceso 3



✓ DFD Nivel 1 Proceso 4



## ▪ Diagrama de Entidad Relación



## ▪ Especificación de Procesos

Por cada Proceso, se elabora una especificación de procesos. Para este caso, se muestra el desarrollo de dos procesos.

### 1.1. Ingresar fecha de hoy

Para cada camión obtener cod\_camion+matricula+ 1{nuro\_serie\_acop}2 + estado

Si estado= "Disponible"  
Imprimir en Listado de Camiones Disponibles

Fin Si

Fin Para



## 1.2. Ingresar matricula+cod\_camionero

Para cada camionero donde cod\_camionero=cod\_camionero

Obtener fec\_vto\_carnet

Si fec\_vto\_carnet &gt; 7

Obtener licencia+fec\_vto\_carnet+nombre+apellido

Para cada camión donde cod\_camionero=cod\_camionero

Obtener fec\_vto\_poliza

Si fec\_vto\_poliza &gt; 7

Obtener matricula+1{nuro\_serie\_acop}2

+nro\_poliza+fec\_vto\_poliza+fec\_vto\_ctrl\_vehi+estado

Fin si

Fin para

Imprimir en Informe de Flotas

Fin si

Fin para

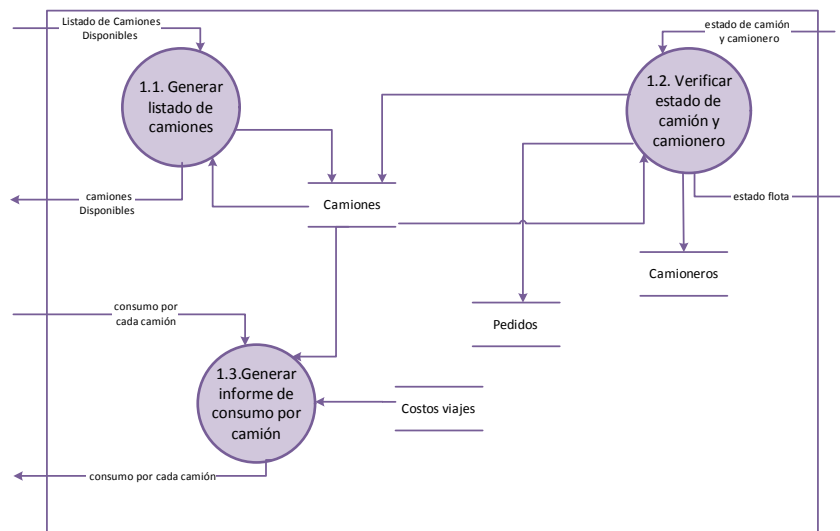
❖ **Modelo de Implantación de Usuario**▪ **Determinación de la frontera de Automatización**

De acuerdo a lo solicitado por el usuario se automatizará:

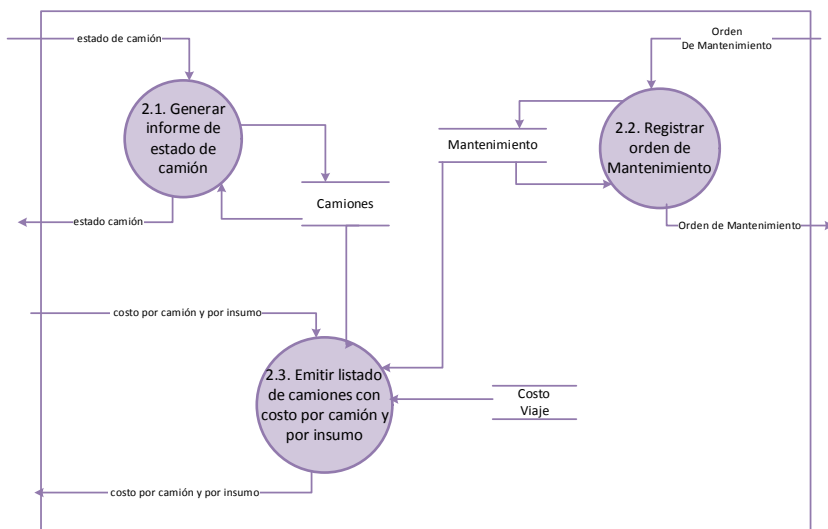
- La situación de cada camión correspondiente a la empresa
- La generación de Orden de mantenimiento
- La registración de Pedidos

En base al DFD se automatizarán los procesos coloreados:

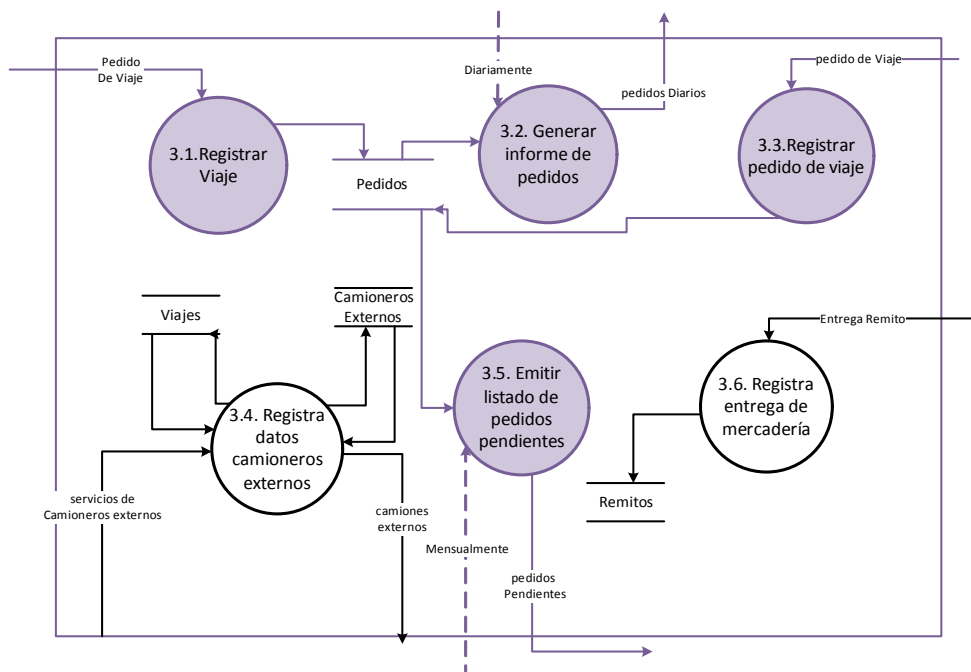
## ▪ Explosión del proceso “Informar situación de camión”



▪ Explosión del proceso “Generar Orden de Mantenimiento”



▪ Explosión del proceso “Registrar Pedidos”



No se automatizara:

- La registración de los camiones externos a la empresa
- La registración de la entrega de la mercadería
- Los gastos y facturación
- Determinación de la Interfaz Humana
- ✓ Interfaz (Pantallas de entrada/salida, formato)  
Pantalla donde el usuario puede obtener Listados de Camiones en un rango de Fechas.

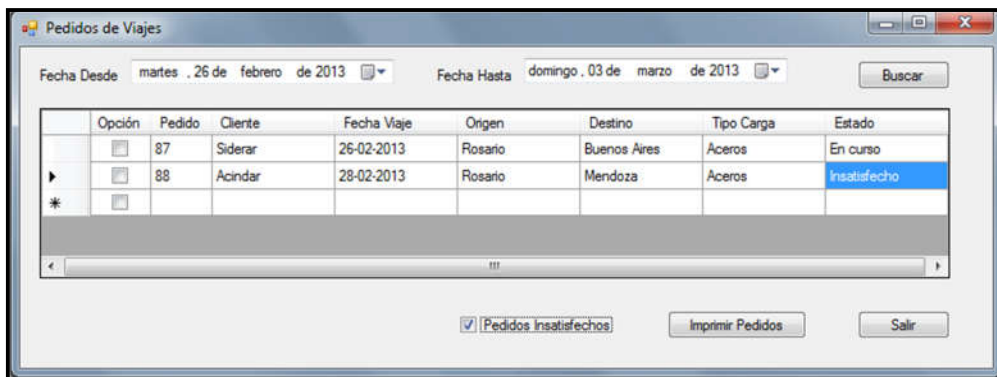
Pantalla que utiliza el usuario para realizar Consulta de datos de camiones.

	Opción	Matrícula	Camionero	Fec.Vto.Carnet	Cond. IVA	CUIT	Fec.Vto.Poliza	Fec.Vto.Técnica	Estado
	<input checked="" type="checkbox"/>	ASD987	Gonzalez, Mauricio	03/07/14	Monotributista	20-16324924-7	08/10/2013	09/06/2015	Disponible
*	<input type="checkbox"/>								

Listado que se obtiene de imprimir en la pantalla "Consulta datos Camión/Camionero"

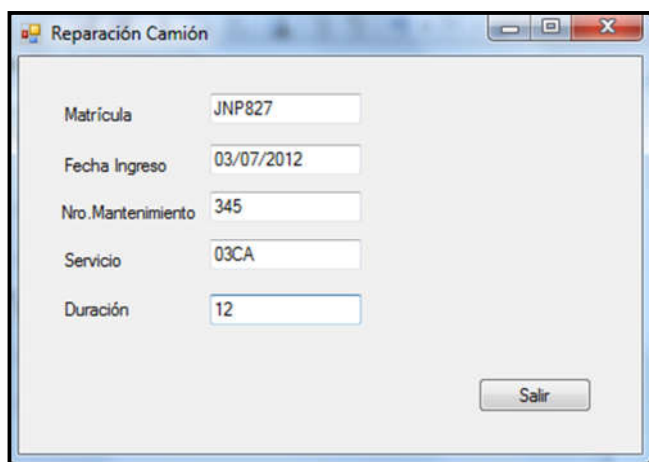
Informe Estado Flota							
Matrícula Desde/Hasta:		IVF453	al	IVJ999			
Matrícula	Camionero	Fecha Vto Carnet	Cond. Iva	CUIT	Fecha Vto.Poliza	Fecha Vto. Técnica	Estado
IVF453	Perez, Juan	09/07/2014	Monotributista	20-19088232-1	12/03/2014	03/08/2015	Disponible
IVF570	Diaz, Mauro	19/08/2013	Monotributista	20-08266542-9	21/07/2015	23/09/2013	Disponible

Pantalla que permite al usuario ingresar los pedidos de viajes

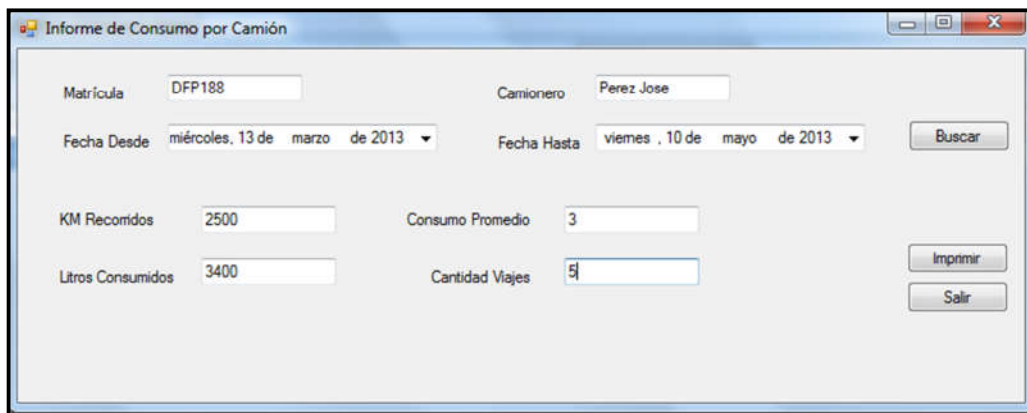


Opción	Pedido	Cliente	Fecha Viaje	Origen	Destino	Tipo Carga	Estado
<input type="checkbox"/>	87	Siderar	26-02-2013	Rosario	Buenos Aires	Aceros	En curso
<input type="checkbox"/>	88	Acindar	28-02-2013	Rosario	Mendoza	Aceros	Insatisfecho

Pantalla donde el usuario puede ingresar la Reparación de una Camión.



En la siguiente pantalla se puede generar informes de consumo por camión.



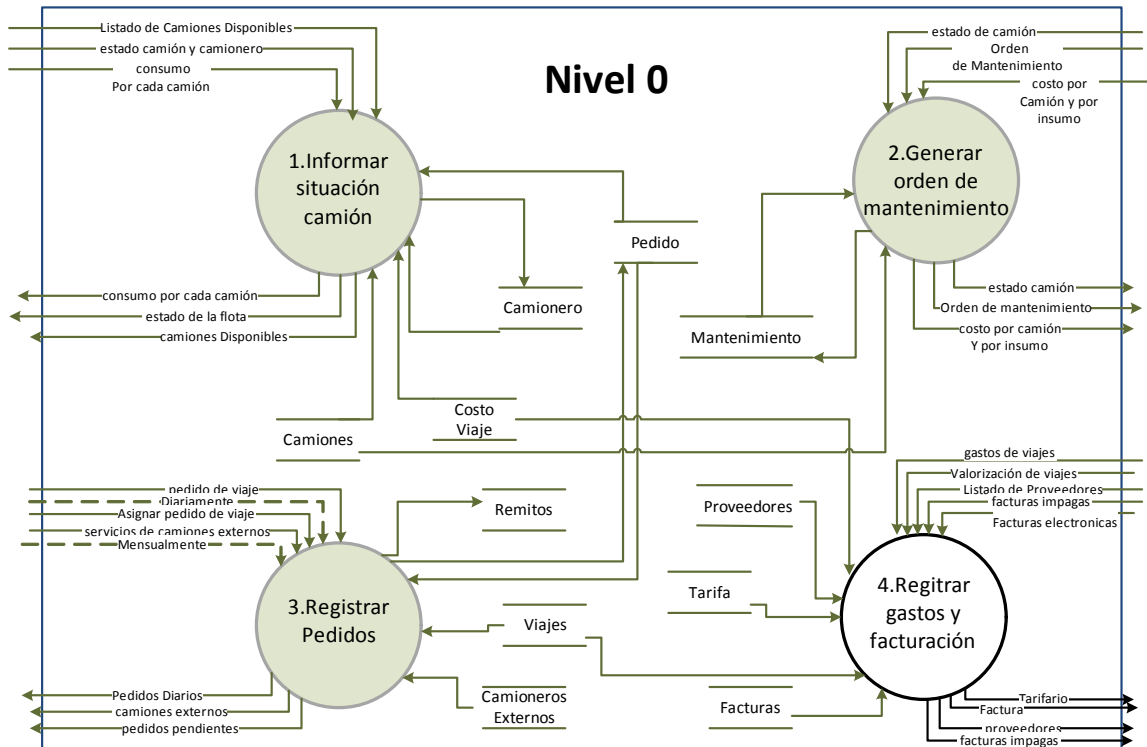


## ❖ Modelo de Implantación del Sistema

### ▪ Modelo de Procesadores

Teniendo en cuenta el Nivel 0 del D.F.D., hemos dispuesto la utilización de 1 único procesador para los correspondientes procesos los cuales detallaremos a continuación:

#### Procesador 1 asignado al Nivel 0 del DFD



Procesos que No se automatizarán:

- La registración de los camiones externos a la empresa
- La registración de la entrega de la mercadería
- Los gastos y facturación



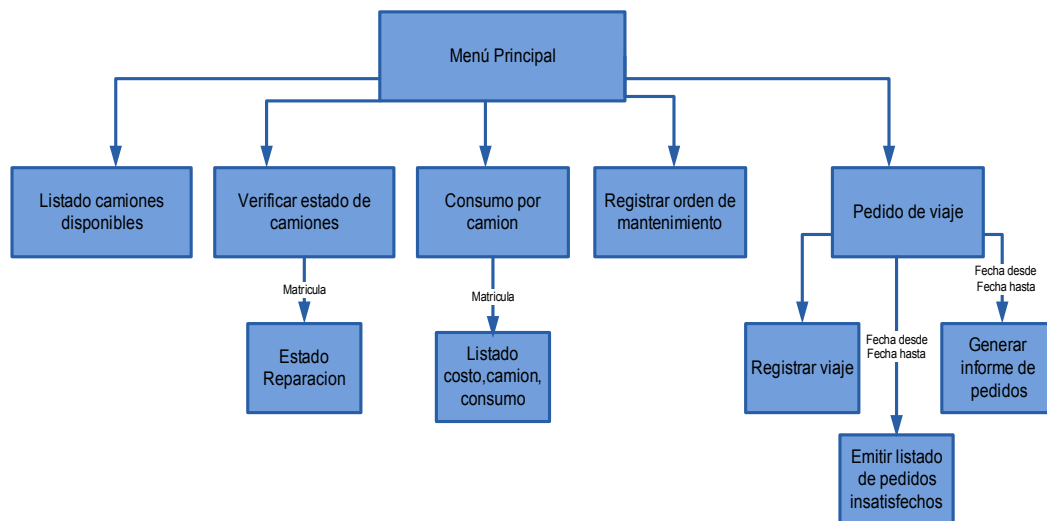
## ▪ Modelo de Tareas

Servidor 1.

Informar Situación Camión			Generar Orden de Mantenimiento			Registrar Pedidos			
Generar Listado de Camiones	Verificar estado de camión y camionero	Generar Informe de consumo por camión	Generar Inform de estado de camión	Registrar Orden de Mantenim.	Emitir Listado de camiones con costo por camión y insumo	Registrar Viaje	Generar Informe de pedidos	Registrar pedido de viaje	Emitir listado de pedidos pendientes

## ❖ Modelo de Implantación de Programas

### ▪ Diagrama de Estructura de Programas



Hasta aquí, se ha mostrado los modelos que propone la Metodología de Análisis y Diseño Estructurado, en donde se puede ver que se requiere de mucha documentación con la representación de diversos diagramas que buscan modelar aspectos diferentes del sistema.

Una vez que se inicia la etapa de codificación del sistema, si surge alguna modificación en algunos de los procesos será necesario ajustar el análisis y diseño inicial, lo cual significa modificar los diagramas correspondientes a ese proceso para que exista correspondencia entre lo codificado y la documentación.

La desventaja en esta metodología es el tiempo que lleva realizar tareas de actualización en forma permanente de la documentación, ya que es muy extensa por cada proceso o módulo que conforman el sistema.

Es una metodología que ha dejado de utilizarse aunque no se ha perdido la esencia del análisis que quedó de su enseñanza.

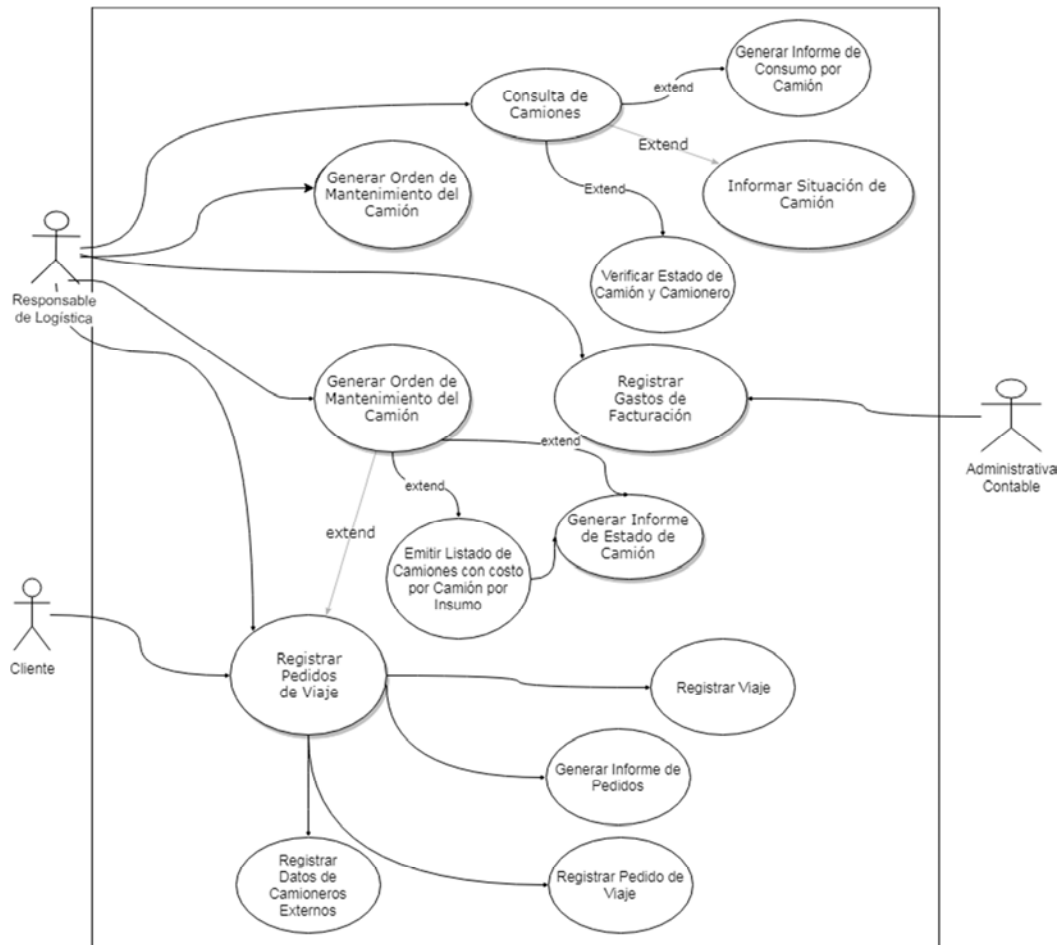
## **8.2. Caso Práctico Documentado con notación UML**

Tomando el mismo Sistema de Transporte de Cargas que se analizó anteriormente utilizando Metodología de Análisis y Diseño Estructurado, realizamos el Análisis con la Notación UML, Lenguaje de Modelo Unificado orientado a Objetos a fin de documentar cada caso de uso.

El objetivo es mostrar en un caso real, la extensa documentación que se requiere mantener, utilizando notación UML, lo cual puede no ser útil y práctico frente a determinados desarrollos.

Vamos a representar la mayoría de los Diagramas de UML y en algunos diagramas nos enfocaremos en un solo Caso de Uso, pero se entiende que los diagramas se deben realizar para todos los casos de uso.

#### ❖ Diagrama de Casos de Uso



### ❖ Especificación de Casos de Uso

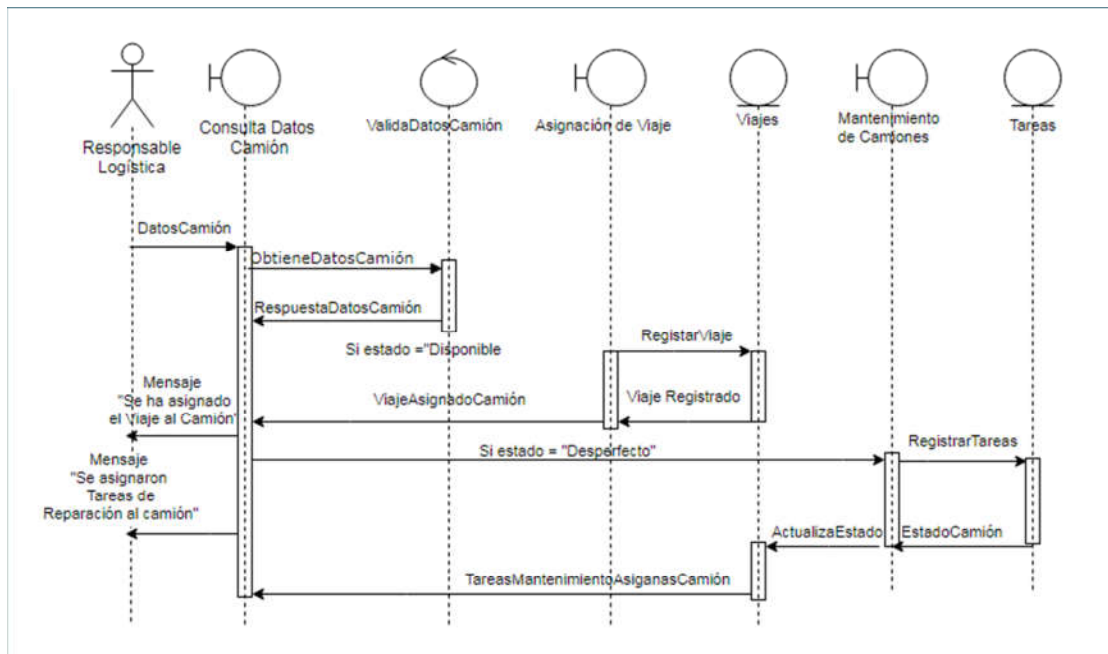
Por cada Caso de Uso, se debe mostrar como interactúa un actor con el Sistema.

Para el ejemplo, vamos a desarrollar el escenario o la especificación de Procesos de un solo caso de uso.

Caso de Uso: Consultar Estado de Camión		Fecha: 20/10/18
Estado: Análisis		Tipo: Primario
Actor: Responsable de Logística		
Descripción: Periódicamente el Responsable de logística consulta el estado de cada camión y el sistema informa los datos del camión junto con el estado del mismo, esto le permite conocer que tarea de mantenimiento debe realizarse a cada unidad, planificar paradas, optimizar tiempos y mejorar el cuidado de la flota. En caso de ser necesario genera un orden de mantenimiento, esto quedará registrado en el Dpto. Mantenimiento, y una vez que el camión es reparado, el mismo emite un informe de estado del camión.		
Precondición: El usuario debe identificarse y autenticarse.		
Post-Condición: El usuario tiene acceso a la información y opciones del módulo correspondiente.		
Actor		Sistema
1- Ejecuta aplicación del Sistema	Muestra una pantalla que permite el ingreso de usuario y contraseña.	
2- Ingresa usuario y contraseña.	El sistema valida datos ingresados y da acceso al menú principal del sistema.	
3- Ingresa al Menú de Logística, a la consulta de Camiones	Despliega Menú y muestra pantalla Consulta de Camiones	
4-Ingresa datos de camión	Muestra datos completos del camión y el estado del mismo.	
5. Selecciona Camión.		
Caminos Alternativos		
2	Los datos ingresados por el usuario son incorrectos. El sistema muestra mensaje, "Usuario y Contraseña incorrecto"	
4	Visualiza un mensaje de error "Los datos ingresados no corresponden a ningún camión registrado".	
5.1 Selecciona Camión con estado "Disponible", Ingresa asignación de viaje.	Llamar al Caso de Uso "Asignación de Viaje". Asigna el Viaje al Camión. Visualiza un mensaje, "El camión fue asignado al viaje". Cambia estado del camión a "Asignado"	
5.2 Selecciona Camión con estado "Desperfecto". Envía camión a Tareas de Mantenimiento	Muestra una pantalla donde permite asignar tareas de Mantenimiento al camión y programar dichas tareas. Llama al <b>Caso de Uso "Mantenimiento de Camiones"</b> . Registra Tareas de Mantenimiento. Cambia estado del camión a "En Reparación". Visualiza Mensaje, "Se asignaron Tareas de Reparación al camión"	
Importancia: Alta		
Comentarios: Ninguno		
Rendimiento Esperado: Optimo		

No existe un estándar para redactar el escenario de los casos de uso, esto quedará librado a cada analista de sistemas, lo cual puede generar interpretaciones diversas en los demás integrantes del proyecto de Ingeniería de Software.

## ❖ Diagramas de Secuencia para el Caso de Uso: “Consultar Estado de Camión”

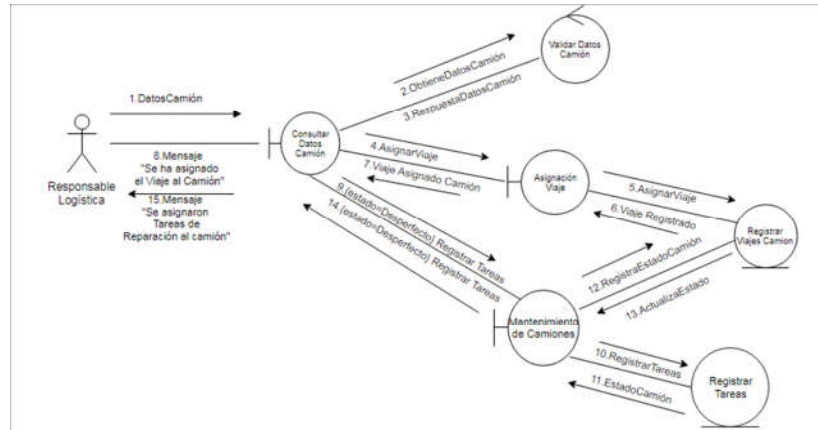


Por cada caso de uso del sistema se elabora un diagrama de secuencia que permite ver como interactúa el usuario con el sistema, cuales son las validaciones que hace el sistema, cuales son las entidades que guardan datos ingresados por el operador y los mensajes que recibe el operador en respuesta a cada una de sus acciones.

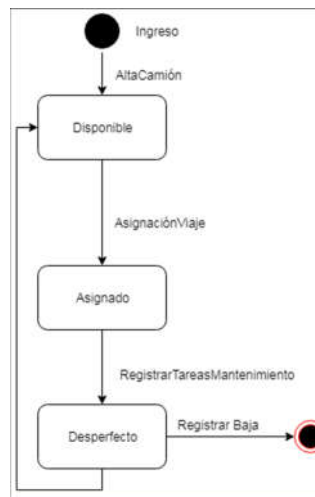
La desventaja que presenta, es que si estamos frente a una funcionalidad del sistema muy extensa y compleja, al intentar armar la secuencia de mensajes con los objetos puede extenderse demasiado generando un diagrama indeterminado y confuso.

❖ **Diagramas de Colaboración del Caso de Uso “Consultar Estado de Camión”**

En este diagrama se busca representar cómo interactúan los objetos de este caso de uso a través de mensajes que se envían en un orden específico. Sólo se intenta mostrar cómo se organizan los mensajes enviados entre los objetos.

❖ **Diagrama de Transición de Estados para el Caso de Uso “Consultar Estado de Camión”**

UML permite representar cambios de estados que se producen en un objeto.

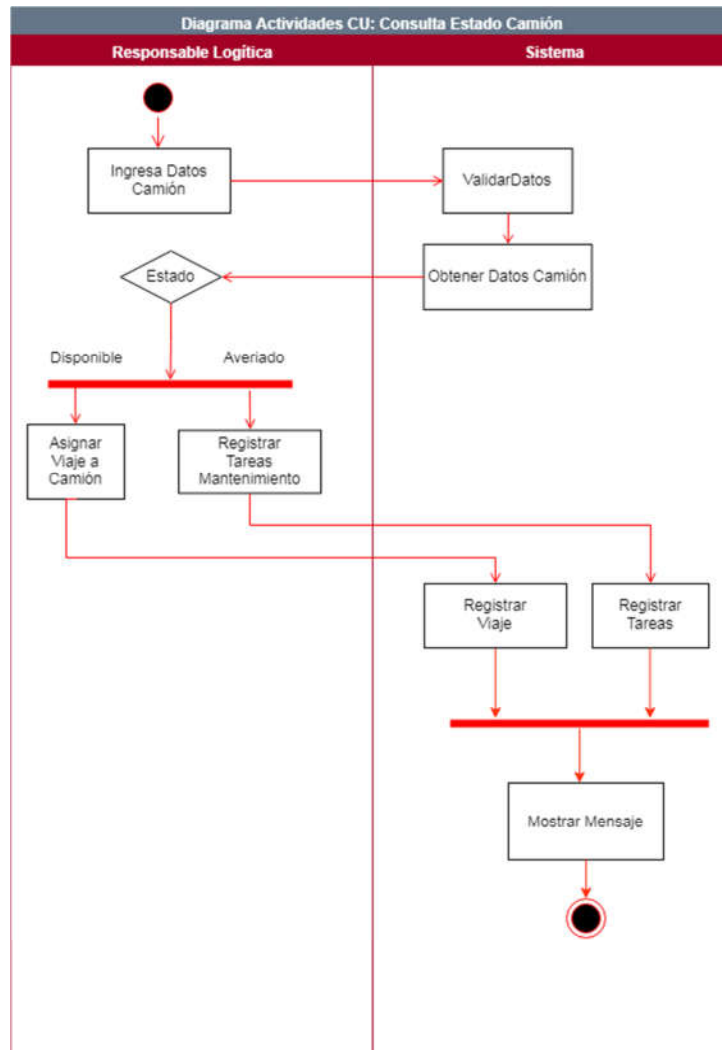


En este caso se puede ver cuando el camión es dado de alta en el sistema, toma estado "Disponible", cuando es asignado a un viaje, pasa al estado "Asignado". En el caso que se registre una avería, entra a tareas de mantenimiento y pasa al estado "Desperfecto". De este estado se puede registrar la baja o volver a estar disponible. No todos los casos de uso

requieren el diagrama de transición de estados. La secuencia de cambios de estados se produce cada vez que se da un evento externo y ocurre sobre un mismo objeto.

### ❖ Diagrama de Actividades

Los diagramas de actividades se realizan para cada Caso de Uso.

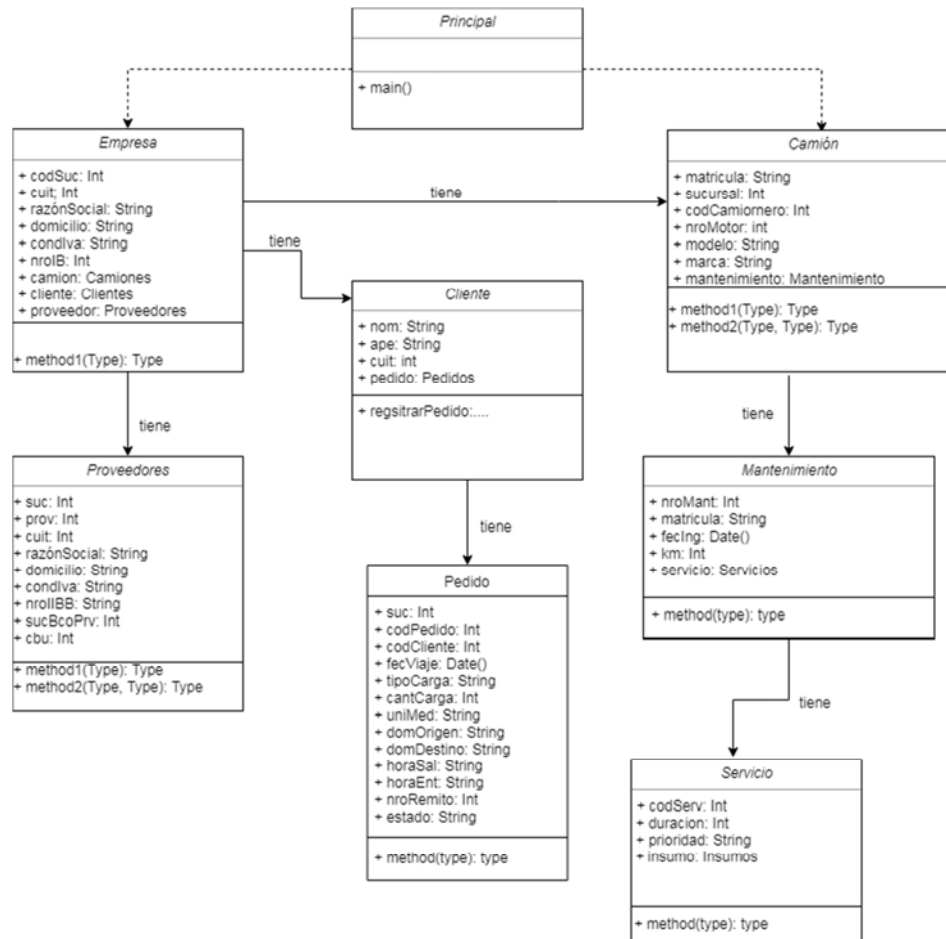


Se arman caminos o columnas, donde en cada columna se anotan las actividades que realizan el actor y el sistema. Representa una secuencia de acciones o actividades donde pueden intervenir elementos de bifurcación de tareas o bucles. Se representa un nodo inicial y un nodo final y conectores para los flujos de control.

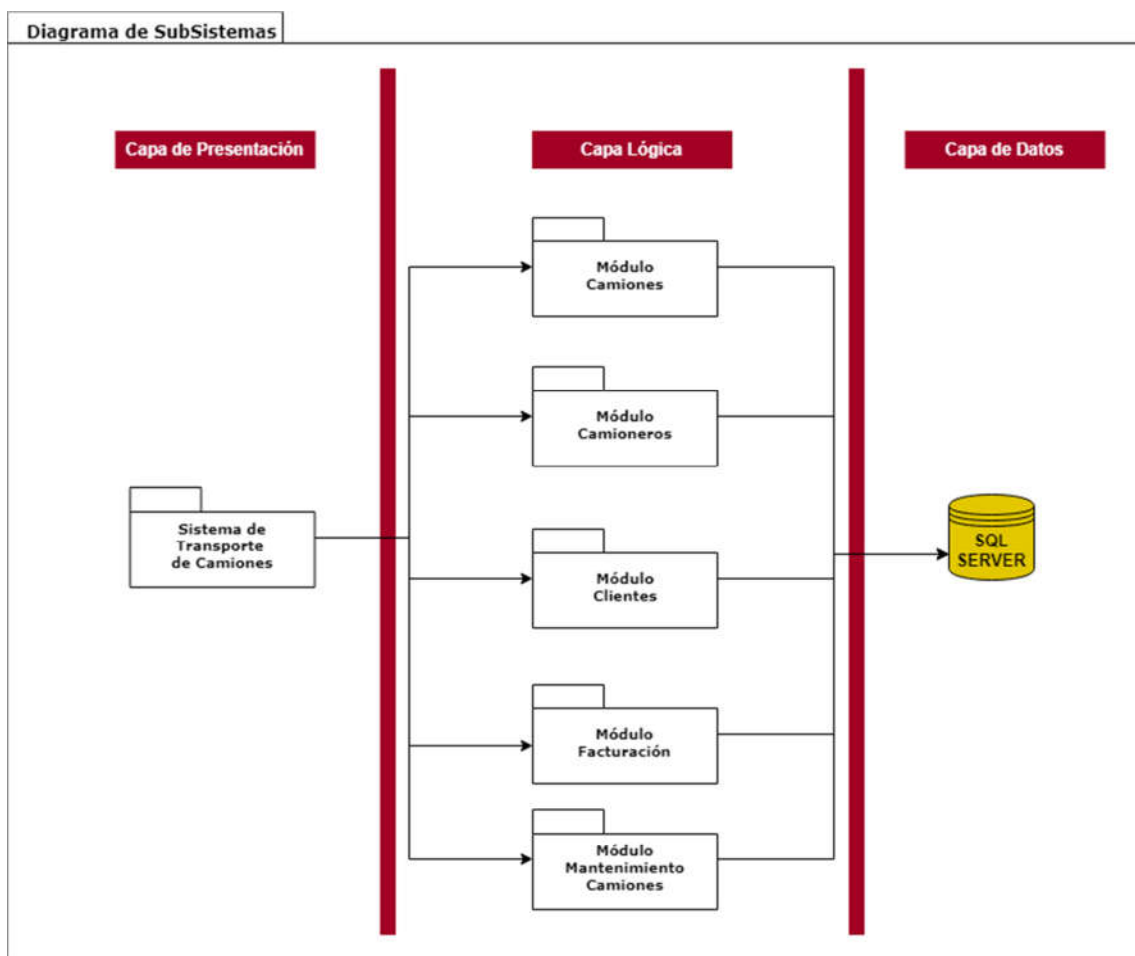


## ❖ Diagramas de Clases

Es un diagrama único para todo el Sistema, que modela la estructura de datos del sistema. Se representa con un conjunto de Clases que identifican objetos del sistema y tienen ciertas características o atributos. Se representan también las operaciones que realiza cada clase y como se relacionan las clases entre ellas.



## ❖ Diagrama de Subsistemas



## 9. Bibliografía

- 9.1. Análisis Estructurado Moderno (Edward Yourdon). Primera Edición – Prentice-Hall Hispanoamericana S.A. (1993).
- 9.2. El Lenguaje Unificado de Modelado (Grady Booch, James Rumbaugh, Ivar Jacobson). Primera Edición – Addison Wesley Iberoamericana es un Sello Editorial de Pearson Educación S.A. (1999).
- 9.3. El Proceso Unificado de Desarrollo de Software (Grady Booch, James Rumbaugh, Ivar Jacobson). Primera Edición – Addison Wesley Iberoamericana es un Sello Editorial de Pearson Educación S.A. (2000).
- 9.4. El Lenguaje Unificado de Modelado. Manual de Referencia. (Grady Booch, James Rumbaugh, Ivar Jacobson). Primera Edición – Addison Wesley Iberoamericana
- 9.5. UML y Patrones 2° Edición (Craig Larman) - Editorial Prentice Hall (2002).
- 9.6. <http://edgarreal10.blogspot.com.ar/2016/09/metodologias-agiles-vs-metodologias.html>
- 9.7. <http://www.oocities.org/es/raicelysgomez/analisis/t1.html>
- 9.8. <https://www.omg.org/spec/UML>
- 9.9. <http://profesores.fi-b.unam.mx/carlos/aydoo/toc.html>
- 9.10. <https://medium.com>
- 9.11. <https://blog.conectart.com/metodologias-agiles/>
- 9.12. <https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/>
- 9.13. <https://apiumhub.com/es/tech-blog-barcelona/startups-proyectos-procesamiento-del-lenguaje-natural/>