PROCESO ÁGIL PARA DESARROLLO AUTOMATIZADO DE SOFTWARE

Autores

Arturo Carlos SERVETTO[(1)] (aserve@mara.fi.uba.ar)

Ramón GARCÍA MARTÍNEZ[(2 y 1)] (rgm@itba.edu.ar)

Gregorio PERICHINSKY[(1)] (gperi@mara.fi.uba.ar)

[(1)] Tel. (011) 4342-9184 / 4343-0891, Int. 142

Laboratorio de Bases de Datos y Sistemas Operativos

Departamento de Computación

Facultad de Ingeniería de la Universidad de Buenos Aires

[(2)] Centro de Ingeniería de Software e Ingeniería del Conocimiento (CAPIS)

Escuela de Posgrado

Instituto Tecnológico de Buenos Aires

Resumen

El proyecto que sustenta este trabajo tiene como objetivo el desarrollo de una herramienta ICASE *(Integrated Computer Aided Software Engineering)* que produzca sistemas, esto es, el código y la documentación completos, a partir de especificaciones gráficas de alto nivel de abstracción. Para ello se definió un proceso de desarrollo ágil basado en la *prototipación evolutiva*, con ciclos conformados por la especificación o evolución de requerimientos, la especificación o evolución de diseño y la generación o regeneración del sistema[11].

Para modelar la estructura estática de los productos se emplean diagramas de clase que representan el modelo de dominio *(entity classes)*, y para modelar la estructura dinámica se emplea un modelo basado en la teoría de autómatas finitos[10]: se concibe a todo sistema como a un autómata cuyos estados se asimilan a interfaces *(boundary classes)*, y sus transiciones a funciones (métodos de un *controller* asociado a cada interfaz) que también se especifican con un alto nivel de abstracción, y que son la base para la generación del código. Para cada ciclo de evolución se contemplan las etapas del proceso refinándose estos diagramas.

Palabras Clave

Ingeniería de Software, ICASE (Integrated Computer Aided Software Engineering), Procesos Ágiles de Desarrollo de Software, Modelos de Comportamiento de Sistemas, Máquinas de Estado Finito.

Antecedentes

La realidad de la industria del software de gestión impone la adopción de procesos ágiles de desarrollo para lograr competitividad. Reflejo de ello, a nivel internacional, es la creciente consolidación de la filosofía Agile[8]. El objetivo principal de un proceso ágil es minimizar la documentación de desarrollo, empleándola fundamentalmente como vehículo de comprensión de problemas dentro del grupo de trabajo y de comunicación con los usuarios.

Otra característica de los procesos ágiles es que, atendiendo la importancia de la participación de los usuarios finales, suelen ser iterativos e incrementales. Un ejemplo de proceso ágil pero que confiere cierta importancia a la documentación de desarrollo es el denominado Iconix[14], formalizado por Rosemberg y Scott, que es iterativo e incremental y se basa en el UML[5] y se guía por casos de uso.

Los cultores del desarrollo ágil[20] suelen ser reacios a utilizar herramientas CASE ya que su utilización requiere tiempo y recursos que los desarrolladores ágiles no están dispuestos a resignar. Pero si se concibiese una herramienta que proponga un proceso ágil de desarrollo bien definido y simple, sería de gran ayuda tanto para los adherentes a la *Agile Alliance* como para los desarrolladores en general.

Descripción del Proceso Ágil

Para la *especificación de requerimientos* se clasifican los objetos entidad y se denotan sus relaciones, sin considerar aún sus propiedades o atributos en detalle, en el modelo estático, y se especifica una primera aproximación funcional del sistema, considerando los actores o categorías de usuarios y/o la organización funcional del sistema de información (agrupamientos funcionales o interfaces tipo menú), en el modelo dinámico.

La *especificación de diseño* se efectúa refinando el modelo de dominio mediante la especificación o agregado de atributos, y refinando el modelo dinámico mediante la especificación detallada de las interfaces, subdiagramas y transiciones. Las transiciones pueden ser simplemente de activación de interfaces, o pueden especificarse por QBE *(Query By Example)* o por un guión de álgebra relacional de entidades, según su complejidad.

La *generación o regeneración del sistema* es automática, y consiste en crear o evolucionar los esquemas de datos a partir de diagramas nuevos o cambios en diagramas preexistentes, y en recrear el código a partir de las operaciones especificadas mediante guiones de álgebra o QBE. La arquitectura de los sistemas producidos es MVC *(Model View Controller)* en tres capas: de interfaz, de reglas del negocio y de acceso a datos.

Caso de Estudio y Descripción del Modelo de Comportamiento

Al efecto de graficar el proceso de desarrollo y los modelos que se proponen, se considera un sistema para la administración de consorcios: registro de gastos corrientes de edificios y liquidación y cobro de expensas.

En la sección de Diagramas se incluyen tres que corresponden al modelo de dominio de la aplicación, ya a nivel de diseño, y dos correspondientes al modelo dinámico. Tal como el modelo estático, el dinámico puede incluir varios diagramas, anidados a partir del diagrama principal. Comparando con el UML[5][12], los diagramas dinámicos podrían considerarse como una amalgama de diagramas de casos de uso, de actividad y de secuencia; esto es, expresan lo mismo que estos tres diagramas del UML de manera integrada.

Cada interfaz es una instancia de interacción con el usuario, por lo que un diagrama de comportamiento representa la misma información que uno de casos de uso pero de manera más

atomizada. Las transiciones entre interfaces denotan la navegabilidad entre interfaces y por extensión, la conectividad entre funciones, por lo que los diagramas de comportamiento también subsumen la información de los diagramas de actividad. Y por último, las transiciones representan métodos de control que vinculan interfaces, por lo que también se puede decir que la información de los diagramas de secuencia está integrada.

Para graficar lo expresado anteriormente, considérese el diagrama de casos de uso incluido en la sección de diagramas, que representa la misma información que los dos diagramas de comportamiento.

El diagrama de primer nivel (ver diagrama titulado "Diagrama Principal de Comportamiento"), representa las agrupaciones funcionales con que se estructura el sistema que se desarrolla. De generarse el sistema a este nivel de especificación, se obtendría el menú principal del sistema con las interfaces iniciales de los casos o actividades de cada división funcional. Los diagramas del segundo nivel o superior, accesibles a partir de cada escenario o estado del nivel anterior, representan los casos, actividades y secuencias de interacción de cada división funcional. En estos, se pueden reutilizar interfaces ya definidas en otros diagramas, representándose con bordes tenues.

En el caso de estudio, se tiene una ventana de diálogo inicial para la acreditación del usuario, y luego una interfaz inicial o principal del sistema con cuatro opciones: Edificios, Cobros, Cambio de Clave y Usuarios. En Edificios, una vez seleccionado uno, se puede mantener la información referente a sus propiedades y personas responsables, clasificar sus tipos de propiedad y gastos, registrar gastos corrientes y también liquidar las expensas. En Cobros se registran pagos de expensas con actualización de saldos a partir de la última liquidación. Las dos últimas interfaces son para que los usuarios puedan cambiar su clave y para registrar nuevos usuarios o cambiar sus perfiles de acceso (autorizaciones).

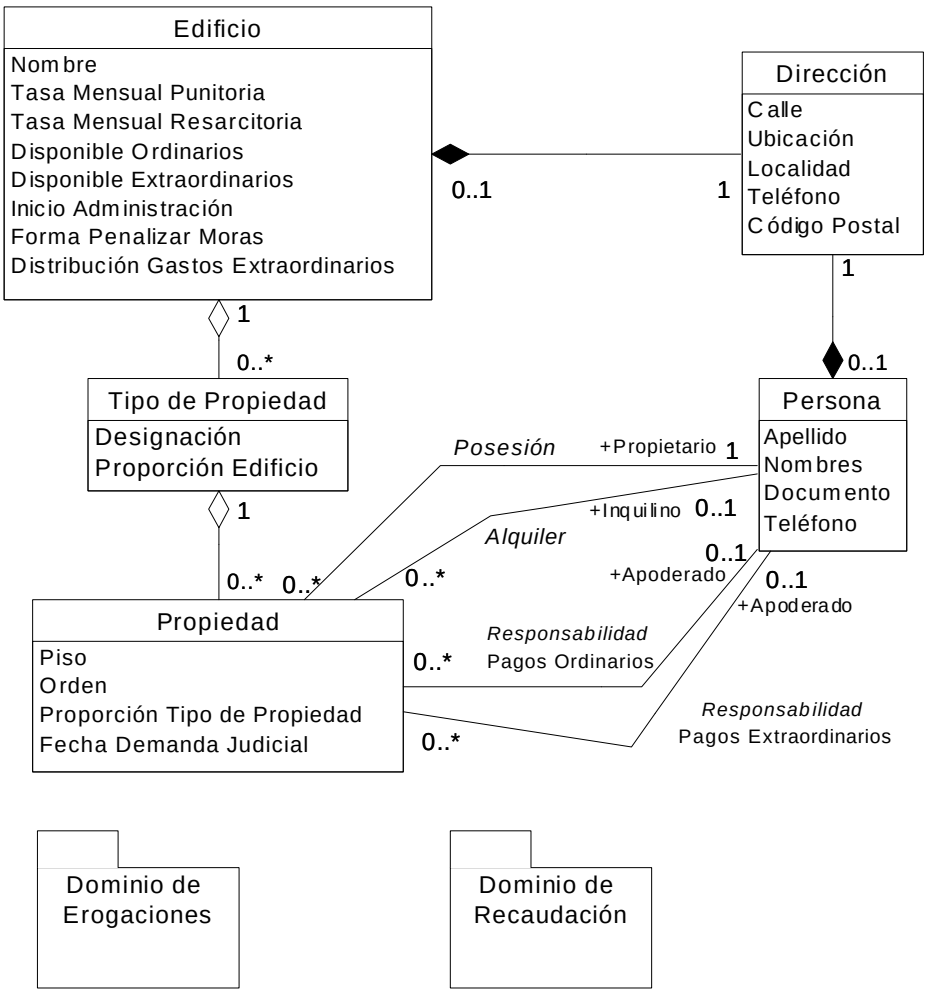
Como corresponde a cualquier sistema con GUI *(Graphic User Interfaces)*, para toda transición se asume por defecto (sin necesidad de denotación) la reversibilidad *(back tracking)* a través de la acción universal de Escape. También se asume que toda acción que modifique datos del sistema requiere confirmación explícita (por ejemplo la edición de un formulario) y que al determinarse tal confirmación, por defecto (si no hay una transición reflexiva) se retorna a la interfaz previa. Toda interfaz que requiera una confirmación seguramente desencadena algún tipo de validación sobre los datos y por consiguiente un curso alternativo de excepción, que por criterio estilístico y para favorecer la simplicidad y legibilidad de los diagramas, se deja para especificar como subdiagrama.

Las interfaces se clasifican en *Menús, Diálogos, Listas, Formularios, Formularios Continuos y Mensajes*. Los *Diálogos* sirven para representar solicitudes de argumentos o condiciones por parte del sistema. Las *Listas* sirven para visualizar referencias a objetos entidad o a instancias de vistas en un cuerpo principal de la interfaz (atributos clave y/o de búsqueda), y una vez seleccionado uno en la lista para acceder en una sección especial de la misma interfaz al resto de los atributos y eventualmente a otros datos relacionados (instancias de vistas), con posibilidades de edición, eliminación e incorporación. Los *formularios* permiten visualizar instancias de vistas en forma desplegada, y pueden contener también otros formularios. Los *formularios continuos* permiten visualizar vistas en forma tabular. Los *Mensajes* sirven para exhibir mensajes de error o de advertencia y se emplean principalmente en los cursos alternativos de excepción. Todos los formularios tienen por defecto una opción de impresión, por lo que se les pueden especificar atributos como encabezado y pie.

Los metadatos para la implementación de cada diagrama se especifican en un archivo XML (definición de clases y relaciones en los diagramas estáticos, e interfaces y transiciones en los dinámicos). En la sección Metadatos se incluye una DTD *(Document Type Declaration)* para Diagramas de Comportamiento y parte de la especificación de uno de los diagramas.

Diagramas

Dominio Principal

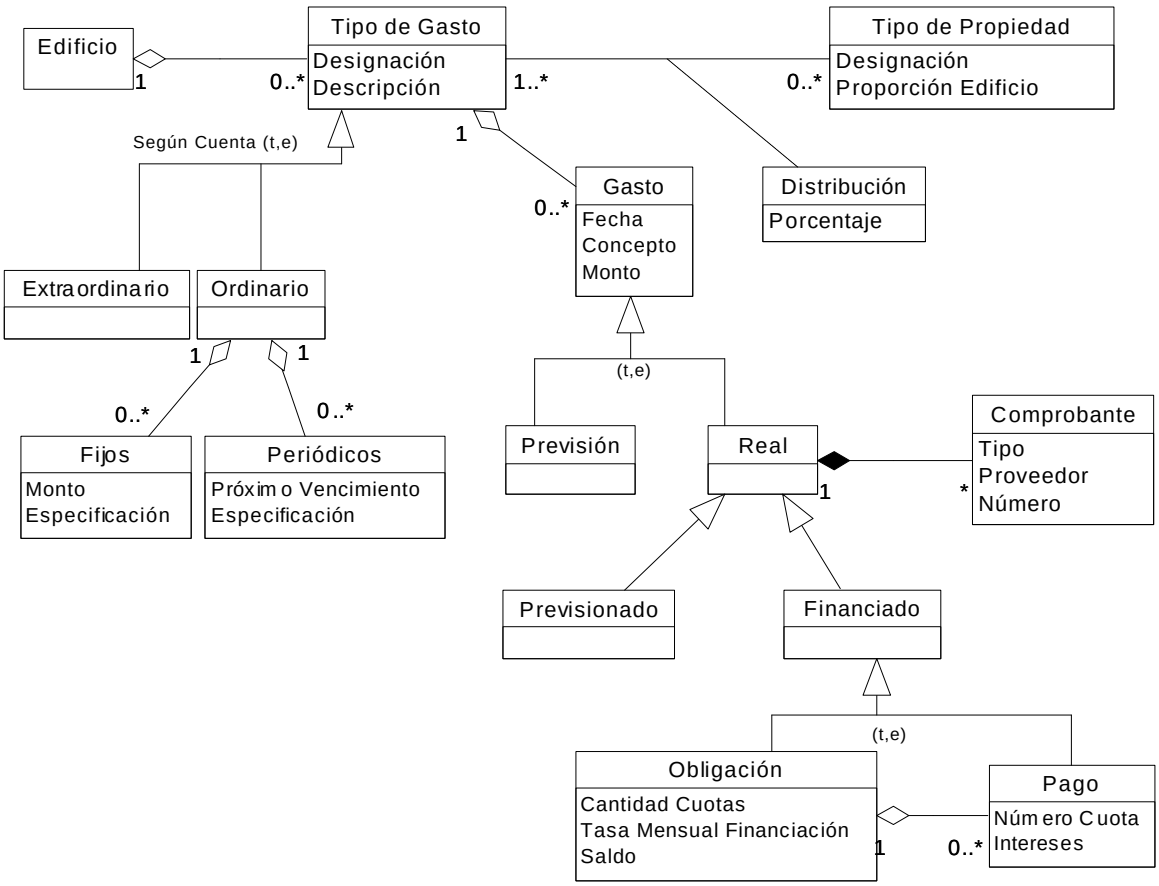


A los efectos de favorecer la visibilidad y comprensión del modelo de dominio, éste se puede organizar en varios diagramas. En este caso, el dominio se divide en tres diagramas: el principal, para definir la composición y personas relacionadas con los edificios, el de erogaciones, para definir los datos que se refieren a gastos corrientes de edificios, y el de recaudación, para definir la información relativa a liquidaciones y cobros de expensas.

Cada edificio tiene su propia clasificación de propiedades en función de su también propia clasificación de gastos. Los gastos de cada tipo se distribuyen entre los tipos de propiedades según un porcentaje de distribución, y proporcionalmente entre las propiedades de cada tipo. Los atributos *Disponible Ordinarios* y *Disponible Extraordinarios* de Edificio representan los saldos de las cuentas de gastos de cada tipo. *Forma Penalizar Moras* determina cómo han de calcularse y liquidarse los intereses punitorios por mora en el pago de las expensas según un número de esquema o modelo. *Distribución Gastos Extraordinarios* indica si la distribución de estos gastos es proporcional a las propiedades (cada una paga según la proporción de la parte con el todo) o uniforme (las unidades de cada tipo pagan todas lo mismo).

Fecha Demanda Judicial en Propiedad determina si han de cobrarse intereses resarcitorios además de punitorios.

Dominio de Erogaciones



Dominio de Recaudación

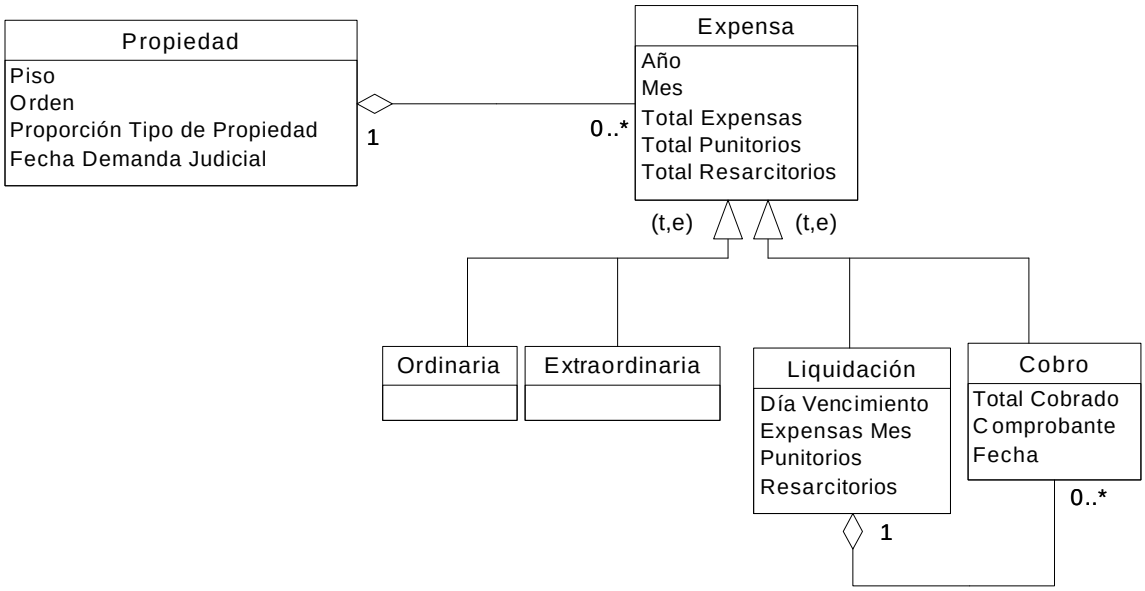
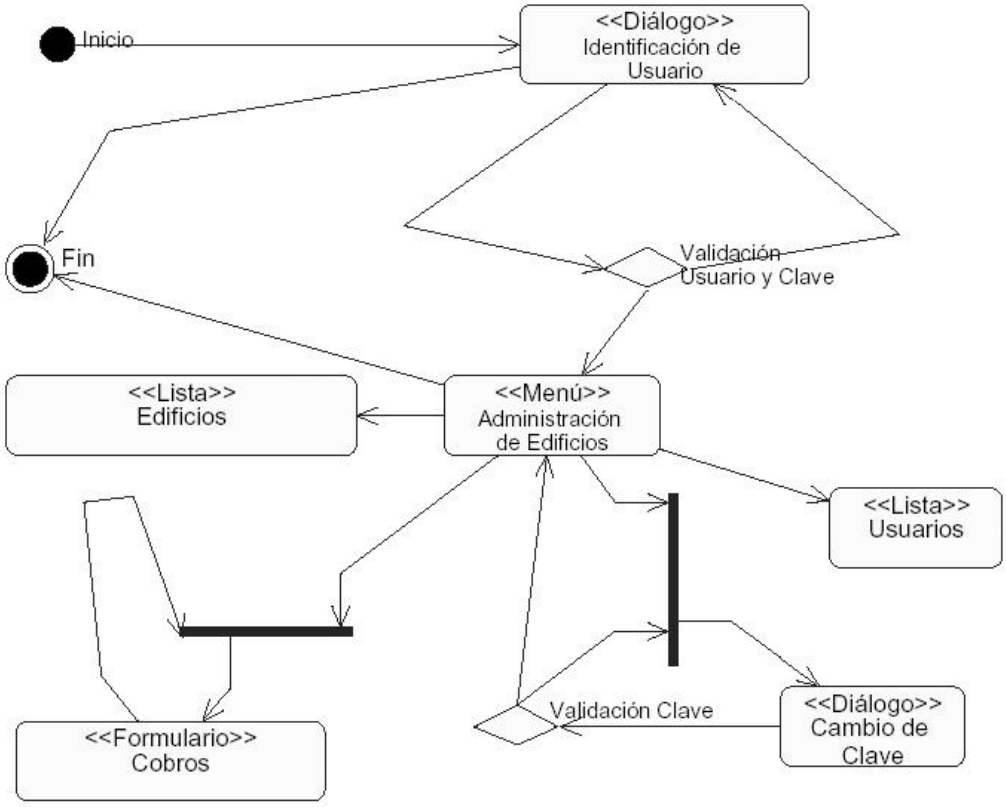


Diagrama Principal de Comportamiento



Subdiagrama de Edificios

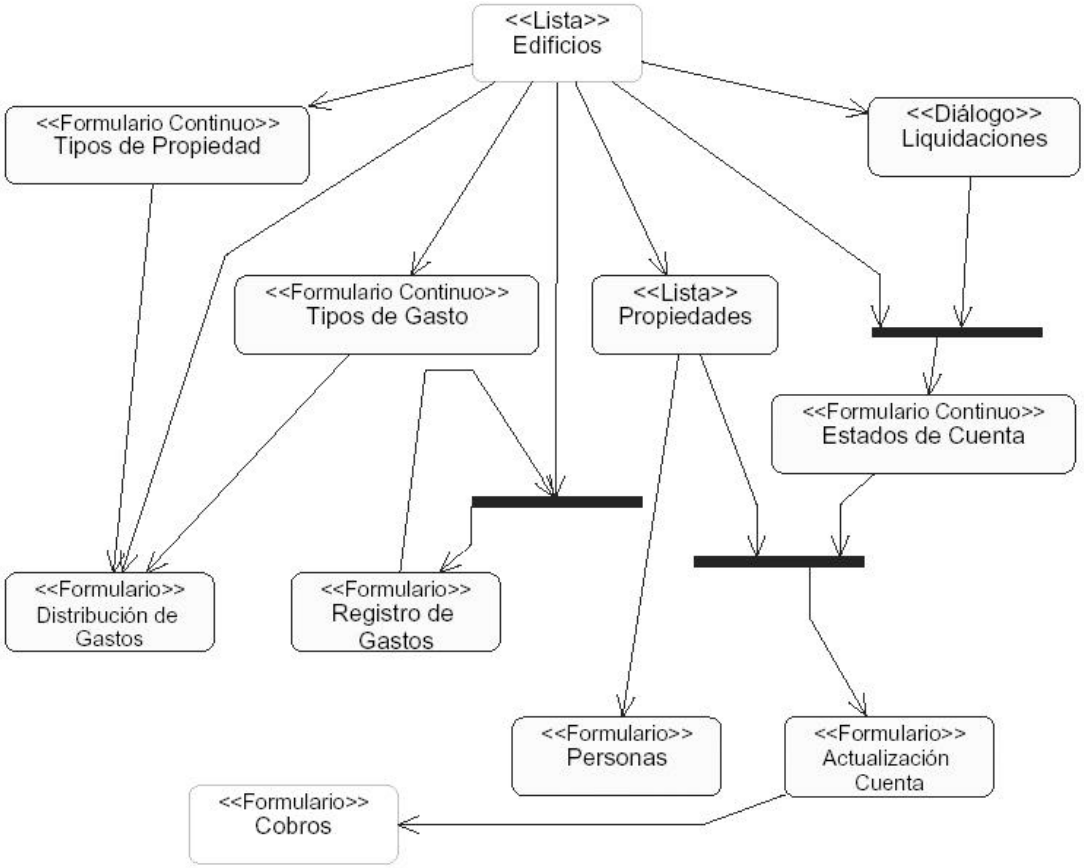
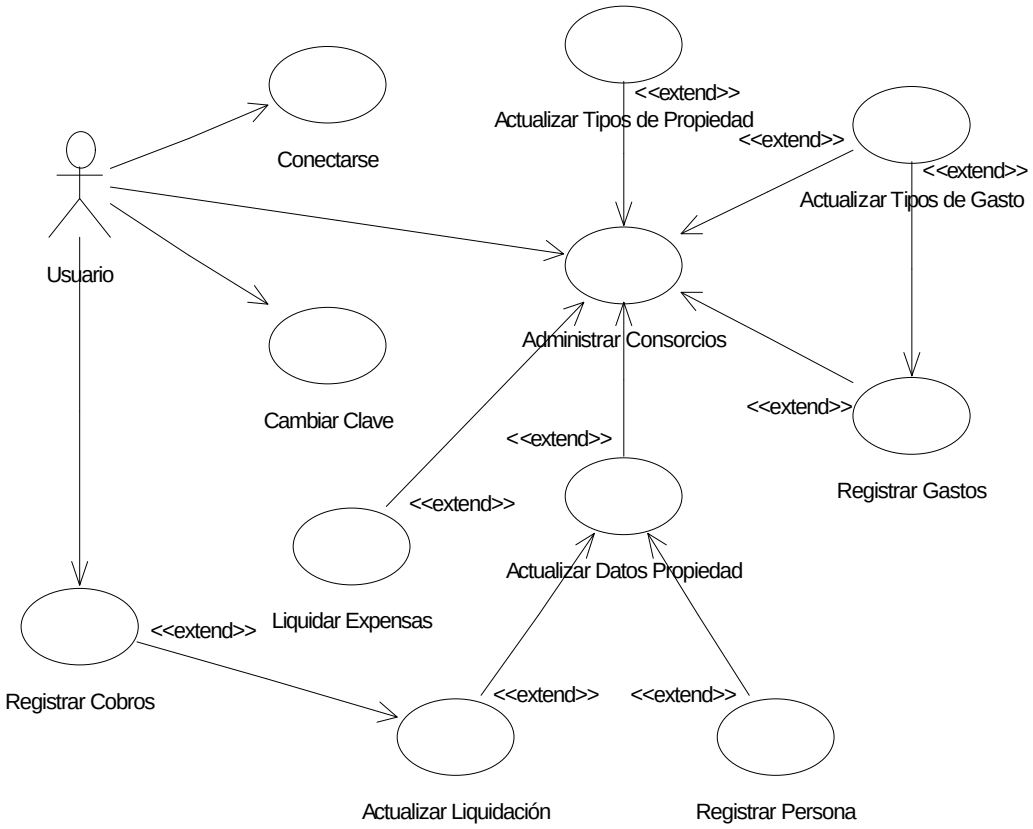


Diagrama de Casos de Uso Equivalente a los de Comportamiento



Medadatos

DTD (Document Type Declaration) de Diagrama de Comportamiento

<!ELEMENT COMPORTAMIENTO (INTERFAZ+, TRANSICIÓN*, DECISIÓN*)+>

<!ELEMENT INTERFAZ (MENÚ | DIÁLOGO | LISTA | FORMULARIO | FCONTINUO | MENSAJE)>

<!ELEMENT MENÚ (OPCIÓN+)>

<!ELEMENT DIÁLOGO (ARGUMENTO+)>

<!ELEMENT LISTA (CUERPO, FORMULARIO)>
<!ELEMENT CUERPO (ATRIBUTO+)>

<!ELEMENT FORMULARIO (SECCIÓN+)>
<!ELEMENT SECCIÓN (ATRIBUTO | RESULTADO | BOTÓN | FCONTINUO)+>

```
<!ELEMENT FCONTINUO          (ATRIBUTO+,  BOTÓN*)+>

<!ELEMENT MENSAJE            (#PCDATA)>

<!ELEMENT TRANSICIÓN         (QBE | CÓDIGO)>

<!ELEMENT QBE                (DISYUNCIÓN+, AGREGACIÓN?, ORDENAMIENTO?)>
<!ELEMENT DISYUNCIÓN         (CONDICIÓN+)>
<!ELEMENT CONDICIÓN          (#PCDATA)>
<!ELEMENT AGREGACIÓN         (ATRIBUTO+, FUNCIÓN)>
<!ELEMENT ORDENAMIENTO       (ATRIBUTO+)>

<!ELEMENT CÓDIGO             (#PCDATA)>

<!ELEMENT DECISIÓN           (#PCDATA)>

<!ATTLIST INTERFAZ           Título       ID       #required
                             Descripción  CDATA    #implied>

<!ATTLIST OPCIÓN             Título       ID       #required
                             Descripción  CDATA    #implied
                             Acción       IDREF    #required>
<!-- Acción refiere a una transición -- >

<!ATTLIST ARGUMENTO          Título       ID       #required
                             Dominio      CDATA    #required
                             Descripción  CDATA    #implied>

<!ATTLIST ATRIBUTO           Nombre       CDATA    #required
                             Origen       IDREF    #required
                             Descripción  CDATA    #implied>
<!-- El origen refiere a una clase definida en el modelo de dominio -- >

<!ATTLIST SECCIÓN            Título       ID       #required>

<!ATTLIST RESULTADO          Título       ID       #required
                             Dominio      CDATA    #required
```

```
                                Descripción   CDATA      #implied>
<!ATTLIST BOTÓN                 Título        ID         #required
                                Descripción   CDATA      #implied
                                Acción        IDREF      #required>
<!-- Acción refiere a una transición -- >

<!ATTLIST FUNCIÓN               Nombre        ( Máx | Mín | Prom | Cont )   #required
                                Argumento     IDREF      #required>

<!ATTLIST DECISIÓN              Título        ID         #required>

<!ATTLIST TRANSICIÓN            Origen        IDREF      #required
                                Destino       IDREF      #required>
<!-- Origen y destino refieren a una interfaz o a una decisión -- >
```

Extracto de Diagrama Principal para Definición de Interfaz Edificios

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE COMPORTAMIENTO SYSTEM “COMPORTAMIENTO.DTD”>
<COMPORTAMIENTO>
...
<INTERFAZ Título=“Edificios”  Descripción=“Edificios Administrados”>
        <LISTA>
                <CUERPO>
                        </ATRIBUTO Nombre=“Nombre”  Origen=“Edificio”
                        Descripción=“Nombre del Edificio”>
                </CUERPO>
                <FORMULARIO>
                        <SECCIÓN Título=“Datos de Administración”>
                                </ATRIBUTO Nombre=“Tasa Mensual Punitoria”  Origen=“Edificio”
                                Descripción=“Tasa por mora en el pago de expensas”>
                                </ATRIBUTO Nombre=“Tasa Mensual Resarcitoria”
                                Origen=“Edificio”  Descripción=“Tasa complementaria por litigio”>
                                </ATRIBUTO Nombre=“Disponible Ordinarios”  Origen=“Edificio”
                                Descripción=“Disponibilidad para gastos ordinarios”>
                                </ATRIBUTO Nombre=“Disponible Extraordinarios”
                                Origen=“Edificio”  Descripción=“ Disponibilidad para gastos
                                extraordinarios”>
                                </ATRIBUTO Nombre=“Inicio Administración”  Origen=“Edificio”
                                Descripción=“Fecha desde que se administra el edicicio”>
```

```
                    </ATRIBUTO Nombre="Forma Penalizar Moras"  Origen="Edificio"
                    Descripción="Modalidad de cálculo y liquidación de punitorios">
                    </ATRIBUTO Nombre="Distribución Gastos Extraordinarios"
                    Origen="Edificio"  Descripción="Forma de prorrateo de gastos
                    extraordinarios">
                </SECCIÓN>
                <SECCIÓN Título="Dirección del Edificio">
                    </ATRIBUTO Nombre="Calle"  Origen="Edificio.Dirección">
                    </ATRIBUTO Nombre="Ubicación"  Origen="Edificio.Dirección"
                    Descripción="Númeración en la Calle">
                    </ATRIBUTO Nombre="Localidad"  Origen="Edificio.Dirección">
                    </ATRIBUTO Nombre="Teléfono"  Origen="Edificio.Dirección"
                    Descripción="Nro. de Portería o de Responsable en el Edificio">
                    </ATRIBUTO Nombre="Código Postal"
                    Origen="Edificio.Dirección">
                </SECCIÓN>
            </FORMULARIO>
        </LISTA>
</INTERFAZ>
...
</COMPORTAMIENTO>
```

Estado de Avance del Proyecto y Trabajos Futuros

Se completó el diseño de la herramienta y ya se ha comenzado su desarrollo. Los metadatos asociados a los diagramas de especificación se modelaron separando la información de definición de la información de representación, y se prevé como futuro proyecto agregar a la herramienta el manejo de repositorios con control de versiones.

Para la modelación estática también se trabaja como alternativa la modelación conceptual de entidades y relaciones con una extensión del modelo propuesta por los autores Batini, Ceri y Navathe [3][15] de gran riqueza expresiva, y que admite la definición de atributos compuestos, atributos polivalentes, cardinalidades mínima y máxima de entidades en relaciones y jerarquías de generalización con cobertura.

También se definieron métricas de calidad, tanto para las estructuras de datos como para las funciones, para validar y orientar el diseño, y métricas de complejidad para la valorización de los sistemas que se produzcan.

Conclusiones

Lo original del proyecto es la definición del proceso ágil o de ingeniería liviana, la aplicación de la teoría de autómatas finitos para sintetizar especificación de requerimientos y modelar comportamiento de sistemas, y la definición del álgebra relacional de entidades conceptuales y su extensión para la especificación de guiones funcionales.

Se cree que esta herramienta importa una contribución para la comunidad informática dedicada al desarrollo de sistemas de gestión, dado que implica la adopción de una metodología simple y

precisa que favorece la participación de los usuarios finales y mantiene a todo desarrollo permanentemente documentado.

La participación y el compromiso de los usuarios finales en desarrollos basados en esta herramienta se presumen garantizados debido a que los modelos empleados para las especificaciones son de un alto nivel de abstracción y comprensibles para personas no especializadas; además el modelo dinámico, tal como el de casos de uso en el Proceso Unificado de Desarrollo permite verificar la completitud y rastrear el cumplimiento de requerimientos, con la posibilidad de la prototipación temprana, asequible con la generación de sistemas a partir de la especificación del diseño de interfaces, optimiza las relaciones contractuales facilitando la aprobación de fases y ciclos de evolución.

Referencias

[1] Anfossi, D. y Servetto, A. “Relevamiento de Herramientas CASE Orientadas a Objetos: Comparación, Evaluación y Conclusiones”. Anales del III ICIE (Congreso Internacional de Ingeniería en Informática) de la Facultad de Ingeniería de la Universidad de Buenos Aires; pág. 449-463.

[2] Bass, L. , Clements, P. and Kazman, R. "Software Architecture in Practice". Addison - Wesley. 1998.

[3] Batini, C., Ceri, S., Navathe, S. “Diseño Conceptual de Bases de Datos: Un enfoque de Entidades-Interrelaciones”. Addison-Wesley Iberoamericana, 1991; ISBN 0-201-60120-6.

[4] Bell, D. and Morrey, I. "Software Engineering, Second Edition".Prentice Hall.1992.

[5] Booch, Grady; Rumbaugh, James; Jacobson, Ivar. “The Unified Modeling Language - User Guide”. Addison-Wesley, 1999.

[6] Brown, A. W. & McDermid, J. A., 1991, On Integration and Reuse in a Software Development Environment, Software Engineering Environments ‘91, F. Long & M. Tedd (Editors), Ellis Horwood, Mar 7.

[7] Constantine, L. and Lockwood, L. "Software for use". A Practical Guide to the Models and Methods of Usage - Centred Design. Addison - Wesley. 1999.

[8] Fowler, M. And Highsmith, J. “The Agile Manifiesto”. Software Development Magazine. Agosto 2001.

[9] Ghezzi, C. , Jazayeri, M. , Mandrioli, D. "Fundamentals of Software Engineering". Prentice Hall. 1991.

[10] Grimaldi, Ralph P. “Matemáticas discreta y combinatoria. Introducción y aplicaciones”. Addison-Wesley Iberoamericana, 1989.

[11] IEEE STD 1074-1991 Customer Request IEEE Standard fo developing Software Life Cycles Process, Identify Ideas or Needs, Preliminary Statement of Need, Feasibility Studies, Statement of need.

[12] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. “The Unified Software Development Process”. Addison-Wesley ISE. 1999.

[13] Pfleeger, S. "Software Engineering: Theory and Practice". Prentice Hall. 1998.

[14] Rosenberg, D. And Scott, K. “Use Case Driven Object Modeling with UML”. Addison-Wesley, 1999.

[15] Servetto, A. “Mecanismos de Abstracción en la Modelación Conceptual de Datos y su Aplicación en una Ampliación del Modelo de Entidades y Relaciones” 3ras Jornadas de

Informática e Investigación Operativa de la Universidad de la República de Uruguay (12 al 14 de diciembre de 1996).

[16] Sommerville, I. "Software Engineering". Addison - Wesley. 1996.

[17] Thomas, I. & Nejmeh, B. A., 1992, Definitions of Tool Integration for Environments, IEEE Software, 9, 2 (Mar), 29-35.

[18] Wasserman, A., 1990, Tool Integration in Software Engineering Environments, Software Engineering Environments: Proceedings of the international Workshop on Environments, F. Long (Editor), Springer-Verlag , 137-149.

[19] Wasserman. A. I., 1988, “Integration and Standarization Drive CASE Advancements, Computer Design”, 27, 22 (Dec), 86.

[20] **http://www.agilealliance.org/**