3. Una propuesta de Modelo Conceptual para Modelado de Procesos

El objetivo de este capítulo es definir un modelo conceptual útil para comprender y soportar a la modelización de procesos. Para una mejor comprensión de esta parte, la estructuraremos del siguiente modo:

- primero realizamos una definición, bajo un enfoque descriptivo, de los diferentes conceptos que intervienen en procesos de software y modelado de los mismos, remitiendo al lector a distintas fuentes de la literatura investigada. El abordaje no pretende ser amplio (en cuanto a la cantidad de conceptos tratados[2]) sino el necesario para facilitar la comprensión del marco conceptual del punto 3.3.

- luego, presentamos un diagrama estático de las principales responsabilidades que intervienen en un proyecto de hipermedia (contextualizado para el modelo de proceso flexible).

- por último, presentamos un modelo conceptual para el modelado de procesos de software. Contar con una base que contenga un conjunto canónico de conceptos de modelado de procesos potencia, por una parte, la combinación de notaciones para describir a los procesos, y, por otra parte, posibilita la creación de capacidades en ambientes que soporten multi-paradigmas para ejecutar procesos.

3.1 Conceptos sobre Procesos de Software y Modelado de Procesos

Uno de los propósitos de esta tesis, según indicamos en el punto 1.1, es consolidar un modelo conceptual de clases y relaciones esenciales que sirva de base para investigaciones presentes y futuras. Dijimos que el conocimiento público de caracter científico, se ve favorecido si se

[2] No obstante, el lector cuenta con un glosario, el cual abarca mayor cantidad de conceptos y referencias que el que abordaremos en la siguiente sección

identifica y comunica, de un modo no ambiguo y representativo, a un conjunto de conceptos primitivos.

En ingeniería de procesos de software se manejan conceptos como el de proceso, tarea, actividad, agente, artefacto, recurso, rol, constructores de proceso, descripción de proceso, modelo de proceso, perspectivas de proceso, arquitectura de proceso, ambientes de ingeniería de software orientados a procesos, entre otros conceptos. Esfuerzos importantes se han venido realizando con el fin de establecer una base conceptual sólida [Curtis et al 92, Dowson et al 91, Feiler et al 93, Lonchamps 93, Madhavji 91]. Trabajos seminales en el área son los de Humphrey, Feiler y Dowson et al. Pensamos que nuestro modelo conceptual (fig. 3. 7), por medio de un diagrama de clases y relaciones, contribuye a representar y sintetizar de un modo no ambiguo el dominio del problema.

A seguir introduciremos los conceptos[3] usando un enfoque lexicográficamente descriptivo.

Una **fase** es una agrupación de *procesos de software* fuertemente relacionados o cohesivos realizados en cierto orden. En el esquema del *modelo de proceso* introducido en la sección 2.4 se puede apreciar tres fases: la de exploración, la de desarrollo y la operativa.

Humphrey ha dicho que un **proceso de software** (o proceso, para abreviar) es un conjunto parcialmente ordenado de *subprocesos* que tienen el fin de alcanzar alguna meta establecida. De un modo mas amplio podemos definir a un proceso de software como a un conjunto parcialmente ordenado de subprocesos a los que se le asocian una colección de *recursos*, *agentes*, *condiciones*, *artefactos* y *constructores de proceso*, con el fin de producir los *distribuibles* conforme a las metas establecidas.

Por **distribuible** entendemos a un artefacto (producto) solicitado por algún proceso o agente interno o externo.

Ejemplos de nombres de procesos a cierto nivel de granularidad son: administración de proyecto, determinación y especificación de requerimientos, análisis conceptual, diseño navegacional, diseño de interfaces abstractas, determinación y resolución de riesgos, análisis de factibilidad, prototipación, validación, etc.

[3] Utilizaremos la/s palabra/s resaltada/s en **negrita** para describir un nuevo concepto, con *itálica* los conceptos a describir en párrafos subsiguientes y, en esta sección, usaremos paréntesis para referirnos a (sinónimos).

Un proceso se puede descomponer en **subprocesos** (paso o elemento de proceso), por lo tanto le cabe una definición recursiva. El nodo raíz del árbol es el mas general y abstracto, el nodo hoja es el más específico y lo denominamos *actividad*. Por lo que una **actividad** es un subproceso que no requiere mas descomposición. Se dice que una actividad es un elemento de proceso descripto a un bajo nivel de granularidad (granularidad fina)

Es oportuno destacar que en la literatura se ha establecido una distinción, a veces sutil, entre el concepto de actividad y el de **tarea**. En el área de administración de proyectos una tarea es una unidad de trabajo a realizar por un agente. Una tarea es un subproceso a la que se le asocian componentes de gestión, es decir, se le pueden asignar agentes, recursos, se la puede planificar, programar, ejecutar y controlar.

En el área de ingeniería de procesos de software también se ha utilizado a la tarea como a la entidad atómica de abstracción, sobre todo en enfoques prescriptivos y orientados al análisis, como apunta Lonchamp. Sin embargo existen enfoques en que a las tareas se las dividen en actividades, siendo las actividades entes que no necesitan ser gerenciados (sobre todo en entornos o herramientas de automatización). Por lo tanto en el caso en que se establezca la diferencia entre tarea y actividad, ésta corresponderá a que un proceso pueda ser o no gerenciado por un agente. En la fig. 3.1 observamos la relación entre entes que representan a las definiciones previas de proceso, actividad y tarea.

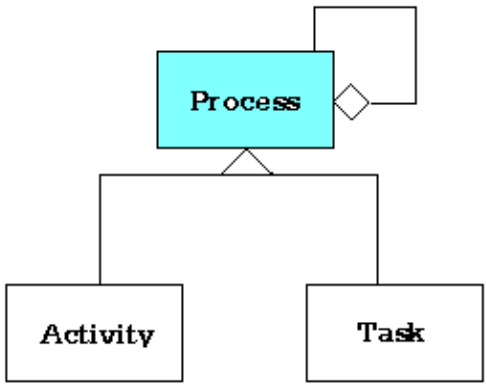

Fig. 3.1 *Diagrama que relaciona a las clases Proceso, Tarea y Actividad.*

En cuanto al concepto de **agente** lo definimos como al ente ejecutor de un proceso. El agente puede ser tanto un ente humano como una herramienta o dispositivo computarizado. Podemos realizar una taxonomía de tareas en función de los agentes teniendo en cuenta que una tarea se puede llevar a cabo por uno o mas agentes. Clasificamos a las tareas en tres tipos:

- manuales
- automáticas
- interactivas

Las tareas manuales solo involucran a agentes humanos; las tareas automáticas solo dependen de agentes computarizados, en tanto que las últimas necesitan de ambos tipos de agentes con el fin de alcanzar la submeta establecida. Para llevar a cabo un proceso el agente humano o computarizado debe contar con un conjunto de habilidades (experticia). Para cumplimentar un *rol* se deben requerir habilidades específicas por parte de los agentes. Por ejemplo, un agente humano debe tener experticia sobre criterios estéticos y cognitivos de interfaces de usuario y aspectos de diseño del look and feel para cumplimentar el rol de "Diseñador de Interfaces".

Un **artefacto** es el producto de realizar una tarea. Un artefacto es el producto creado, evolucionado, mantenido o destruido durante el proceso de desarrollo ya como un resultado requerido por un agente o para facilitar la prosecución de otro proceso. Con lo dicho podemos razonar que un artefacto puede servir de entrada a un proceso y, mediante la transformación correspondiente, ser la salida del mismo. Además un artefacto puede ser un objeto compuesto, es decir, se da una relación de agregación entre componentes.

En nuestra visión los procesos de software fundamentalmente existen para crear, modificar y reusar artefactos.

Entre los objetos considerados artefactos se encuentran los documentos -por ejemplo: documento del plan del proyecto, especificaciones de análisis y diseño, prototipos y versiones finales de productos -código fuente y/o ejecutable-. Los artefactos pueden ser administrados bajo una estrategia de configuración de cambios.

En cuanto a un **recurso** es un ente del mundo real necesario para que las tareas de un *proyecto de software* se puedan efectuar. Una tarea usa recursos. Ejemplos de recursos son: recursos humanos, monetarios, materiales, tecnológicos. En la figura 3.2 mostramos una jerarquía general de recursos usando una notación de [Rumbaugh et al 91].

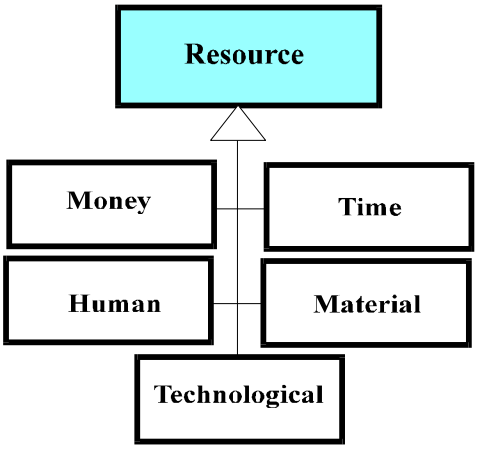

Fig. 3.2 *Diagrama general de jerarquía de recursos.*

Un **rol** es un conjunto de permisos y obligaciones que se debe asociar a un agente durante la realización de un tipo de tarea. El agente debe tener un conjunto de permisos para realizar las actividades de la tarea conforme a la submeta establecida y obligado a satisfacer un conjunto de *condiciones.*

Se puede establecer una jerarquía de roles: roles más generales y roles más específicos. En la figura 3.3 observamos que el rol de diseñador de interface es más específico que el rol de desarrollador de software.

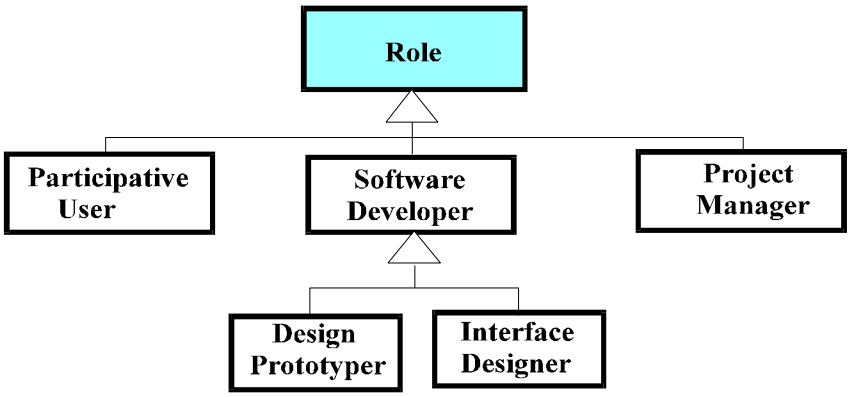

Fig. 3.3 *Ejemplo reducido de jerarquía de roles de un proyecto de hipermedia.*

En la página 493 de [Goldberg et al 95], los autores presentan un diagrama -de Venn- en donde se muestra los roles ocupados por agentes, agrupados estos en equipos de trabajo. Cuando un proceso se está llevando a cabo puede existir una asignación dinámica de roles a agentes y

viceversa. Los agentes pueden ocupar (jugar) diferentes roles a diferentes momentos, un rol dado puede ser ocupado por distintos agentes a diferentes momentos [Dowson et al 91].

Una **condición** de un proceso es la declaración del estado de situación que debe acontecer para el inicio, ejecución y finalización de un proceso. Una actividad puede comenzar cuando se cumple un conjunto de precondiciones y puede finalizar cuando se alcanzan las postcondiciones establecidas.

Un **constructor de proceso** es un enfoque específico (método) que se puede usar para realizar una tarea. Ampliando el concepto podemos decir que para llevar a cabo una tarea especificada en la *descripción de proceso* se pueden usar diferentes métodos que responden a distintos enfoques (ver fig. 3.4).

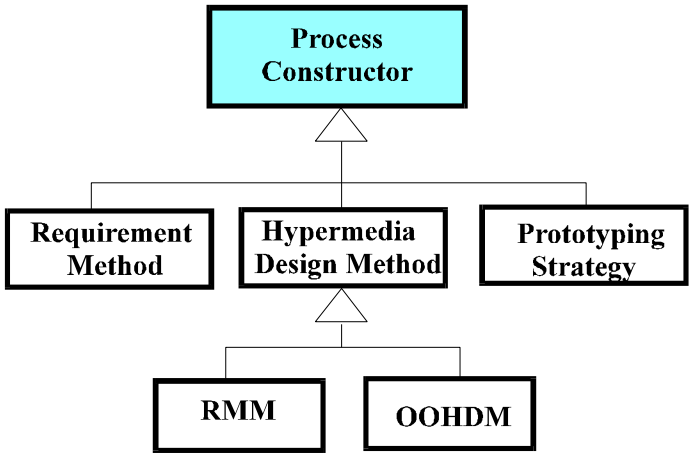

Fig. 3.4 *Ejemplo reducido de jerarquía de constructores de proceso de un proyecto de hipermedia.*

Por ejemplo, para una tarea de "modelado del dominio de la aplicación", podemos usar el método especificado en RMM, denominado "diseño de E-R", o podemos usar para esa tarea el método especificado en la metodología OOHDM, denominado "modelado conceptual". Son enfoques diferentes para una descripción de proceso dada.

En el trabajo seminal sobre el área de ingeniería de procesos, Osterweil dijo [Osterweil 87] que los procesos de software son también software. Si bien el enfoque era formal, centrado en descripciones procedurales y declarativas, hay diferentes paradigmas para describir a los procesos. Primero veamos el concepto.

Una **descripción de proceso** es una manera de representar y especificar la secuencia parcial de actividades de un proceso. Una descripción completa de proceso debe considerar las actividades y las operaciones asociadas, las precondiciones y postcondiciones para cada actividad, y otros entes (objetos) intervinientes en el proceso de software como artefactos, agentes y roles. Pueden coexistir descripciones alternativas para un proceso y se puede considerar a una descripción de proceso como a un artefacto especial.

Una descripción de proceso está para que pueda ser interpretada y ejecutada por un agente humano o un agente computarizado. Un área de investigación atractiva es la de soporte a procesos de software. Se han realizado algunas contribuciones y avances, en el área de *ambientes de ingeniería de software centrado en procesos*. En estos sistemas muchas de las tareas son interpretadas y ejecutadas automáticamente por un *motor de procesos*. Puede coexistir para una cierta actividad descripciones alternativas bajo un mismo enfoque, como indicamos anteriormente, o descripciones multi-paradigmáticas.

Los aspectos de mayor o menor formalidad en la descripción de procesos es una cuestión de relevancia. Una representación precisa, formal, es ejecutable por una máquina, en tanto que una representación informal, esto es en lenguaje natural, o una representación semiformal estructurada en un script, puede ser ejecutada por un agente humano y no por un agente computarizado - aunque en algunos casos una representación semiformal puede ser interpretada.

En la figura 3.5 el lector puede observar una taxonomía de descripciones de procesos en función de la formalidad de su representación. Otra taxonomía se podría realizar o complementar teniendo en cuenta el nivel de granularidad de las actividades descriptas. Los autores en [Curtis et al 92], han expresado que la experiencia sugiere que los procesos automatizados requieren una representación de granularidad más fina, en tanto que una representación semiformal como los script, de granularidad más alta, son adecuadas para ser interpretadas y ejecutadas por agentes humanos.

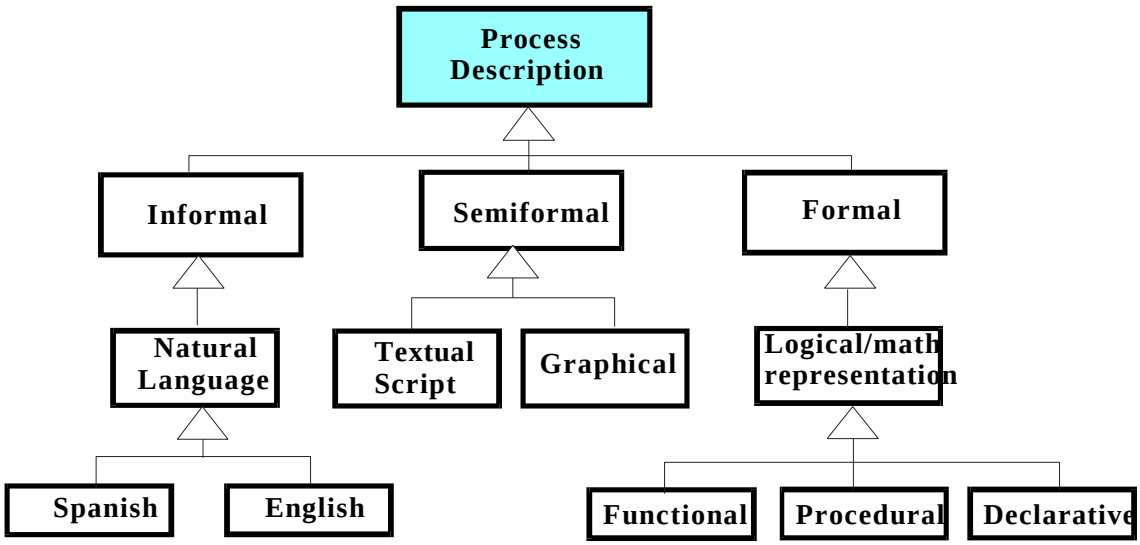

Fig. 3.5 *Diagrama reducido de tipos de descripciones de proceso en función de la formalidad de su representación.*

Relacionado a la representación de las descripciones de procesos y a las distintas *perspectivas* de un *modelo de proceso* está el concepto de *lenguajes de modelado de procesos*, que describiremos en párrafos subsiguientes.

Los procesos de software se pueden abstraer en un **modelo de proceso** (modelo de ciclo de vida). Definimos a un **modelo** como a una representación abstracta de entes o fenómenos de la realidad en la que se consideran los aspectos relevantes de los mismos y se desechan los menos relevantes sin que por ello deje de representar significativamente a esa realidad.

Definimos a un modelo de proceso de software como a una estrategia apropiada para abstraer, organizar, ejecutar y/o controlar a las distintas fases, tareas, recursos y artefactos de un proyecto con el objeto de alcanzar las metas establecidas. Un modelo de proceso es una descripción más o menos formal del proceso de desarrollo de software. Por lo que un modelo de proceso expresa: 1) un cierto nivel de abstracción; 2) una *perspectiva* particular del proceso de desarrollo.

Modelos de ciclo de vida como los discutidos en la sección 2.2 -modelo de cascada, espiral y recursivo/paralelo-, son modelos muy abstractos y generalmente semiformales. No se especifica con suficiente detalle y rigor formal, las entradas y salidas de las actividades, condiciones de comienzo y finalización, flujos de artefactos, sincronización de actividades, entre otros asuntos.

Un modelo de proceso integra diferentes tipos de información de los procesos de software. Una **perspectiva** (vista o submodelo) es un enfoque particular para especificar y comunicar información del modelo. En la sección 2.3 dijimos que un modelo de proceso debe ayudar a responder preguntas tales como: qué hacer, cómo y cuándo hacerlo, dónde y quiénes lo harán, qué dependencias existirán entre tareas y cómo se sincronizarán, etc.

Curtis et al agrupan la información en cuatro perspectivas con el fin de abstraer la complejidad subyacente en los procesos de software, y son, a saber: *perspectiva funcional*, *perspectiva informacional*, *perspectiva de comportamiento* y *perspectiva organizacional.* Pensamos que una *perspectiva metodológica* es fundamental para representar a los diferentes métodos (constructores de proceso) y, además, puede ser integrado en *ambientes de ingeniería de software centrado en procesos.*

La **perspectiva funcional** representa qué tareas se deben realizar en cada fase, qué estructura jerárquica de actividades existe para cada tarea, y qué dependencias funcionales -flujos de información y control, documentos, artefactos-, son relevantes para las entradas y las salidas.

La **perspectiva de información** se centra en los artefactos producidos o requeridos por los procesos de software; en la estructura de los artefactos y sus interrelaciones; en la estrategias de administración de cambios de artefactos y seguimiento (traceability) de los mismos.

La **perspectiva de constructores de proceso** representa los métodos a aplicar para efectuar las distintas actividades especificada en la descripción de procesos. Para un método en particular puede haber una o más herramientas que den soporte a la tarea.

La **perspectiva de comportamiento** representa aspectos dinámicos del modelo de proceso: el secuenciamiento y la sincronización de las tareas; información de cómo se realizan las actividades, esto es, iteraciones, ciclos de retroalimentación, paralelismo, condiciones de inicio y terminación, entre otros asuntos. Asimismo puede especificar el ciclo de vida de un ente como un artefacto, proceso, agente con formalismos como: diagrama de transición y estados, statecharts, ADV-charts, etc.

En la **perspectiva organizacional** se consideran aspectos de qué agentes participan/planifican/ejecutan y controlan a qué tareas; qué roles se asignan a los agentes participantes; qué estrategias de comunicación y de dinámica de grupos se aplican, etc.

Por lo tanto, un modelo de proceso integral y completo, necesita de una **arquitectura de proceso** (meta-modelo de proceso) para conectar a las diferentes perspectivas. Una arquitectura de proceso es una estructura conceptual útil para describir las perspectivas relevantes y sus relaciones.

Un **lenguaje de modelado de procesos** es un paradigma capaz de representar a un esquema de meta-modelo de proceso. Como todo lenguaje debe proveer una sintaxis precisa y una semántica no ambigua y amplia. Desafortunadamente, es difícil contar con un lenguaje que represente a las distintas perspectivas de un modelo de proceso y provea constructores de soporte a las distintas actividades de modo que pueda ser interpretado y ejecutado con eficacia en diferentes *proyectos de software*. En las práctica, la mayoría de las descripciones de proceso, cuando existen, se basan en mecanismos informales o semiformales difíciles de ser ejecutados en *ambientes centrados en procesos*.

No obstante algunos aspectos de un modelo de proceso pueden ser automatizados. Curtis et al presentan un conjunto de tipos de lenguajes que responden a distintos paradigmas, que pueden ser útiles para representar a las diferentes perspectivas de un modelo de proceso. Si bien este trabajo requeriría ser actualizado, creemos que es válido para obtener una visión inicial del estado del arte.

Un **ambiente de ingeniería de software centrado en procesos** (ambiente (o sistema) de soporte a procesos de software) es un entorno de trabajo que ofrece asistencia a los usuarios en la ejecución de un *proyecto de software*. Un componente esencial del ambiente es el **motor de procesos** el cual es el intérprete automático de las descripciones de proceso provistas de un modo estático o dinámico. Con dinámico queremos significar que durante la ejecución del proceso la descripción puede cambiar (evolucionar).

La arquitectura de un ambiente de software centrado en procesos debe ofrecer servicios de administración de objetos e interrelaciones, servicios de herramientas, servicios de trabajo colaborativo, y otros como administración de cambios, versionamiento, seguimiento de artefactos, seguridad.

La realización de un **proyecto de software** comprende a un conjunto de tareas, tanto técnicas como de gerenciamiento, a un conjunto de recursos, a un conjunto de estrategias, métodos y heurísticas, con el propósito de lograr los objetivos y las metas acordadas. Un proyecto de

software puede ser autocontenido o parte de otro proyecto mayor y está inserto en el marco de alguna estrategia organizacional.

3.2 Principales componentes de un Proyecto de Software de Hipermedia

En la figura 3.6 representamos con un diagrama de clases, relaciones y subsistemas, los componentes intervinientes en un proyecto de software de hipermedia. El fin del diagrama es agrupar un conjunto de responsabilidades y colaboraciones -usando una notación ampliada a la de [Rumbaugh91], y representar de un modo estático al proyecto de hipermedia en el contexto del modelo de proceso flexible.

Se puede apreciar los siguientes componentes:

1. el de *modelo de proceso* el cual representa o abstrae al proyecto. Un proyecto de software de hipermedia debe adoptar un modelo de ciclo de vida (en caso de no adoptar uno se guiará por una estrategia ad-hoc).
2. el módulo de *tarea* que representa a la unidad fundamental de gerenciamiento de un proyecto. Una tarea se la puede planificar, programar, ejecutar y controlar.
3. *el de meta y objetivos*, que representa a un conjunto de declaraciones de los resultados que se desean alcanzar en el contexto de la estrategia organizacional.
4. *el de recurso*: tecnológicos, humanos y otros. Los primeros están conformados por los componentes de software y hardware, los que automatizan parcialmente fases y tareas; los segundos por los agentes humanos, los que se le asocian diferentes roles en el transcurso del proceso. Desde otro punto de vista el proyecto está sujeto a restricciones de recursos monetarios, materiales, humanos, etc.
5. el de *constructor de proceso* para las distintas fases, tareas y actividades. Para un proyecto de hipermedia se requiere genéricamente: un modelo de Plan del Proyecto, un método para Análisis de Requerimientos, un método de Diseño, una estrategia de Constucción e Integración, criterios Cognitivos y Estéticos para la construcción de interfaces y espacios de navegación, una estrategia de Aseguramiento de la Calidad.
6. el *repositorio de artefactos de software* el cual sirve de entrada y de salida a las tres fases del modelo de proceso flexible (que discutiremos en el capítulo 4). Los artefactos del proyecto debieran estar sujetos a una estrategia de Configuración de Cambios, de todos aquellos ítems que se establezcan como configurables.

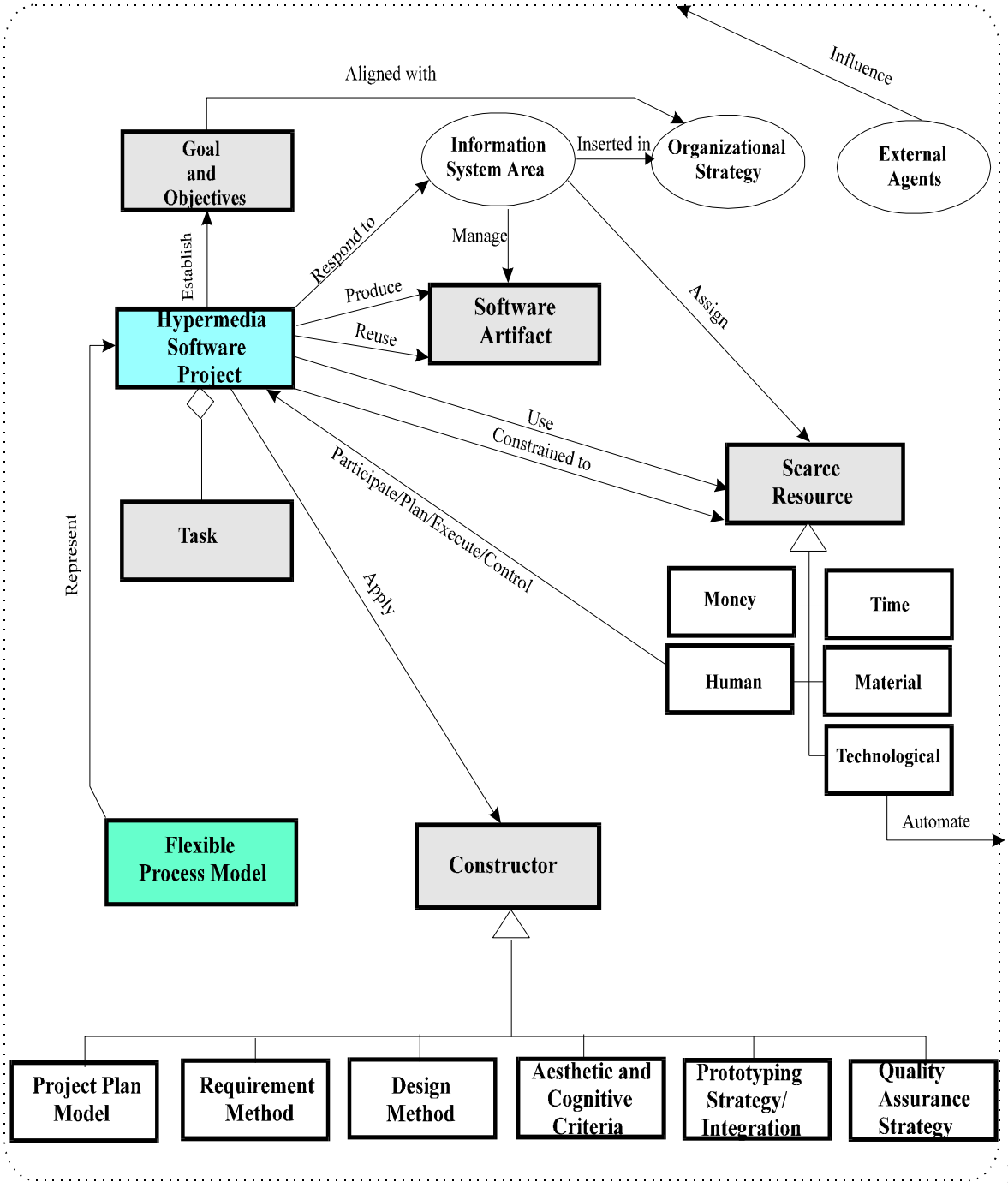

Fig. 3.6 *Visión estática de un proyecto de hipermedia como parte del Modelo de Proceso Flexible.*

3.3 Definición de un Modelo Conceptual

En esta sección presentamos un modelo conceptual para el modelado de procesos de software. Pensamos que al definir un marco base que contenga a un conjunto canónico de conceptos que intervienen en modelado de procesos, potencia a las investigaciones en el área, al permitir:

- la posibilidad de combinar diferentes clases y relaciones para dar soporte a las distintas perspectivas.

- la posibilidad de combinar formalismos y/o notaciones para describir a los procesos que, en definitiva, propenderá a la estandarización de procesos (sección 1.1).

- la posibilidad de crear capacidades en ambientes de ingeniería de software centrado en procesos para que soporten multi-paradigmas en la ejecución de actividades.

Del análisis de la literatura investigada sobre el área, y de las observaciones iniciales efectuadas en los proyectos de desarrollo de hipermedia en que hemos participado, se puede indicar que:

- no existe una única notación universal para especificar a todos los aspectos de un modelo de proceso.

- como consecuencia se requiere un lenguaje multi-paradigma para describir y especificar a los distintos entes y relaciones. El ambiente debiera soportar los diferentes tipos de descripción de procesos (formales, semiformales e informales), para ser ejecutados por agentes humanos o automáticos.

- Dada la complejidad de los elementos y fenómenos que intervienen en un proyecto de software, es importante centrar la atención en las perspectivas del modelo de proceso para ofrecer facilidades de ambiente.

- Un ambiente debe proveer capacidad de redefinición dinámica de descripciones para soportar aspectos evolutivos del proceso de desarrollo.

Algunas de las hipótesis e indicaciones previas serán motivo de investigaciones futuras.

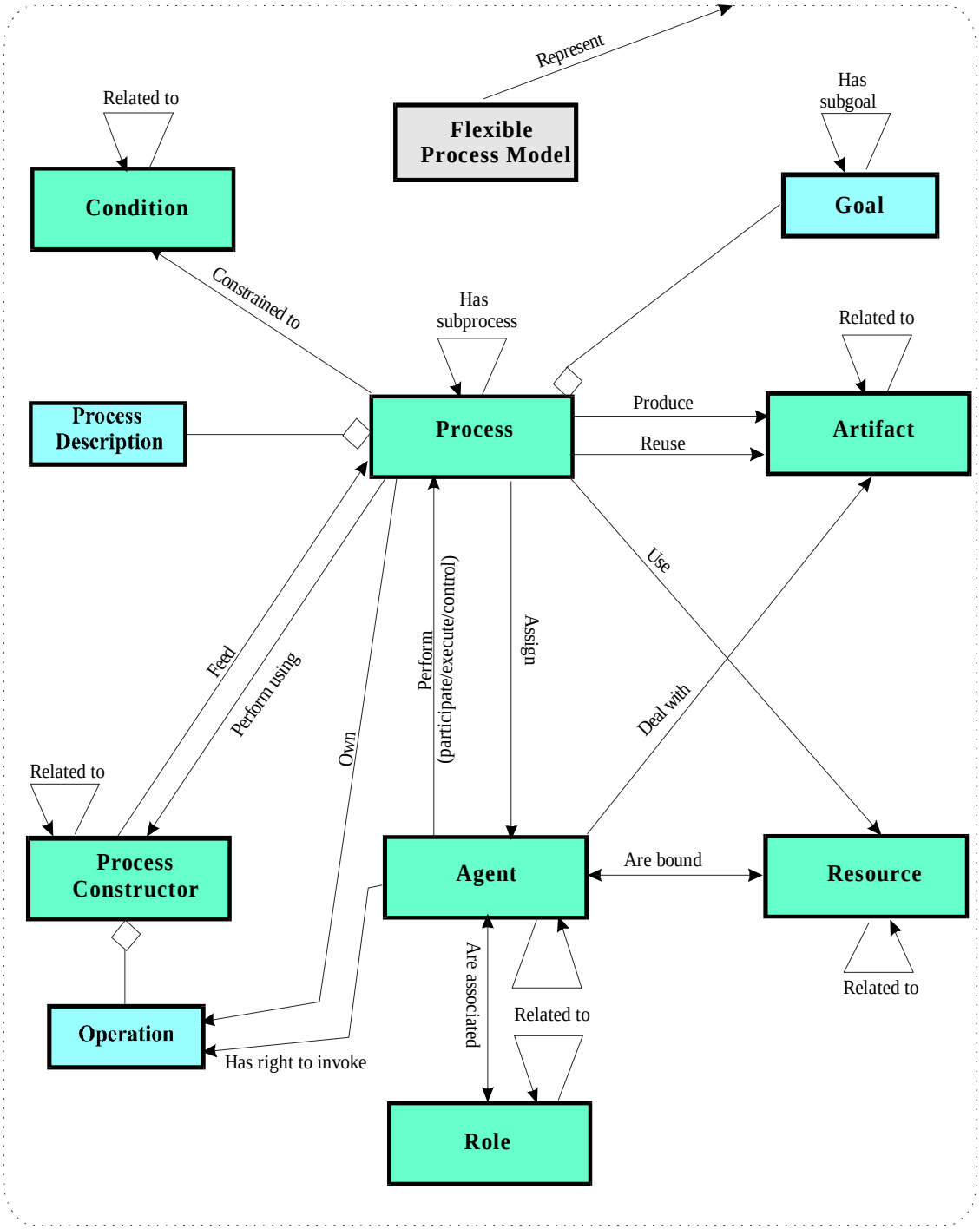

Fig. 3.7 *Diagrama de clases y relaciones fundamentales para un Modelo de Proceso Flexible.*

El objetivo de esta parte del trabajo consiste en presentar un modelo conceptual especificado mediante un diagrama de clases y relaciones fundamentales (figura 3.7), soportado por los

principios de Orientación a Objetos.

Las clases (o conceptos clave para nuestro dominio del problema que es el modelado de proceso) representan comportamiento y estado. En definitiva disponen de métodos para la ejecución de operaciones y encapsulan atributos del ente. Existen métodos denominados constructores y destructores entre otro tipos de métodos. Por ejemplo una clase artefacto puede tener como atributos su identidad, su fecha de creación, y terminación (para una versión dada), el identificador del documento asociado, etc. y puede responder a solicitudes de un agente por medio de operaciones de creación, destrucción, modificación del contenido o de sus atributos.

Por otra parte, las clases soportan diferentes mecanismos de relaciones [Booch 94, Rumbaugh et al 91]. Las relaciones pueden darse entre entes de una misma clase, o entre clases distintas. Podemos citar mecanismos de herencia, de agregación, de asociación. Por ejemplo en la fig. 3.3 mostramos un diagrama simplificado de la jerarquía de herencia para la clase "Rol"; un artefacto compuesto está relacionado por un mecanismo de agregación; las clases fundamentales se relacionan entre sí por medio de mecanismos de asociación (y uso, como caso particular de la asociación).

Por ejemplo en la fig. 3.7 una clase de tipo "Agente"[4] asociada o jugando un tipo de "Rol" realiza un tipo de "Tarea". (Por ejemplo: "un agente, Gustavo" controla "la tarea, diseño de contextos de navegación" cuando ocupa el "rol, coordinador del proyecto"; "un agente, Luis" ejecuta "la tarea, diseño de contextos de navegación" cuando ocupa el "rol, desarrollador del proyecto"; "un agente, Marina" participa "en la tarea, diseño de contextos de navegación" cuando ocupa el "rol, usuario participante". El lector puede observar que los agentes tienen distintos derechos, y que se puede establecer un mecanismo de delegación).

En la figura 3.7 consideramos explícitamente las relaciones fundamentales entre las clases. Sin embargo con siete clases bases podríamos tener un número de relaciones potenciales de siete factorial (7!). Por ello una división de preocupaciones por perspectivas puede disminuir la complejidad en el modelado de procesos.

3.3.1 Descripción de las Clases y sus Relaciones Fundamentales

Si bien en la sección 3.1 introducimos un conjunto de conceptos, a seguir profundizaremos

[4] a seguir subrayaremos a las relaciones y pondremos entre comillas a las clases y objetos (en algunos casos acompañada de otra información).

sobre los entes y las relaciones mostradas en la fig. 3.7

Una “tarea” representa una unidad de trabajo que se le asigna a un “agente” para su realización. Una “tarea” contiene “descripciones de proceso”, las que pueden comprender una colección de alternativas -instanciables-, para representar a la misma unidad de trabajo. Asimismo pueden estar especificadas en diferentes formalismos y/o notaciones (fig. 3.5), las que serán comprensibles para los distintos “agentes” involucrados.

Si observamos (leemos) algunas clases y relaciones del diagrama podemos decir que un “agente” realiza una “tarea” contenida en la “descripción de proceso”, usando algún “constructor de proceso” en particular. Para la efectivización de la tarea el “agente” tiene el derecho a invocar las “operaciones” que son parte del “constructor”. Esta clase al ser usada alimenta al “proceso” cuya finalidad es producir un “artefacto” que esté dentro de las “metas” establecidas. El “proceso” se lleva a cabo bajo ciertas “condiciones”.

Además de la “operaciones” que posee una “tarea” asociada al constructor de proceso en particular, la tarea también posee métodos para crear una actividad (constructor), y para la terminación de la misma (destructor). Por otra parte, hay métodos para la delegación de actividades.

Dado el caracter jerárquico de las tareas, en la que una actividad puede depender de otras actividades, delegar una actividad al agente correcto implica que éste tendrá el derecho de invocación de ciertas operaciones. La idea de delegación surge de la relación jerárquica de actividades y en los permisos de los agentes a controlar, ejecutar y/o participar en el desarrollo de las tareas. De este modo podemos distinguir entre el concepto de que una tarea posee operaciones y el concepto de que el agente tiene el derecho a invocarlas. (Observar el ejemplo de Gustavo, Luis y Marina, dado mas arriba, para ejercitar estas ideas).

Un artefacto es un objeto persistente que representa el producto de realizar una tarea. Esto es, un “proceso” produce “artefactos” por medio de la colaboración de uno o más “agentes”, y a su vez un “proceso” reusa “artefactos”.

Un artefacto puede ser un objeto simple o compuesto y puede estar sujeto a un sistema de configuración de cambios. El mismo consiste en una estrategia -asociado a algún método- para identificar, mantener, y administrar los cambios de los artefactos bajo cierta configuración. Se debe considerar aspectos de procedimientos de aceptación y congelamiento de los cambios,

ítems configurables, versionamiento, entre otros asuntos.

Desde el punto de vista de un sistema de configuración y de un modelo de seguimiento, un artefacto es un objeto con propiedades y comportamiento visibles. Un artefacto es un objeto persistente que se caracteriza por su identidad, tipo de artefacto (por ej. un documento es un tipo de artefacto- ver la definición en el glosario), fecha de creación, de ultima actualización, versión, estado (por ej. en estado de procesamiento o en estado de aprobación). Además los agentes, humanos y computarizados, disponen de un conjunto de servicios para realizar sus acciones. Una taxonomía de métodos podría ser la siguiente:

- métodos de contenido, para operar sobre el contenido de un objeto (en general se dispone de herramientas específicas asociadas a constructores de proceso)
- métodos estructurales, para crear nuevos objetos, para agregar a la jerarquía existente, para eliminar artefactos, para consultar la estructuras de relaciones
- métodos para operar sobre el estado
- métodos para dar soporte a un modelo de seguimiento

Considerando una perspectiva de comportamiento, y centrándonos solamente en un modelo de ciclo de vida de un artefacto, podemos utilizar formalismos de modelado tradicionales como, máquinas de estado finita, cartas de estado (statecharts) y sus derivados o redes de Petri. Un posible esquema de transiciones y estados para el ciclo de vida de un ente [Humphrey et al 89], en nuestro caso un artefacto, es el que mostramos en la figura 3.8.

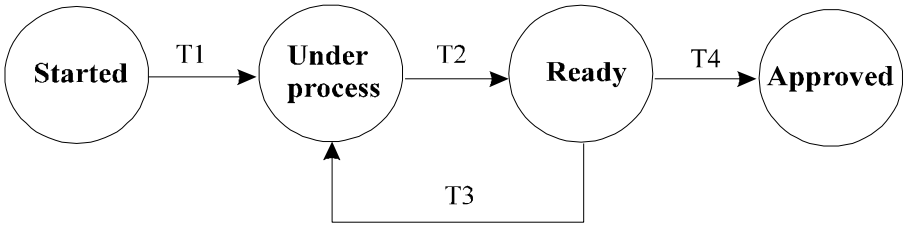

Fig. 3.8 *Un diagrama de transiciones y estados que representa el ciclo de vida de un artefacto.*

Una máquina de estados asociada a un artefacto puede facilitar la definición de la semántica de los estados y transiciones de un producto. Por ejemplo para un artefacto compuesto un objeto de la jerarquía puede pasar a estado “aprobado” si todos sus componentes están aprobados y pasaron a conformar la línea base del sistema de configuración y administración de cambios.

En cuanto a las transiciones son disparadas por eventos. Para que una transición provoque un cambio de estado, se deben cumplir las precondiciones establecidas; el efecto del cambio se puede representar en las postcondiciones. Una forma de especificar lo anterior es mediante la siguiente notación:

Transición $\mathbf{T_n}$: <*Evento, Estado previo, Estado siguiente, Precondición, Postcondición*>

Un ejemplo de condición para pasar del estado "listo" al estado "aprobado" de un objeto compuesto es el indicado más arriba; otra condición es que el agente que dispara la transición, tenga el permiso apropiado (en el cumplimiento de su rol) para provocar los cambios de estado de "listo" a "aprobado".

Finalmente, en cuanto a la clase "agente" abstrae al ente que controla y/o ejecuta y/o participa en la realización de las "actividades" y, además, se relaciona con la clase "artefacto".

La clase "agente" tiene una relación muchos a muchos con la clase "recurso". En la figura 3.2 mostramos un diagrama general de tipos de recursos. Un recurso se puede asociar a varios agentes y, un agente puede estar asociado o usar varios recursos. Por ejemplo una persona, que es un recurso, puede asociarse a varios agentes, y un agente puede usar varios recursos como, una persona, hardware y sistema operativo de una computadora, espacio físico, etc. Un agente puede usar una composición de recursos del mismo tipo (equipo de personas).

3.3.2 Comentarios finales

Esta división de responsabilidades entre la clase "agente" y la clase "recurso" es relevante para los objetivos de la modelización. Se puede pensar en la ejecución de una actividad sin que estén disponibles los recursos. Estos representan a objetos del mundo real, en tanto que los agentes representan a objetos del mundo lógico -que en definitiva implica mayor nivel de abstracción.

La división de preocupaciones entre las clases "agente", "recurso" y "rol" es también de importancia. Principalmente a la separación entre un agente abstracto asociado a un rol para realizar una actividad, de un agente específico, con habilidades particulares. Por ejemplo, la sentencia "El agente X -uno no instanciado-, está asociado al rol de desarrollador del proyecto, cuando realiza el subproceso de diseño navegacional", es de un nivel más abstracto e independiente de la alocación de recursos y ocupación de roles que la siguiente sentencia: "El

agente Gustavo, ocupa el rol de desarrollador del proyecto, dado que posee experticia reconocida en la intervención de los proyectos Menphis y Pampa I, para la realización de la tarea de diseño navegacional, en el proyecto Pampa II". Aquí el ocupante del rol, que es un recurso específico provisto por la organización, posee habilidades específicas, para que integre el nuevo proyecto de hipermedia.

La definición de un modelo conceptual y la distinción entre distintos niveles de abstracción y de preocupaciones son, como indicamos anteriormente, de importancia para la modelización de procesos por varios motivos:
Primero, porque representa a las abstracciones claves del dominio del problema, agrupando comportamientos comunes y haciendo explícitas un conjunto de relaciones. Esto favorece la distribución de responsabilidades e identificación de colaboraciones.

Segundo, favorece el esquema de perspectivas, es decir, la división de preocupaciones en submodelos o vistas del modelo de proceso. Como describimos en la sección 3.1, para disminuir la complejidad, es conveniente separar diferentes tipos de información de los procesos para especificar, comunicar y controlar porciones del modelo. Definimos las perspectivas funcional, informacional, de comportamiento, organizacional y metodológica. Por ejemplo, una perspectiva funcional se focaliza principalmente en la clase proceso y sus relaciones; una vista informacional se focaliza en la clase artefacto, sus componentes y relaciones; una vista organizacional se concentra en las clases recurso, agente y rol y la asignación a las distintas tareas.

Por último, y desde otro punto de vista, es importante la separación de preocupaciones de estos dos enfoques para la modelización: aquel enfoque que no depende de los recursos específicos ni de las diferentes instancias de constructores a usar para una descripción de procesos, de aquel otro enfoque que se concentra esencialmente en los recursos específicos, habilidades particulares de los agentes y ocupación de roles. La figura 3.9 muestra un diagrama representativo del primer enfoque.

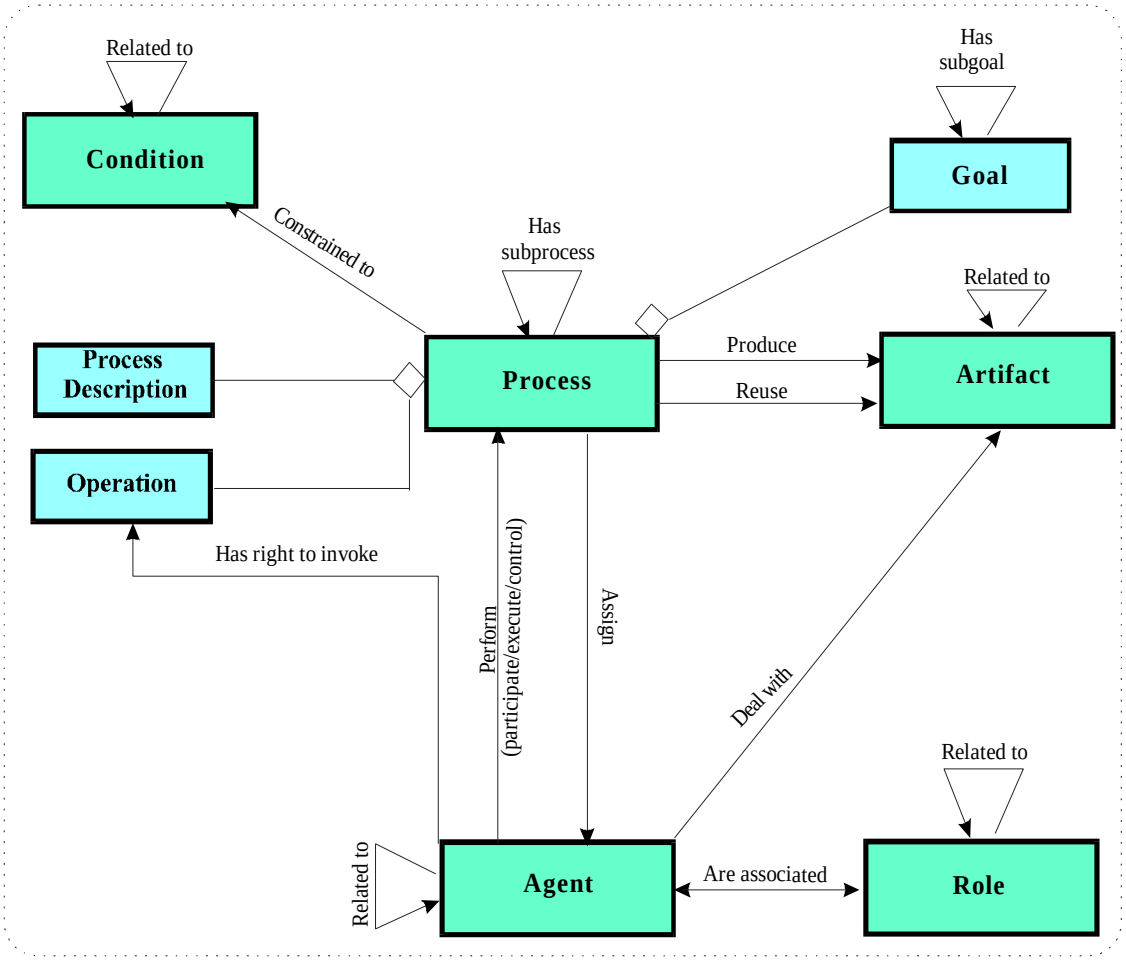

Fig. 3.9 *Diagrama de clases y relaciones fundamentales para un enfoque independiente de los recursos y constructores de proceso.*