Università degli Studi di Torino

Dipartimento di Informatica

Corso di Laurea in Informatica

Relazione di Stage

PROPOSTA DI UN META-MODELLO BASATO SU GSM

E UN ESEMPIO DI APPLICAZIONE

Relatore

Prof. CRISTINA BAROGLIO

Candidato

DARIO FILIPPO

a.a. 2017/2018

12 Aprile 2019

Ringraziamenti

Non so davvero da dove partire. Non riesco nemmeno ancora a credere di essere arrivato a questo traguardo. Ma ci siamo. Sì, al plurale. Perché in questo percorso, per mia fortuna, non sono mai stato solo. Ho avuto la fortuna di incontrare persone speciali che mi hanno sostenuto e aiutato nei periodi più difficili, sia a livello personale che professionale.

I miei ringraziamenti vanno a tutti in egual misura, perciò ho deciso di ordinarli cronologicamente rispetto a quando ho conosciuto le persone che nominerò.

Grazie alla mia famiglia, per avermi reso possibile affrontare questa sfida credendo in me e dandomi il loro supporto. Li ringrazio anche per avermi insegnato importanti valori con ogni piccolo gesto nella vita quotidiana, facendomi diventare quello che sono ora.

A mio fratello, grazie per tutto. Soprattutto per i dolori alle guance per le risate ad ogni mio rientro a casa.

Grazie a Stefania, per il suo sostegno e la sua presenza al mio fianco durante questi anni: sono felice di aver affrontato con lei questo percorso e di condividere con lei una delle tappe fondamentali della mia vita.

A Simone, per essersi dimostrato un amico sincero e leale nonostante per molti motivi non ci siamo potuti vedere molto. Però, ci siamo sostenuti a vicenda, sia durante i momenti di fatica e sconforto, che nei momenti di gioia e soddisfazione al raggiungimento del traguardo. Per questo, grazie amico mio.

Grazie anche a tutte le altre persone che ho avuto la fortuna di conoscere durante questo percorso, perché ognuno di loro ha avuto un impatto su di me: da Ancuta a Zack.

Ringrazio il professor Felice Cardone, per la sua incredibile disponibilità durante molti momenti di questi anni e per avermi dato un punto di riferimento in questo mio cammino.
Ringrazio infine il mio relatore, la professoressa Cristina Baroglio per la sua disponibilità ed i suoi preziosi consigli, sia per quanto riguarda la realizzazione di questa tesi, sia a livello personale.
Un sincero grazie a tutti.

Indice

INDICE

PREFAZIONE

Durante il periodo di stage mi sono concentrato sullo studio di quello che viene definito “Guard-Stage-Milestone Approach”: un sistema sviluppato da IBM per la rappresentazione delle attività di business.
Successivamente, ho realizzato un meta-modello basato su questo approccio.
Nel farlo, ho apportato delle modifiche, al fine di renderlo esprimibile nel linguaggio O-Telos e di estenderne l’utilizzo, per poter implementare un’ampia moltitudine di modelli.
O-Telos è un linguaggio che permette di definire ECA-rules, vere e proprie regole, utili a definire le reazioni del sistema rispetto agli eventi.
La realizzazione del meta-modello è avvenuta in collaborazione con Stefania Bazzano, la quale, nel periodo precedente, ha approfondito il tema della meta-modellazione e il software di riferimento.
In seguito alla realizzazione del meta-modello, me ne sono servito per costruire un modello di una metodologia di sviluppo agile: il Feature Driven Development

Capitolo 1

LA META-MODELLAZIONE

1.1 Introduzione alla meta-modellazione

Prima di poter spiegare che cosa sia un meta-modello e cosa si intenda per meta-modellazione è necessario dare una definizione di modello.

Con modello si intende una rappresentazione astratta di fenomeni del mondo reale. In informatica è uso comune, nelle tecniche di sviluppo software, servirsi di modelli durante le fasi preliminari al fine di creare una sorta di linea guida per lo sviluppo. Esso rappresenta infatti, un'astrazione di ciò che si andrà ad implementare così come, in campo architettonico, il progetto di un edificio rappresenta un'astrazione dell'edificio stesso.

Sapendo ciò, è facile intuire che cosa sia un meta-modello: esso è semplicemente un ulteriore step di astrazione; un'astrazione del modello.

A cosa serve ciò in campo informatico? Dal momento che

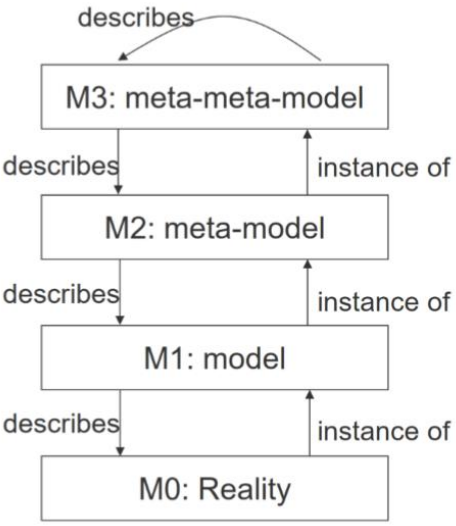


Figura 1.1.1: Gerarchia della meta-modellazione

l'informatica è una delle scienze più utilizzata nel Mondo, è necessario creare degli standard.
Un meta-modello è la rappresentazione della struttura di un modello ovvero ciò che ne evidenzia le proprietà, i vincoli e le regole.
Con meta-modellazione, quindi, si intende quel processo di analisi, costruzione e sviluppo di un (meta-)modello utile a modellare uno specifico contesto. La *figura 1.1.1* riportata a lato descrive la gerarchia della meta-modellazione.

1.2 La meta-modellazione nell'informatica

Come affermato precedentemente, l'uso di modelli è ampiamente diffuso nelle tecniche di sviluppo software. Per l'ingegneria del software infatti, esso è un'alternativa alle più comuni tecniche di sviluppo basate sul codice, tanto valida da arrivare a diventare un ramo di questa, chiamata: *Model-driven Engineering* (MDE) o Ingegneria guidata dal modello.
Uno degli approcci di sviluppo di sistemi software più utilizzati del MDE è denominato *Model-driven architecture* (MDA) proposto da OMG (Object-Management-Group).
Questo approccio si basa sull'utilizzo di un linguaggio per la scrittura di meta-modelli chiamato *Meta Object Facility* o MOF. Tra i meta-modelli proposti da OMG troviamo UML, SysML, SPEM e CWM.
La *Figura 1.1.1* mostra un terzo livello di astrazione non analizzato in precedenza: il meta-meta-modello. Esso non è altro che un modello che descrive la struttura di un meta-modello. Un esempio di meta-meta-modello è quello appena descritto: il MOF.

M3 MOF	Modello che definisce un linguaggio per specificare un meta-modello
M2 UML	Modello che definisce un linguaggio per scrivere modelli
M1 User Models	Un modello definisce un linguaggio utile a descrivere la semantica di un contesto
M0 Instance Models	Contiene le istanze generate run-time degli elementi del modello

Capitolo 2

GUARD-STAGE-MILESTONE APPROACH

2.1 Introduzione

Uno dei sistemi utilizzati per gestire le operazioni di business si basa sulle *business entities with lifecycles* (BEL) anche conosciute come *business artifacts*.
Le BEL sono entità concettuali fondamentali, utili a guidare le operazioni di un lavoro e il cui contenuto muta mentre si avanza in tali operazioni.
Un'entità BEL include un *information model* che acquisisce, in forma concreta o virtuale, tutti i dati rilevanti per l'attività e un modello del ciclo di vita, che specifica i possibili modi in cui l'entità potrebbe progredire durante il lavoro.
Nella maggior parte dei precedenti lavori riguardanti BEL venivano utilizzati modelli dei *cicli di vita* basati su varianti di macchine a stati finiti.
Un team di ricercatori della IBM, con l'aiuto di esperti del settore, ha deciso di definire una struttura che permettesse di gestire la crescente complessità delle attività di business: il "*Guard-Stage-Milestone* Approach" (GSM).
I cicli di vita di GSM sono più dichiarativi delle varianti di macchine a stati finiti e supportano la gerarchia e il parallelismo all'interno di una singola

istanza di entità. La semantica operazionale di GSM si basa su una forma di regole Event-Condition-Action conosciute come *ECA rules*.

2.2 Event-Condition-Action rules

Anche chiamate *Active Rule*, sono un particolare tipo di regole usato nella programmazione basata su eventi e nei database attivi, che specificano come reagire a determinati eventi, sia interni che esterni al sistema.

Il nome ECA ne descrive brevemente la struttura, Event-Condition-Action, sono infatti, le tre parti fondamentali di queste regole:

- Event: specifica quali sono gli eventi che devono innescare l'esecuzione della regola.
- Condition: è una condizione logica che, se soddisfatta, permette l'esecuzione della terza parte della regola.
- Action: specifica le azioni da eseguire.

Basandosi su questa struttura base, sono state create molte varianti, per esempio alcune aggiungono strumenti per una selezione più dettagliata degli eventi e altre prevedono una strutturazione più complessa della terza componente.

2.2.1 ECA-Rule in GSM

Secondo GSM, nell'avanzamento di una entità attraverso il suo ciclo di vita sono individuabili dei *macro-steps*. Un macro-step si compone di *micro-steps* che corrisponde ad un singolo cambiamento degli attributi della BEL. Questi cambiamenti posso avvenire per due motivi: o come effetto diretto di un evento esterno o per effetto di una reazione a catena a partire da esso. Un macro-step si compone di un evento esterno e di tutti i micro-steps causati dall'evento stesso.
Questi micro-steps vengono definiti, in GSM, tramite ECA rules.

2.3 Guard-Stage-Milestone Approach

L'aspetto fondamentale di questo GSM Approach è che esso permette di definire i cicli di vita in una versione dichiarativa, la quale consente di esprimere agevolmente attività di business anche complesse.
Qui di seguito analizzerò i concetti che stanno alla base di GSM.

- **Entità**

  Un'*entità* è definita come un oggetto che possiede un ciclo di vita; essa quindi muta nel tempo in base agli eventi che si verificano.
  Un *ciclo di vita* è costituito dai possibili scenari in cui un'entità può trovarsi e dalle regole che ne determinano il flusso. Un'entità si muove attraverso il suo ciclo di vita per effetto di *eventi esterni* che influenzano gli *status* di *Stage* e *Milestone*.
  All'entità è associato un *Information Model* ovvero un elenco di informazioni rilevanti riguardanti l'entità stessa. Esso solitamente include: i *Data Attribute* (attributi di vario genere che descrivono l'entità) gli *Event Attribute* (attributi che descrivono quali eventi sono accaduti coinvolgendo l'entità e quando sono avvenuti) e lo *status* di ogni *Milestone* ed ogni *Stage* che fanno parte del ciclo di vita dell'entità e quando esso è cambiato.

- **Stage**

  Rappresenta una delle fasi del ciclo di vita dell'entità. Uno *Stage* si compone di una o più *Milestone*, che rappresentano le condizioni da raggiungere per la terminazione dello stesso, una o più *Guard* che indicano le condizioni d'accesso nello Stage, uno *status* ed un corpo.

Lo status indica se lo Stage sia aperto o meno. Esistono solo due tipi di Stage:

1. Uno Stage si dice *Atomico* se il suo corpo possiede uno o più *Task* e non contiene altri Stage.
2. Uno Stage si dice *Non Atomico* se il suo corpo non possiede alcun *Task* ma contiene altri Stage detti *Sotto Stage*.

Un Sotto Stage si può aprire solo se lo Stage Non Atomico che lo contiene è aperto. Ogni Stage deve avere un nome univoco all'interno del ciclo di vita cui appartiene.

Un Sotto Stage, essendo esso stesso uno Stage, può essere Atomico o Non Atomico.

- **Guard**

Una *Guard* rappresenta le *precondizioni* necessarie allo svolgimento dello Stage.

Queste precondizioni vengono espresse tramite *Sentry* (di cui parlerò in seguito).

Una Guard, se verificata, scatena l'apertura dello Stage a cui appartiene, facendogli assumere status *true*. Questo significa che, potendo uno Stage avere più Guard, basterà che solo una di esse si verifichi per rendere aperto lo Stage. Quindi, ad esempio, per far sì che uno Stage si possa aprire solo al verificarsi di due condizioni, è necessario esprimerle con un'unica Sentry.

Se uno Stage possiede due Guard, è necessario scrivere le rispettive Sentry in modo tale che esse non possano in alcun modo verificarsi nello stesso momento.

- **Milestone**

  Ogni *Milestone* ha un nome univoco all'interno del ciclo di vita cui appartiene e rappresenta l'obiettivo che si cerca di raggiungere con lo stage.
  Una Milestone si compone di uno *status*, una o più *Validating Sentry* e zero o più *Invalidating Sentry*. Al verificarsi di una delle Validating Sentries lo status della Milestone assume valore *true*, esprimendo il raggiungimento della Milestone.
  Al verificarsi di una Invalidating Sentry, invece, lo status assume valore negativo, invalidando la Milestone. Lo status di una Milestone non può mutare da false a true se lo Stage a cui appartiene è chiuso. Il raggiungimento di una Milestone fa chiudere lo Stage cui appartiene.
  Se uno Stage possiede due Milestone, è necessario scrivere le rispettive Sentry in modo tale che esse non possano in alcun modo verificarsi nello stesso momento.

- **Sentry**

  Una *Sentry* è una condizione logica che può assumere valore affermativo o negativo a seconda della situazione. Le Sentry hanno la forma: " on E " oppure " on E if Y " dove E rappresenta un evento esterno o uno Status Change Event mentre Y rappresenta una condizione che non coinvolge eventi. La Sentry viene verificata nel momento in cui accade l'evento E e la condizione Y assume valore true.

- **Eventi**

  Gli *Eventi* sono ciò che fa muovere l'entità attraverso il suo ciclo di vita. Vi sono due tipi di eventi: gli *External Event*, che rappresentano

avvenimenti che provengono dall'esterno del modello (input dell'utente, …) , e gli *Status Change Event*, che rappresentano il cambiamento di status di uno Stage o di una Milestone.

- **Task**

I *Task* rappresentano le azioni che si svolgono mentre lo Stage a cui appartengono è aperto. Vi sono cinque tipi di Task:
Si dice di *tipo A*, un Task in cui si assegnano dei valori ad attributi;
Si dice di *tipo B*, un Task che invoca un Servizio Esterno;
Si dice di *tipo C*, un Task che risponde all'invocazione di un Servizio Esterno;
Si dice di *tipo D*, un Task che genera un Evento che coinvolge altre istanze di un'entità;
Si dice di *tipo E*, un Task che permette la creazione di una nuova istanza di un'entità.
Un Atomic Stage può contenere al massimo un Task per ogni tipo, tranne per i Task di Tipo A che possono essere più numerosi.

2.4 GSM-mondo esterno, parallelismo e gestione delle eccezioni

BEL Service Center

È il componente che gestisce le entità e le loro istanze. Offre un'interfaccia per supportare l'interazione con il mondo esterno. Essa permette di effettuare e ricevere chiamate per servizi esterni e l'interazione con operatori umani.

Gestione del parallelismo

Secondo il modello GSM più occorrenze possono essere in esecuzione nello stesso momento, perciò occorre gestire l'accesso agli attributi per evitare che aggiornamenti e letture di essi avvengano contemporaneamente. GSM prevede la creazione di un insieme *ReadSet* ed uno *WriteSet* per ogni Stage, che contengono gli attributi che quello Stage ha necessità di scrivere o leggere. Inoltre, ad ogni attributo deve essere associato uno status, che prende valore *uninitialized* se l'attributo non ha valore, *active* se è aperto uno stage che lo contiene nel WriteSet e *stable* negli altri casi.
Se uno Stage contiene in uno dei propri insiemi un attributo che ha status active, esso non può aprirsi. Ciò è necessario per fare in modo che, se esiste uno Stage aperto che modifica l'attributo x, non possa aprirsi nessun altro Stage che utilizzi quello stesso attributo.

Gestione delle eccezioni

GSM suggerisce una gestione delle eccezioni basata sull'aggiunta di una Milestone di fallimento ad ogni Stage. Inoltre, ad uno Stage non atomico si potrebbe aggiungere un sotto-Stage per la gestione degli errori generati dagli altri sotto-Stage (che si trovano allo stesso livello); in caso contrario, un errore in un sotto-Stage validerà sia la propria Milestone di fallimento che quella dello Stage che lo contiene.

Capitolo 3

STRUMENTI E LINGUAGGI

3.1 Introduzione a ConceptBase e O-Telos

ConceptBase.cc è un sistema multiutente object-oriented basato su un database deduttivo. È un potente strumento per la meta-modellazione e la creazione di linguaggi di modellazione personalizzati. Il sistema dispone di un'interfaccia utente testuale e una grafica.
Questo software è stato sviluppato dal ConceptBase Team presso le Università di Skövde (HIS) e di Aquisgrana (RWTH); è disponibile per Linux, Windows e Mac OS-X.

3.1.1 Livelli di astrazione

Questo strumento permette di rappresentare un numero illimitato di livelli di astrazione, senza tuttavia creare una gerarchia fissa tra essi. La gerarchia tra oggetti è definita solamente tramite il costrutto di istanziazione.
Tutti gli oggetti vengono rappresentati tramite la stessa struttura, una quadrupla detta P-Fact (di cui parlerò più avanti). Gli oggetti, i loro attributi, le relazioni di specializzazioni e istanziazioni tra essi, sono tutti rappresentati come P-Fact. Questo permette, tra le altre cose, di supportare l'istanziazione tra attributi.

3.1.2 Regole deduttive, vincoli d'integrità e query

Questo sistema permette di definire regole deduttive, vincoli d'integrità e query sotto forma di formule in logica del prim'ordine. Internamente, il sistema, le trasforma di clausole di Horn (formule nelle quali i letterali sono in disgiunzione tra loro e al più uno può essere positivo) interpretate da una macchina di valutazione basata su Datalog. Le espressioni logiche in ConceptBase.cc possono operare sugli oggetti indipendentemente dal loro tipo e dal suo livello di astrazione. Datalog è noto per essere il più robusto sistema computazionale per la valutazione delle espressioni logiche.

3.1.3 Active Rules

In ConceptBase è possibile definire delle regole chiamate: Active Rules. Esse si basano sulla nozione di ECA rule, ovvero regole che reagiscono al verificarsi di un evento. Esse possono modificare il database tramite le azioni di TELL, UNTELL e RETELL oppure invocare procedure esterne. Inoltre, queste ultime possono essere aggiunte in modo incrementale al database utilizzando una semplice interfaccia di programmazione basata su Prolog.

3.1.4 Funzioni

ConceptBase.cc supporta funzioni ed espressioni aritmetiche per definire il calcolo all'interno dei modelli. Esse possono essere definite ricorsivamente in modo simile a come avviene nella programmazione funzionale. Le funzioni sono particolarmente utili per definire metriche complesse sui modelli. Alcune di esse, come SUM e AVG, sono predefinite.

3.2 O-Telos

O-Telos è il linguaggio per la modellazione di dati che sta alla base del sistema ConceptBase. Si fonda su un singolo costrutto, il P-Fact, mediante il quale possono essere definiti concetti appartenenti a qualsiasi livello di astrazione (istanza, classe, meta-classe, ecc.). Regole deduttive e vincoli di integrità estendono il modello computazionale a quello di un database deduttivo.
O-Telos dovrebbe essere considerato un linguaggio di modellazione dei dati di un meta-database. È capace di rappresentare caratteristiche semantiche di altri linguaggi di modellazione come diagrammi Entity-Relationship e diagrammi di flusso.

3.2.1 Origini e basi del linguaggio

Il diretto predecessore di O-Telos è il linguaggio per la rappresentazione della conoscenza Telos e del quale è una versione object-oriented. Ideato da John Mylopoulos, Alex Borgida, Matthias Jarke e Manolis Koubarakis presso l'Università di Toronto, Telos è stato progettato per rappresentare informazioni riguardanti sistemi informatici. Esso è stato ottenuto basandosi su RML (linguaggio per la modellazione di requisiti sviluppato per una dissertazione di laurea da Greenspan) e su una sua evoluzione, il CML (Conceptual Modeling Language) (descritto e formalizzato da Stanley nella seconda metà degli anni '80). Quest'ultimo adotta una complessa struttura dei dati per rappresentare la conoscenza permettendo l'espressione di complessi concetti temporali e la definizione di meta-concetti. Telos ne è una versione "ripulita", sia dalla prospettiva della definizione che dell'implementazione del linguaggio.
Durante l'implementazione di Telos divenne evidente che la semantica originale era troppo complessa per una implementazione efficiente. La componente temporale di Telos comprendeva sia un *valid time* (quando un'informazione è vera nel dominio) che un *transaction time* (quando si

considera l'informazione parte del database). Questa caratteristica produceva effetti indesiderati nella valutazione delle query e nella rappresentazione uniforme delle informazioni come oggetti; per questo motivo, fu eliminato il *valid time* mantenendo tuttavia il *transaction time.* Alcune altre caratteristiche come le operazioni TELL, UNTELL e RETELL sono implementate solo in un modo limitato, essenzialmente proibendo modifiche dirette a fatti derivati. D'altro canto, O-Telos estende la rappresentazione ad oggetti a qualsiasi informazione esplicita e riduce il numero di oggetti essenziali a soli cinque.

3.2.2 O-Telos in ConceptBase

L'implementazione di O-Telos presente in ConceptBase ha alcune caratteristiche specifiche. Innanzi tutto, è presente un linguaggio per le query dedicato, CBL, che fornisce un'interpretazione basata sugli oggetti per le query. In secondo luogo, sono stati introdotti i moduli gestire le funzionalitá multi-utente offerte da ConceptBase; essenzialmente, l'identificativo del modulo è aggiunto all'identificatore dell'oggetto. In terzo luogo, ConceptBase supporta una versione limitata delle *Active Rule*. Infine, ConceptBase supporta funzioni ed espressioni aritmetiche definite ricorsivamente.

3.2.3 P-Fact

Un database in O-Telos è una collezione di P-Fact, ognuno dei quali rappresenta un oggetto. Il P-Fact si esprime come P (*oid*, *x*, *n*, *y*, *tt*) dove *oid* è l'identificatore del P-Fact, *x* è la sorgente, *n* è l'etichetta, *y* è la destinazione e *tt* è l'intervallo (transaction time) in cui l'oggetto è considerato valido nel database (per ulteriori chiarimenti vedere la sezione 3.2.4). O-Telos identifica 5 oggetti predefiniti:

- *Proposition*

  Contiene tutte le proposizioni come istanze. Una proposizione è qualsiasi P-fact che ha la forma P (oid, x, n, y, tt). Ogni proposizione deve rientrare esattamente in uno dei casi successivi.

- *Individual*

  È la classe di tutti i P-Fact che hanno la forma P (oid, oid, n, oid, tt). Tali P-Fact sono indicati come nodi nella rappresentazione grafica di un database O-Telos.

- *InstanceOf*

  Contiene tutte le ralazioni di istanziazione esplicite sotto forma di istanze. Questi P-Fact hanno la forma P (*oid, x, * instanceof, c, tt*), ovvero *x* è un'istanza (esplicita) di *c*. Nella rappresentazione grafica, una relazione di istanziazione è un collegamento tra un oggetto *x* e la sua classe *c*.

- *IsA*

  Contiene tutte le relazioni di specializzazione esplicite come istanze. Questi P-fact hanno la forma P (oid, c, * isa, d, tt) Una relazione di specializzazione viene visualizzata graficamente come collegamento tra una sottoclasse c e la sua superclasse d.

- *Attribute*

  contiene tutte relazioni di attribuzione esplicite come istanze. Questi P-Fact hanno la forma P (*oid, x, m, y, tt*) dove *m* deve essere diverso da **instanceof* e **isa*. Viene visualizzato da un collegamento tra l'oggetto di origine *x* e l'oggetto di destinazione *y* su cui è scritta l'etichetta. L'oggetto *y* è anche chiamato valore dell'attributo. In O-Telos, l'attributo assume le caratteristiche di una relazione tra oggetti poiché i valori sono essi stessi oggetti in O-Telos.

ConceptBase supporta la derivazione di relazioni di istanziazione e di attribuzione tra oggetti tramite regole deduttive. Queste relazioni derivate non hanno le proprietà di un oggetto, cioè non sono identificate e rappresentate come una proposizione. Nello specifico, l'istanziazione tra gli oggetti definiti dall'utente e i cinque predefiniti è derivata tramite regole deduttive.

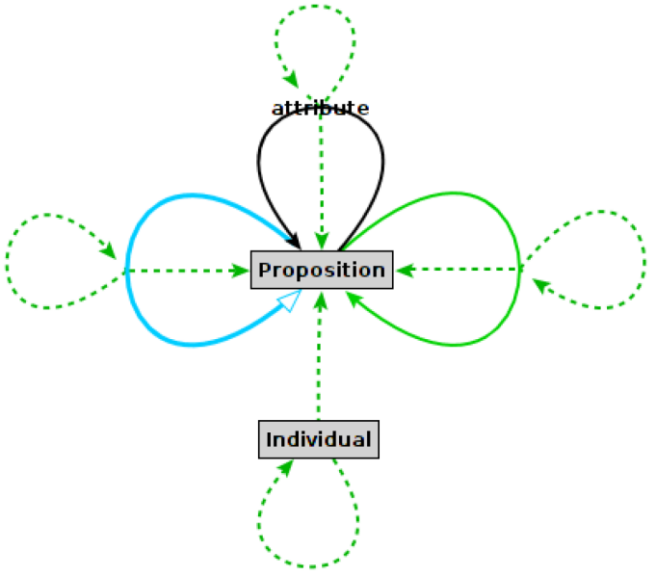


Figura 3.2.1: Le frecce di colore verde rappresentano relazioni di istanziazione, quelle blu di specializzazione e quelle nere di attribuzione. Se una linea é tratteggiata significa che la relazone che rappresenta é frutto di una deduzione.

Oltre a quelli sopra nominati, ConceptBase fornisce altri quattro oggetti predefiniti:

- *Class*

  Contiene tutte le cosiddette classi (inclusa sé stessa) come istanze. L'unica proprietà speciale è la definizione di due attributi rule e constraint, che quindi permettono le sue istanze di avere regole e vincoli.

- *Integer, Real* e *String*

  Queste tre classi sono supportate dai predicati di confronto (come x<y) e dalle funzioni come PLUS, MINUS, ecc.

L'utente non utilizza direttamente i P-fact ma una versione testuale o grafica del database. Entrambe queste notazioni non si basano sugli *oid* degli oggetti ma sulle loro etichette. Per questo motivo è necessario seguire alcune convenzioni, quali l'unicità delle etichette degli oggetti, escluse quelle degli attributi che possono ripetersi se hanno sorgente diversa, e la presenza di, al massimo, una relazione di istanziazione o di specializzazione tra 2 specifici oggetti.

```
1. Stage in Class with
2.   attribute, necessary, single
3.     name : String
4.   attribute, necessary
5.     milestones : Milestone
6.   constraint
7.     parentStageConstraint:
8.       $ forall s/Stage
9.           forall ns1,ns2/NotAtomicStage
10.            (ns1 body s) and (ns2 body s) ==> (ns1 = ns2)
   $
11. end
```

3.2.4 Database e Transaction Time

In O-Telos il database che contiene tutti i P-Fact è detto Database Storico ed è espresso come:

OB = {P(oid, x, n, y, tt) | oid, x, y, tt ∈ ID, n ∈ LABEL}

Il tempo di transazione *tt* è rappresentato da due punti temporali *tt (a,b)* dove *a* rappresenta il momento in cui è stata eseguita la TELL del P-Fact e *b* la UNTELL.

TELL e UNTELL, sono le principali operazioni presenti in O-Telos: TELL serve per inserire i P-Fact nel Database Corrente, UNTELL, semplicemente, lo elimina. Come si evince dalla formula, all'interno del

Database Storico, sono presenti sia i P-Fact attualmente presenti nel Database Corrente, sia tutti quelli che ci sono stati in passato. Perciò, il Database Corrente, non è altro che quella porzione del Database Storico che contiene i P-Fact di cui non è ancora stato eseguito UNTELL. Il *tt* di questi sarà composto dal momento nel quale è stata eseguita la TELL (*a*) e *infinity* (che sta a specificare che la UNTELL non è ancora avvenuta).
Intuitivamente, quando viene eseguita l'operazione UNTELL su di un P-Fact, il sistema ne modifica la fine del *tt*, che prende come valore il momento attuale.

3.2.5 Regole deduttive e vincoli d'integrità

O-Telos permette di definire regole deduttive utili a derivare attributi e vincoli d'integrità utili a definire condizioni inviolabili. Queste devono essere inserite come attributi all'interno di oggetti che siano istanze di Class. Nel farlo, è necessario definire le regole deduttive come appartenenti alla categoria *rule* mentre, i vincoli d'integrità, alla categoria *constraint*. Il linguaggio utilizzato per esprimere queste "regole" è detto CBL e si tratta del linguaggio predicativo di ConceptBase.
Le variabili all'interno delle formule devono essere assegnate a un "tipo", che rappresenta l'oggetto di cui devono essere istanze, e quantificate con l'operatore forall (quantificatore universale) o exists (quantificatore esistenziale).
Nelle formule è inoltre possibile utilizzare diversi predicati per indicare le possibili relazioni presenti tra gli oggetti.

```
1. ExternalEvent in Class isA Event with
2.   attribute, necessary, single
3.     name : String
4.   attribute, necessary
5.     entity : Entity
6.   rule
7.     entityIsNecessary :
8.       $forall a/ExternalEvent!entity
9.          In(a,Proposition!necessary)$
10.end
```

```
Date in Domain, Class with
  fields
    year: Integer;
    month: Integer;
    day: Integer
  constraint
    yearConstraint :
      $forall d/Date i/Integer
         (d year i) ==> (i > 2017)$;
  monthConstraint :
    $forall d/Date i/Integer
       (d month i) ==> (i > 0) and (i < 13)$;
  dayConstraint :
    $forall d/Date i/Integer
       (d day i) ==> (i > 0) and (i < 32)$;
end
```

3.2.6 Query

ConceptBase realizza le query sotto forma di classi, le cui istanze soddisfano il vincolo di appartenenza della query. Le query sono istanze di una classe di sistema chiamata *QueryClass*, appunto, che definisce la struttura delle query come di seguito:

```
QueryClass in Class isA Class with
attribute
    retrieved_attribute: Proposition;
    computed_attribute: Proposition
  single
    constraint: MSFOLquery
end
```

Esempio di query:

```
ProgettiConFunzionalità in QueryClass isA Progetto with
  constraint
    c1: $ exists f/Funzionalità (this funzionalità f) $
end
```

Ogni query, come già detto, deve essere istanza di *QueryClass* e inoltre, deve essere una specializzazione di una o più classi, di cui sarà necessario essere istanze per soddisfare la query; queste classi possono a loro volta

essere query. Il vincolo *constraint* esprime le condizioni che devono essere rispettate per soddisfare la query; può essere presente un solo vincolo, l'attributo constraint appartiene, infatti, alla categoria *single*. I vincoli all'interno delle query usano lo stesso linguaggio di quelli presenti negli altri oggetti, con in piú la possibilitá di fare riferimento all'oggetto in questione, il *this*.

La funzione di *retrieved_attribute* è simile a quella della proiezione in algebra relazionale; la query può elencare, in questa categoria, alcuni attributi delle classi di cui è specializzazione, in modo da potervi far riferimento tramite la query; tali attributi devo appartenere alla categoria *necessary*, ovvero avere cardinalità minima 1.

Nella categoria *computed_attribute* è possibile inserire attributi che si vuole assegnare alla query ma che non appartengono alle classi che essa specializza. Il valore di tali attributi viene assegnato all'interno del vincolo.

3.3 ECArule in O-Telos

In O-Telos le ECArule vengono definite come istanze della classe ECArule definita qui di seguito:

```
1. Class ECArule with
2.   attribute
3.     ecarule : ECAassertion;
4.     priority_after : ECArule;
5.     priority_before : ECArule;
6.     mode : ECAmode;
7.     active : Boolean;
8.     depth : Integer;
9.     rejectMsg : String
10.constraint
11.  [...]
12.end
```

La regola vera e propria è l'istanza di ECAassertion. Qui di seguito un esempio di ECArule con all'interno l'ECAassertion:

```
1. CloseStageRule in ECArule with
2.   mode m : Deferred
3.   ecarule
4.   myecarule:
5.     $s/StageType m/MilestoneType
6.      e/MilestoneAchievedEvent d/Individual
7.     ON Tell In(e, MilestoneAchievedEvent)
8.     IFNEW (e in MlestoneAchievedEvent)
9.           and (m in MilestoneType) and (e milestone m)
10.        and (s in StageType) and (s [milestones] m)
11.        and (s status TRUE)
12.  DO Retell A(s, status, FALSE),
13.     CALL CreateIndividual (StClEvent, d),
14.     Tell (d in StageClosedEvent),
15.     Tell (d stage/st s)
16.  ELSE reject
17.  $
18.end
```

Una ECAassertion si compone di quattro parti:

1) Dichiarazione delle variabili; avviene nello stesso modo in cui avviene nelle regole deduttive e nei vincoli d'integrità. Tuttavia, i tipi assegnati alle variabili in questa fase non vengono presi in considerazione nelle successive parti della regola, necessitando quindi di essere "ribaditi" tramite il predicato In.
2) ON-part; corrisponde alla parte Event delle active rules. Gli eventi rilevabili in O-Telos sono tre: Tell, Untell e Ask (comando utilizzato per eseguire una query). Questi eventi possono essere generati dall'utente o da altre ECArule; le modifiche al database apportate da regole deduttive non vengono considerate eventi. È possibile specificare un solo evento.
3) IF-part; corrisponde alla parte Condition delle active rules. All'inizio di questa parte è possibile usare la parola IF o IFNEW. Con IF la condizione verrà valutata considerando lo stato del database prima dell'inizio della transazione che ha scatenato l'ECArule. Con IFNEW verranno prese in considerazione le modifiche apportate dalla transazione fino al momento dell'esecuzione della regola. Nella condizione possono essere usati gli stessi predicati validi nelle regole

deduttive, ma non sono consentiti i quantificatori. Se si ha necessità di usarli occorre creare e sfruttare una query.

4) DO-part; rappresenta la parte Action delle active rules. Viene eseguita su tutti gli oggetti che corrispondono alla condizione espressa nella IF-part. In essa è possibile specificare più azioni da eseguire. Le azioni possibili sono:
   - Tell: aggiunge oggetti al database
   - Untell: elimina oggetti dal database
   - Retell: modifica oggetti (esegue prima un Untell e poi un Tell)
   - Call: invoca procedure esterne in Prolog.
   - Ask: esegue una query
   - Raise: lancia una query ma senza valutarla, è utile a far reagire altre regole che sono scatenate da un evento ASK
   - Noop: non compie nessuna operazione
   - Reject: annulla tutti i cambiamenti apportati al database dalla transazione che ha innescato la regola.

   La DO-part può essere seguita da un ELSE, questo dà la possibilità di specificare una serie di azioni da compiere nel caso si verifichi l'evento (espresso nella ON-part) e la condizione (espressa nella IF-part) non fosse verificata.

Oltre all'ECAassertion, un ECArule in O-Telos presenta altri parametri:

- *prioriry_after* e *priority_before* servono a definire un ordine nell'esecuzione delle ECArule, nel caso più di una venga innescata nello stesso istante.
- *mode* serve a definire quando l'ECArule vada eseguita, rispetto alla transazione. Può assumere tre valori: *Immediate* fa sì che la regola venga eseguita non appena viene innescata; *ImmediateDeferred* fa sì che la IF-part vengo eseguita non appena la regola viene innescata, mentre la DO-part verrà eseguita a transazione terminata; *Deferred* fa sì che sia la IF-part che la DO-part vengano eseguite dopo la fine della transazione.

- *active* permette di disattivare temporaneamente la regola senza eliminarla dal database.
- *depth* viene utilizzato se si vuole porre un limite al numero di ECArule innescate a catena. Utile ad evitare loop.
- *rejectMsg* definisce il messaggio da mostrare nel caso in cui l'ECArule porti all'annullamento della transazione.

Tutti questi attributi sono opzionali.

Capitolo 4

IMPLEMENTAZIONE DEL METAMODELLO GSM

4.1 Introduzione

In questo capitolo descriverò gli elementi che compongono il meta-modello, basato sull'approccio Guard-Stage-Milestone, che abbiamo progettato e realizzato durante il tirocinio. Durante la progettazione, abbiamo ritenuto opportuno introdurre alcune variazioni, rispetto all'approccio GSM, per rendere più efficiente e accessibile il meta-modello. Tali scelte verranno spiegate e motivate nel corso del capitolo.

4.2 Realizzazione

Il meta-modello è formato da cinque oggetti principali:

1) *Entity*

```
1. Entity in class with
2. attribute, necessary, single
3.     name: String
4.   attribute, necessary
5.     lifecycle: Stage
6.   attribute
7.     dataAttributes: Domain
8. end
```

Questo oggetto rappresenta l'entità GSM. Questo, possiede 3 attributi.

Il primo dei tre, *dataAttributes*, è quello di cui sono istanza gli attributi descrittivi dell'entità.

Un altro attributo è *lifeCycle*. Abbiamo deciso di includerlo nell'oggetto *Entity,* per rappresentare il concetto, espresso in GSM, secondo cui ogni entità possiede un ciclo di vita. Esso è utile a contenere come istanze gli attributi che indicano gli stage di cui l'entità è composta.

Il terzo e ultimo è *name*, del quale deve essere presente una sola istanza per ogni entità. Abbiamo deciso di aggiungere questo attributo, non previsto da GSM, per dare la possibilità di assegnare un nome adeguato ad ogni entità del modello.

Secondo l'approccio GSM l'entità è caratterizzata da un Information Model. Questo comprende degli attributi che descrivono l'entità, le informazioni riguardanti gli eventi in cui è stata coinvolta e lo status delle Milestone e degli Stage facenti parte del suo ciclo di vita.

Durante l'implementazione del meta-modello abbiamo constatato che sarebbe stato più efficace distribuire gli attributi che compongono l'Information Model nel seguente modo: gli attributi descrittivi come attributi di Entity (rappresentati da *dataAttributes)*; gli attributi riguardanti gli eventi come *dataAttributes* di *Event* e lo status di Milestone e Stage come *status* di *MilestoneType* e *StageType,* di cui parleremo più avanti. Questa modifica rende più semplice la scrittura delle condizioni logiche contenute nelle Guard e nelle Milestone, senza introdurre ridondanze.

2) Stage

```
1. Stage in Class with
2.   attribute, necessary, single
3.     name: String
4.   attribute, necessary
5.     milestones: Milestone;
6.     guards: ECArule
7.   constraint
8.     parentStageConstraint:
```

```
9.       $ forall s/Stage
10.           forall ns1,ns2/NotAtomicStage
11.                 (ns1 body s) and
12.                 (ns2 body s) ==> (ns1 = ns2) $
13. end
```

Questo oggetto rappresenta una fase del ciclo di vita dell'entità. Possiede un attributo *name*, avente le stesse caratteristiche dell'attributo omonimo in *Entity*. L'attributo *milestones* è stato inserito per rappresentare il concetto, espresso in GSM, secondo cui ogni stage è caratterizzato da una o più milestone. Esso è quindi istanziato almeno una volta in ogni stage, indicando le milestone che vi appartengono.

L'approccio GSM prevede, per ogni stage, la presenza di una o più guard. Ognuna di esse contiene, a sua volta, una sentry, ovvero una condizione logica, che deve essere verificata per aprire lo stage. Nel meta-modello, da noi creato, abbiamo deciso di non implementare una classe che rappresenti le guard. Esse sono dichiarate come istanze della classe *ECArule*, di cui ho parlato nel capitolo riguardante O-Telos. Abbiamo ritenuto migliore questa soluzione in quanto la classe guard non avrebbe avuto altri attributi oltre all'*ECArule* stessa.

Un'altra differenza rispetto all'approccio GSM è la posizione dell'attributo *status*, che simboleggia l'apertura o chiusura dello stage. Questo non è contenuto nella classe *Stage,* come vorrebbe GSM, bensì in *StageType*.

```
1. StageType in Class with
2.   attribute, single
3.     status : Boolean;
4.     parentEntity : EntityType
5. end
```

Secondo il nostro meta-modello, ogni istanza di *Stage* del livello M0 è istanza di *StageType* (questa relazione viene ottenuta tramite l'attivazione di *ECArule*) e ne istanzia (tramite *ECArule*) i due attributi, *status* e *parentEntity*. Il primo rappresenta, appunto, lo stato di apertura dello stage; mentre il secondo indica l'entità a cui lo stage appartiene. Il posizionamento dell'attributo *status* in *StageType* si è reso necessario in

quanto questo attributo non può, e non deve, prendere valore nel livello M1.

Mentre per l'attributo *parentEntity,* non previsto da GSM, è stato aggiunto per rendere più agevole la scrittura delle condizioni logiche delle milestone e delle guard.

Nel meta-modello sono presenti due classi che specializzano *Stage*:

- *AtomicStage,* che possiede un attributo *tasks.* Ogni istanza di questo oggetto deve avere uno o più attributi in questa categoria; essi conterranno i task da svolgere all'apertura dello stage.

```
1. AtomicStage in Class isA Stage with
2.   attribute, necessary
3.     tasks: Task
4.   constraint
5.     taskTypeBConst:
6.       $ forall s/AtomicStage
7.           forall t,t2/Task (t type TaskTypeB) and
8.                  (t2 type TaskTypeB) and
9.                  (s tasks t) and
10.                (s tasks t2) ==> (t = t2) $;
11.   taskTypeCConst:
12.     $ forall s/AtomicStage
13.         forall t,t2/Task (t type TaskTypeC) and
14.                (t2 type TaskTypeC) and
15.                (s tasks t) and
16.                (s tasks t2) ==> (t = t2) $;
17.   taskTypeDConst:
18.     $ forall s/AtomicStage
19.         forall t,t2/Task (t type TaskTypeD) and
20.                (t2 type TaskTypeD) and
21.                (s tasks t) and
22.                (s tasks t2) ==> (t = t2) $;
23.   taskTypeEConst:
24.     $ forall s/AtomicStage
25.         forall t,t2/Task (t type TaskTypeE) and
26.                (t2 type TaskTypeE) and
27.                (s tasks t) and
28.                (s tasks t2) ==> (t = t2) $
29. end
```

NotAtomicStage, che possiede un attributo *body.* Anche in questo caso, ogni istanza di questo oggetto deve avere uno o più attributi in questa categoria; essi conterranno altri stage, detti sotto-stage.

Come illustrato in GSM, i sotto-stage rappresentano delle fasi interne allo stage, ognuna delle quali ha le stesse caratteristiche di uno stage (milestones, guards, etc.) e due aggiuntive. La prima consiste nell'impossibilità, da parte del sotto-stage, di aprirsi se lo stage che lo contiene è chiuso. La seconda consta invece, nella sua chiusura se si chiude lo stage in cui è contenuto. Nel nostro meta-modello abbiamo implementato tali caratteristiche tramite *ECArule*.

```
1. NotAtomicStage in Class isA Stage with
2.   attribute, necessary
3.     body: Stage
4. end
```

Per il corretto funzionamento di un modello è necessario che, in esso, ogni stage sia istanza di uno tra *AtomicStage* e *NotAtomicStage*; tuttavia nel meta-modello non è stato possibile, a causa dei limiti espressivi del linguaggio, inserire regole che vincolino questo aspetto.

3) Milestone

```
1. Milestone in Class with
2.   attribute, necessary, single
3.     name: String
4.   attribute, necessary
5.     validatingSentries: ECArule
6.   attribute
7.     invalidatingSentries: ECArule
8. end
```

Questo oggetto rappresenta un obbiettivo da raggiungere tramite lo stage. Possiede un attributo *name*, avente le stesse caratteristiche dell'attributo omonimo in *Entity*.

L'attributo *validatingSentries* indica le *ECArule* che portano al raggiungimento della milestone, mentre *invalidatingSentries* indica, se esistono, quelle che ne causano l'invalidamento. Nell'approccio GSM, queste condizioni e quelle che portano all'apertura degli stage vengono chiamate Sentry; esse hanno forma analoga alle ECArule e per questo non

abbiamo ritenuto necessario, né opportuno, creare una classe per rappresentarle, bensì le abbiamo espresse tramite la classe predefinita ECArule.

Un'altra differenza rispetto all'approccio GSM è la posizione dell'attributo *status*, che simboleggia il raggiungimento o meno della milestone. Questo non è contenuto nella classe *Milestone,* come vorrebbe GSM, bensì in *MilestoneType*.

```
1. MilestoneType in Class with
2.   attribute, single
3.     status: Boolean
4. end
```

Secondo il nostro meta-modello, ogni istanza di *Milestone* del livello M0 è istanza di *MilestoneType* (questa relazione viene ottenuta tramite l'attivazione di *ECArule*) e ne istanzia (tramite *ECArule*) l'attributo *status*. Il posizionamento dell'attributo *status* in *MilestoneType* si è reso necessario in quanto questo attributo non può, e non deve, prendere valore nel livello M1.

4) Task

```
1. Task in Class with
2.   attribute, necessary, single
3.     type: TaskType;
4.     description: String
5. end
```

È l'oggetto che rappresenta un'operazione da svolgere all'interno di uno stage. Possiede un attributo *description*, non previsto da GSM, la cui istanza contiene una descrizione dello specifico task. Esso è stato inserito per una migliore usabilità del meta-modello.

È stato, inoltre, inserito un altro attributo, *type*, che deve essere obbligatoriamente istanziato in ogni task e ne deve definire il tipo. Secondo GSM, infatti, sono individuabili cinque tipi di task:

```
1. TaskType in Class end
2.
```

```
3. TaskTypeA in TaskType end
4. TaskTypeB in TaskType end
5. TaskTypeC in TaskType end
6. TaskTypeD in TaskType end
7. TaskTypeE in TaskType end
```

A. task in cui vengono assegnati valori ad attributi dell'entità
B. task in cui viene invocato un servizio esterno
C. task in cui si risponde ad un servizio esterno
D. task in cui si genera un evento che coinvolge altre entità
E. task in cui viene creata una nuova istanza di un'entità

Nel meta-modello sono stati creati degli appositi oggetti per simboleggiare questi tipi (*TaskTypeA*, *TaskTypeB*, etc.). GSM pone un ulteriore vincolo secondo cui, in uno stage, può essere presente al massimo un task per ogni tipo; ad eccezione del tipo A che non presenta limitazioni. Nel nostro meta-modello, abbiamo scelto di esprimere tale vincolo tramite alcune *constraint* all'interno di *AtomicStage*.

5) Event

```
1. Event in Class with
2.   attribute
3.     dataAttributes : Domain
4. end
```

È l'oggetto che rappresenta gli eventi che accadono all'entità provocandone l'avanzamento nel ciclo di vita. Nonostante GSM non lo preveda, abbiamo ritenuto opportuno inserire in questa classe un attributo, *dataAttributes*, con significato e utilizzo analoghi all'attributo omonimo di *Entity*.
GSM distingue due categorie di eventi:

- Eventi esterni, ovvero prevenienti dal mondo esterno al sistema.
- Eventi interni, eventi generati all'interno del sistema, ovvero i cambiamenti di stato degli elementi di quest'ultimo.

Nel nostro meta-modello abbiamo rappresentato queste categorie come due classi, *ExternalEvent* e *StatusChangeEvent*, che specializzano la classe *Event*.

In un modello, le istanze della classe ExternalEvent rappresentano i tipi di eventi esterni che esistono in quel contesto.

```
1. ExternalEvent in Class isA Event with
2.   attribute, necessary, single
3.     name : String
4.   attribute, necessary
5.     entity : Entity
6.   rule
7.     entityIsNecessary :
8.       $forall a/ExternalEvent!entity
9.         In(a,Proposition!necessary)$
10. end
```

La classe ExternalEvent possiede due attributi, *name* e *entity*. Il primo è analogo all' omonimo attributo presenti in *Entity*. Per quanto riguarda il secondo, ogni istanza di *ExternalEvent* deve avere almeno un'istanza di quest'attributo; in ognuna di esse viene indicata un'entità che viene coinvolta da quel tipo di evento.

```
1. StatusChangeEvent in Class isA Event end
```

Le istanze di *StatusChangeEvent* sono, invece, già definite nel meta-modello:

- *MilestoneAchivedEvent*

```
1. MilestoneAchivedEvent in StatusChangeEvent with
2.   attribute, single, necessary
3.     milestone: MilestoneType
4. end
```

Questo oggetto viene istanziato quando lo status di una milestone passa da FALSE a TRUE.

- *MilestoneInvalidatedEvent*

```
1. MilestoneInvalidatedEvent in StatusChangeEvent with
2.   attribute, single, necessary
```

```
3.     milestone: MilestoneType
4. end
```

Questo oggetto viene istanziato quando lo status di una milestone passa da TRUE a FALSE.

- *StageOpenedEvent*

```
1. StageOpenedEvent in StatusChangeEvent with
2.   attribute, single, necessary
3.     stage: StageType
4. end
```

Questo oggetto viene istanziato quando lo status di uno stage passa da TRUE a FALSE.

- *StageClosedEvent*

```
1. StageClosedEvent in StatusChangeEvent with
2.   attribute, single, necessary
3.     stage: StageType
4. end
```

Questo oggetto viene istanziato quando lo status di uno stage passa da FALSE a TRUE.

Ognuna di esse ha un solo attributo, nelle prime due ha etichetta *milestone*, nelle altre due *stage*. Nonostante i diversi nomi, questi attributi hanno tutti la stessa funzione: indicare quale oggetto ha cambiato stato.

I due schemi seguenti, che abbiamo creato utilizzando il tool grafico di ConceptBase, mostrano la struttura del meta-modello del GSM Approach. ConceptBase permette di derivare questo tipo di grafi direttamente dall'implementazione dei modelli, perciò è possibile notare al loro interno ognuno degli elementi descritti in precedenza, riportato con lo stesso nome con cui compare nel codice O-Telos che costituisce l'implementazione del meta-modello.

Il meta-modello è qui diviso in due parti: la prima comprende gli elementi le cui istanze descrivono la struttura del ciclo di vita di un'entità, la seconda comprende invece tutti quegli elementi utili a descrivere gli eventi che fanno procedere un'entità attraverso il suo ciclo di vita.

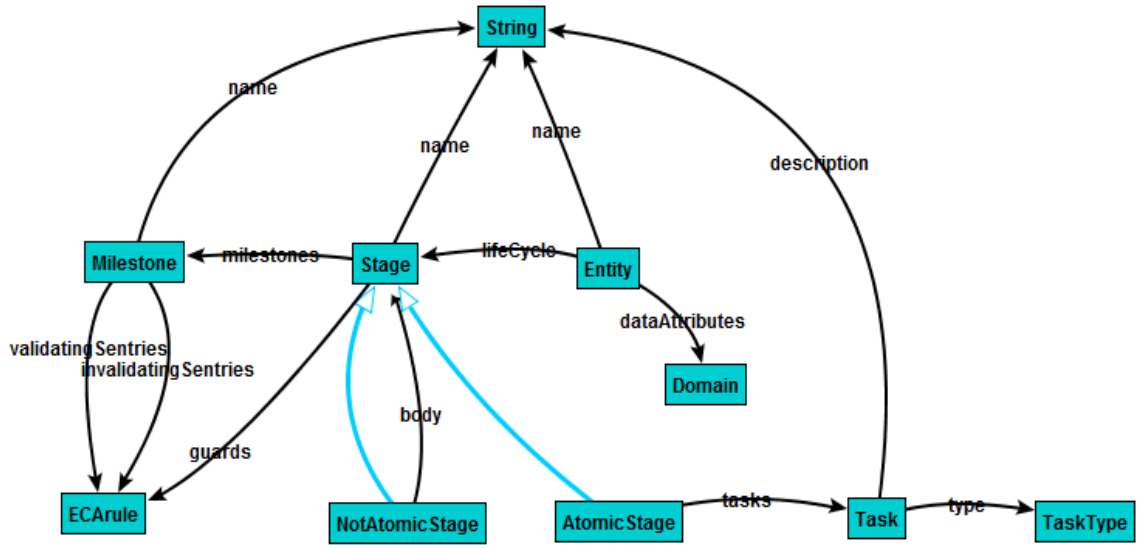


Figura 4.2.1: Parte del meta-modello contenente gli elementi che permettono di definire il "ciclo di vita"

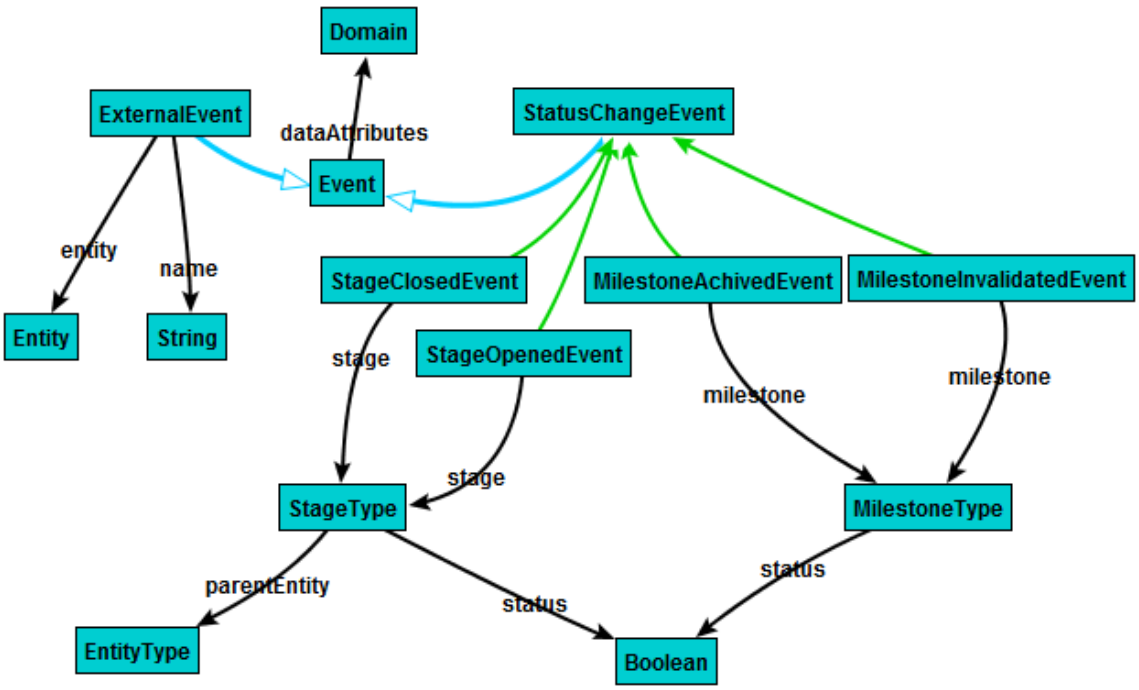


Figura 4.2.2: Parte del meta-modello che contiene gli elementi che permettono di definire gli eventi che fanno progredire l'entità nel ciclo di vita.

Oltre agli elementi mostrati negli schemi precedenti, fanno parte del meta-modello anche le numerose ECA-rule necessarie al suo funzionamento. Qui di seguito mostrerò e spiegherò due di esse, per chi desiderasse approfondire, l'implementazione completa di tutte le ECA-Rule è inclusa nell'appendice.

```
1. CloseSubStageRule in ECArule with
2.   mode m: Deferred
3.   ecarule myecarule:
4.     $
5.       parent,sub/StageType e/StageClosedEvent
   d/Individual
6.       ON Tell In(e, StageClosedEvent)
7.       IFNEW (parent in StageType) and (sub in StageType)
8.             and (e   stage parent) and (parent [body]
   sub)
9.       DO Retell A(sub, status , FALSE),
10.         CALL CreateIndividual (StClEvent, d),
11.         Tell (d in StageClosedEvent),
12.         Tell (d stage/st sub)
13.     $
14.  end
```

Il GSM Approach prevede che quando uno Stage si chiude, anche tutti i suoi eventuali Sotto-Stage si chiudano; per realizzare tale condizione abbiamo implementato l'ECA-rule CloseSubStageRule. Come si può vedere dal codice sopra riportato, questa ECA-rule viene innescata dal verificarsi dei un evento di tipo StageClosedEvent, che rappresenta appunto la chiusura di uno stage; viene poi valutata una condizione logica che sarà verificata per tutti e soli quegli stage che sono sotto-stage dello stage che si è chiuso (ovvero sono contenuti nell'attributo body di tale stage). Gli stage che verificano la condizione vengono poi modificati tramite la DO-part dell'ECA-rule; in particolare, viene cambiato il valore dello status del sotto-stage in FALSE e viene generato un evento di chiusura stage per il sotto-stage. Nel codice dell'ECA-Rule si fa riferimento agli stage come istanze della classe StageType perché questa ECA-Rule non agisce sugli oggetti del modello ma sulle loro istanze.

```
1. InvalidateMilestoneRule in ECArule with
2.   mode m: Deferred
3.   ecarule myecarule:
4.     $
5.       s/StageType e/StageOpenedEvent m/MilestoneType
6.       d/Individual
7.       ON Tell In(e, StageOpenedEvent)
8.       IFNEW (s in StageType) and (e stage s)
9.              and (m in MilestoneType)
10.              and (s [milestones] m)
11.              and (m status TRUE)
12.       DO Retell A(m, status , FALSE),
13.          CALL CreateIndividual (MlInEvent, d),
14.          Tell (d in MilestoneInvalidatedEvent),
15.          Tell (d milestone/ml m)
16.     $
17.  end
```

In alcune circostanze è possibile che uno stage venga riaperto dopo il raggiungimento di una delle proprie milestone; in questi casi è necessario che la milestone venga invalidata. Le struttura della Milestone permette l'inserimento di InvalidatingSentries, ECA-Rule che servono appunto a riportare lo status a FALSE in al verificarsi di una certa condizione. Tuttavia, queste ECA-Rule vengono definite durante la costruzione del modello e perciò non vi è modo di assicurarsi che ogni milestone disponga delle invalidating sentries necessarie ad invalidarla nel caso di riapertura dello stage cui appartiene. Per tale motivo e per ridurre il numero di ECA-Rule necessarie in un modello, abbiamo deciso di introdurre, nel meta-modello, l'ECA-Rule sopra riportata che si occupa, appunto, di invalidare tutte le milestone appartenenti a uno stage nel momento della sua apertura.

Come si può notare dall'implementazione, l'ECA-Rule viene innescata dall'evento di apertura di uno stage (StageOpenedEvent). La condizione contenuta nella IF-Part è verificata per sole e tutte le milestone appartenenti allo stage che si è aperto, che hanno status TRUE; la DO-Part si occupa di riportare lo status di queste milestone al valore FALSE e di generare un evento che lo notifichi.

Capitolo 5

FEATURE DRIVEN DEVELOPMENT

5.1 Che cos'è il Feature Driven Development

Il Feature Driven Development (o FDD) è una metodologia di sviluppo agile, progettata per permettere un monitoraggio costante del progetto garantendo la qualità del software. Questa metodologia si basa su brevi iterazioni, che consentono consegne tangibili del prodotto in tempi brevi (in genere massimo due settimane).
È stato ideato da Jeff De Luca e Peter Coad nel 1997. De Luca propose un insieme di cinque *processi* che riguardavano lo sviluppo completo basato sulle features.
La prima forma strutturata di FDD fu presentata nel libro *Java Modeling in Color with UML* di Peter Coad, Eric Lefebvre e Jeff De Luca nel 1999.

5.1.1 Processi

- *Sviluppare un modello globale*

  All'inizio dello sviluppo viene costruito un modello che tiene conto della visione, del contesto e dei requisiti che il sistema da costruire deve possedere.

  Questo modello è suddiviso in aree che vengono analizzate nel dettaglio. Per ognuna di queste viene costruito un *modello di dominio* questi ultimi verranno uniti a formare il modello globale.

- *Costruire una lista di features*

  Viene stilata una lista che riassume le funzionalità che il sistema deve avere. Questa lista verrà inoltre valutata dal cliente. Ogni funzionalità dell'elenco viene suddivisa in funzionalità più piccole per una migliore comprensione del sistema.

- *Pianificare*

  Si procede ad ordinare le funzionalità in base alla loro priorità e dipendenza e, per ognuna di esse, vengono assegnati i programmatori.

- *Design*

  Viene scelto un insieme di funzionalità dalla lista. Si procede a progettare e costruire la singola funzionalità attraverso un processo iterativo, decidendo quale verrà eseguita in ogni iterazione. Qualora fosse possibile, nulla vieta che due o più team di programmatori possano lavorare in parallelo su feature distinte.

- *Build*

  Si procede alla costruzione totale del progetto.

5.1.2 Ruoli

Il team di lavoro è strutturato in gerarchie, ci dovrebbe sempre essere un project manager e, sebbene sia un processo leggero, è necessario includere la documentazione (quanto meno il minimo necessario affinché un nuovo membro possa comprendere lo sviluppo).
Generalmente, i ruoli di cui si compone questa metodologia di sviluppo sono:

- *Chief Architect (o capo architetto)*

  Il capo architetto realizza il design generale del sistema.

- *Direttore dello sviluppo*

  Il Direttore dello sviluppo svolge le normali attività quotidiane di sviluppo e risolve problemi relativi alle risorse e conflitti nel team.

- *Capo programmatore*

  Il capo programmatore esegue l'analisi dei requisiti e progetta il lavoro.

- *Titolare delle classi*

  Il titolare delle classi è il responsabile dello sviluppo delle classi che gli sono state assegnate (può essere un programmatore o un capo programmatore). Partecipa anche alla decisione di quale classe dovrà essere inclusa nell'elenco delle funzionalità della successiva iterazione.

- *Esperto di domini*

  Figura che conosce i requisiti che dovrà avere il sistema. Fornisce un feedback continuo agli sviluppatori per garantire la consegna di un sistema completo. Questo ruolo può essere ricoperto da un utente, un cliente, un analista o da una miscela di questi.

5.1.3 Vantaggi e Svantaggi

VANTAGGI
Il team di sviluppo non spreca tempo e denaro del cliente nello sviluppo di soluzioni generali e complesse che non sono realmente un requisito del cliente
Ogni componente del prodotto finale viene testato durante lo sviluppo, perciò soddisfa i requisiti e funziona
In quanto metodologia agile, fornisce una risposta rapida ai cambiamenti dei requisiti durante lo sviluppo
Consegna continua e in tempi brevi di software funzionale
Permette il lavoro congiunto tra il cliente e il team di sviluppo, consentendo al team di avere un feedback immediato e garanzie di riuscita del software
Ottimo rapporto *costi/modifiche fatte*
Evita di svolgere lavoro non necessario rendendo il sistema più semplice
Impone un'attenzione all'eccellenza tecnica e al buon design. Permette inoltre un miglioramento continuo dei processi e del team di sviluppo

SVANTAGGI
Carenza di documentazione di progettazione. Nei sistemi di grandi dimensioni è necessario leggere le centinaia o migliaia di pagine dell'elenco dei codici sorgenti
Problemi derivati dalla comunicazione orale: poco conservativa e ambiguità nel tempo
Mancanza di riusabilità. La carenza di documentazione rende arduo il riutilizzo di codice

5.2 Modello del FDD basato sul meta-modello GSM

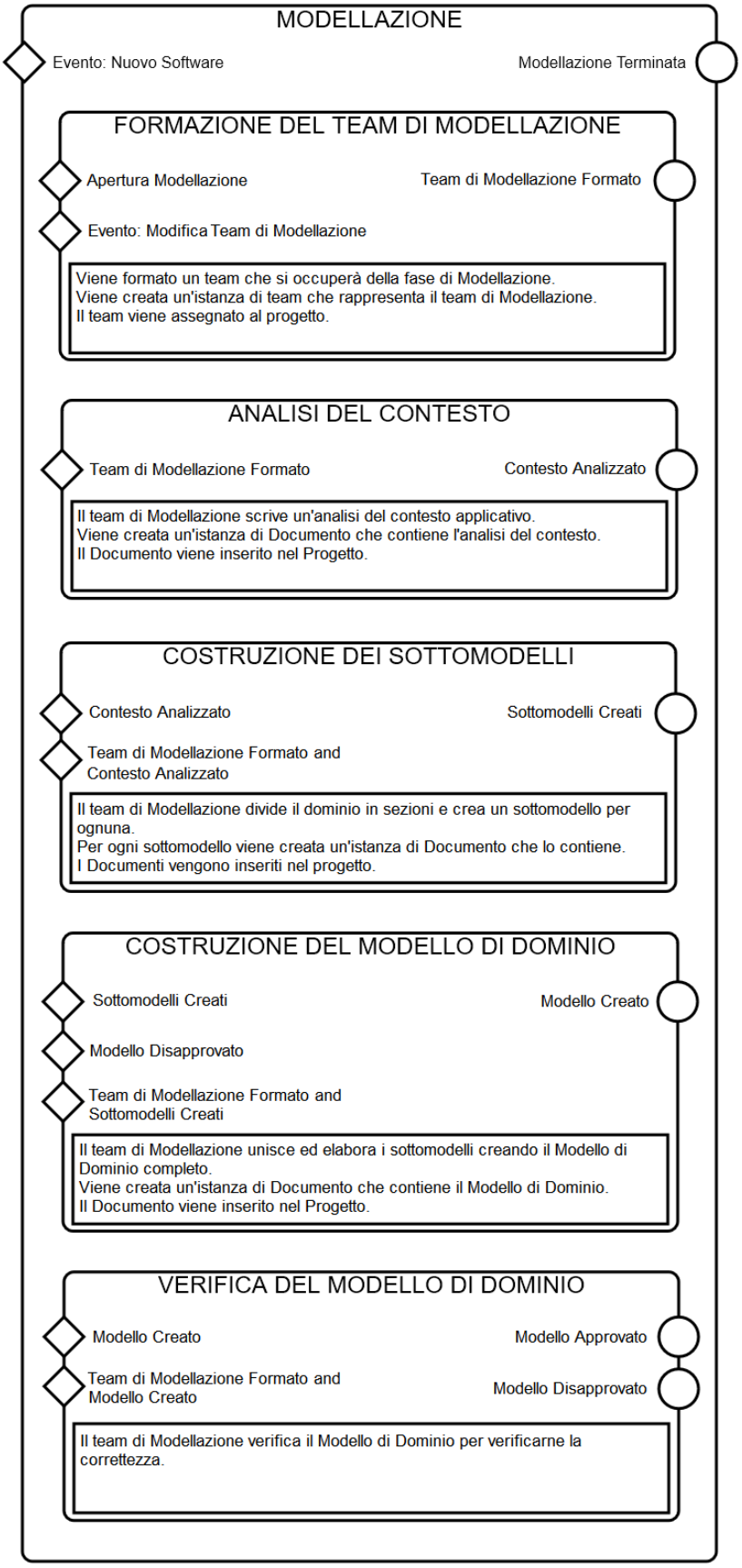
Durante la prima fase di realizzazione del modello di questa tecnica di sviluppo, ho riconosciuto in essa due entità, aventi un ciclo di vita.
La prima delle due è l'entità Progetto mentre la seconda è la singola Funzionalità.
Il Progetto racchiude quelle fasi nelle quali viene studiato e realizzato il sistema nel suo complesso, mentre la seconda interessa le fasi di realizzazione e implementazione di ogni funzionalità.

5.2.1 Entità Progetto

Per quanto riguarda il ciclo di vita del Progetto, ho riconosciuto 4 stages: *Modellazione*, *Stesura della lista delle funzionalità*, *Pianificazione* e *Sviluppo.*
Descriverò brevemente ognuno di questi stages servendomi della notazione utilizzata da GSM per rappresentare i cicli di vita.

Legenda: Ogni riquadro (con gli angoli arrotondati) rappresenta uno stage, i rombi le guard, i cerchi le milestone e gli altri riquadri i task.

- *Modellazione*



Questo primo stage è uno stage non atomico. Esso infatti racchiude altri stages e non possiede alcun task. L'evento che causa l'apertura dello stage Modellazione è un evento esterno, generato quando si vuole creare un nuovo software. Questo stage si chiuderà alla terminazione della fase di modellazione, ovvero quando viene raggiunta la milestone *Modello Approvato* dell'ultimo sotto-stage.

Ho chiamato il primo stage interno (atomico) "*Formazione del team di modellazione"*. Essendo questo necessario e primario alla creazione di un software utilizzando il FDD, ho scelto come condizione di apertura il solo vincolo che debba essere aperto lo stage che lo contiene. Dopo che il team sarà stato formato e assegnato al progetto, dovrà essere generato un evento esterno come conferma del completamento di questa fase. Ho inoltre inserito la possibilità di riaprire questo stage, per consentire eventuali modifiche al team di modellazione, grazie ad una seconda guard.

Il secondo stage interno (atomico) corrisponde alla fase di analisi del contesto. Essendo il team di modellazione a dover svolgere questo compito, è necessario che sia stato formato per poter passare a questa fase. Una volta terminati i task di questa fase, dovrà essere generato un evento esterno per confermarne il completamento. Ho reso necessaria la conferma, per dare la possibilità, in fase di sviluppo, di inserire provvisoriamente il documento contenente l'analisi del contesto all'interno del Progetto.

Il terzo sotto-stage (atomico) si apre una volta terminata l'analisi del contesto. In questa fase il team di modellazione costruisce dei sotto-modelli a partire dal dominio. Come nei casi precedenti, è necessario un evento esterno di conferma per terminare la fase.

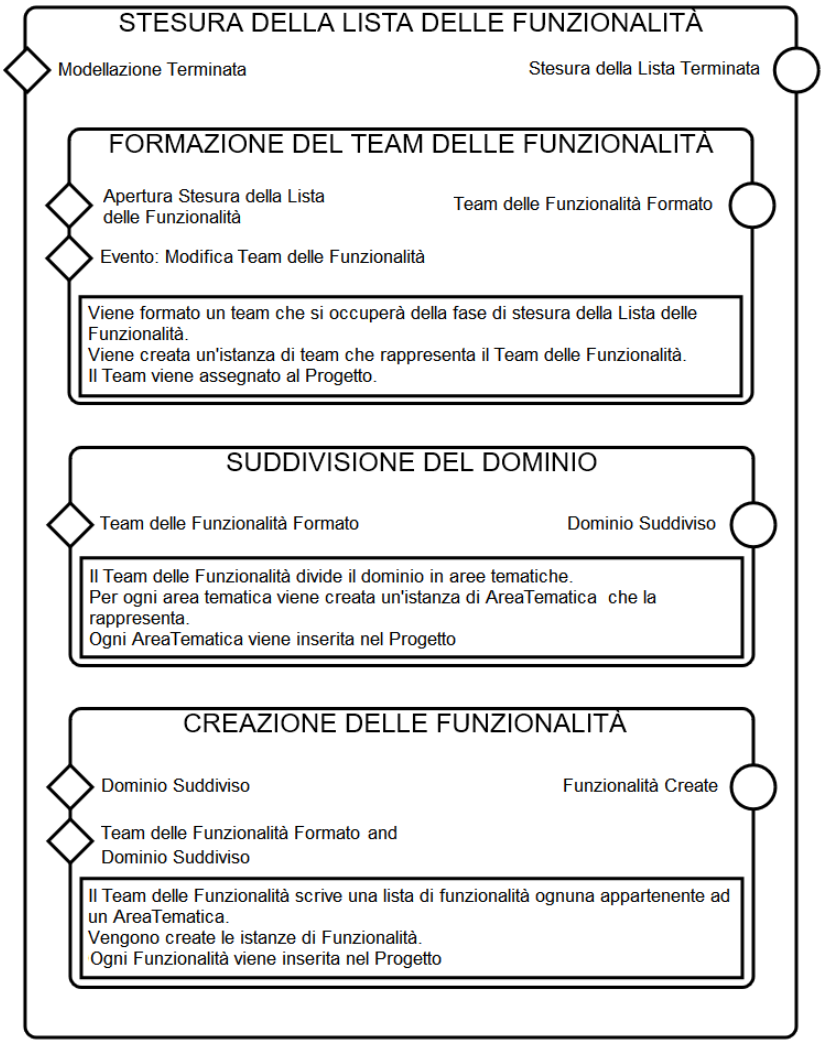
Creati i sotto-modelli, si passa alla penultima fase in cui il team costruisce un unico modello di dominio a partire dai sotto-modelli. Anche questa volta lo stage si concluderà grazie ad un evento esterno di conferma.

Al termine della costruzione del modello, si aprirà l'ultimo sotto-stage in cui il team di modellazione ne verificherà la correttezza. Questa verifica può concludersi in due modi:

1. Con l'evento di approvazione del modello, che provocherà la chiusura sia di questo sotto-stage che dello stage Modellazione.
2. Con l'evento di disapprovazione del modello. In questo caso il sotto-stage precedente si riaprirà, grazie alla sua seconda guard. In questo modo il modello potrà essere modificato e si procederà nuovamente alla verifica fino ad approvazione.

In qualsiasi momento a partire dalla chiusura del primo sotto-stage fino alla chiusura, con approvazione, dell'ultimo sotto-stage, è possibile generare l'evento esterno che richiede la modifica del team di modellazione. Questo evento provoca la chiusura del sotto-stage attualmente aperto e la riapertura (grazie alla seconda guard) del sotto-stage di formazione del team. Quando questa fase sarà completa, il sotto-stage che era stato chiuso per colpa dell'evento si riaprirà, permettendo la ripresa del normale flusso precedentemente descritto.

- *Stesura della Lista delle Funzionalità*



Il secondo grande stage che compone il ciclo di vita del progetto è chiamato *Stesura della Lista delle Funzionalità*.

Come si evince dal nome, questa è la fase in cui verranno decise le funzionalità di cui si comporrà il sistema. Per entrare in questa fase è necessario aver concluso lo stage Modellazione.

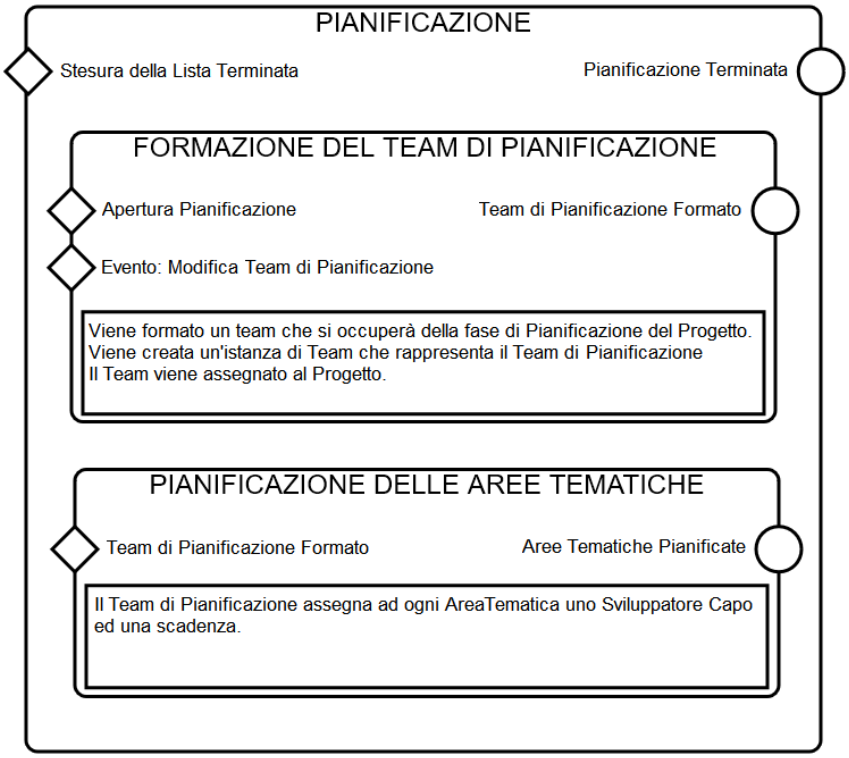
Questo stage è suddiviso in tre sotto-stage di cui parlerò qui di seguito.

L'apertura del primo sotto-stage è innescata da quella dello stage che lo contiene. Durante questa fase deve essere formato un team il cui compito sarà quello di determinare le funzionalità del progetto. Per concludere questa fase è necessario un evento esterno di conferma.

Il sotto-stage successivo, *Suddivisione del Dominio*, si apre al raggiungimento della milestone del precedente. In questa fase, il team

delle funzionalità divide il dominio del progetto in aree tematiche. Una volta terminata la suddivisione, occorrerà un evento di conferma per il raggiungimento della milestone.
Il terzo e ultimo sotto-stage si apre al raggiungimento della milestone di Suddivisione del Dominio. Durante questa fase il team crea le funzionalità appartenenti alle varie aree tematiche, e ne stila una lista. Quando la lista sarà ritenuta completa, occorrerà generare un evento esterno di conferma per raggiungere la milestone di questo sotto-stage e dello stage che lo contiene.
Come nella fase di Modellazione, anche per il team delle funzionalità è presente lo stesso meccanismo di modifica del team.

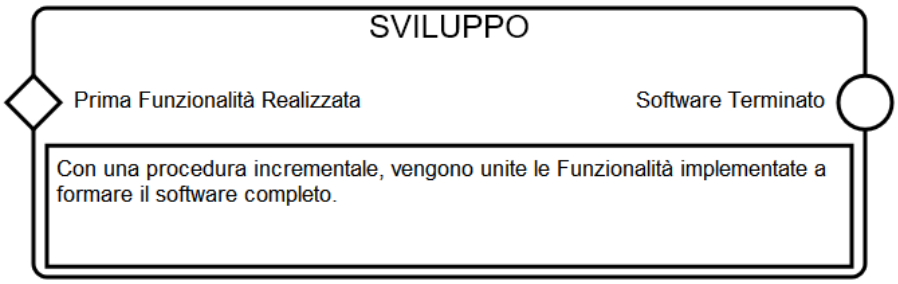
- *Pianificazione*



La terza fase del ciclo di vita del Progetto è la Pianificazione che è divisa in due sotto-stage. Questo stage si apre al raggiungimento della milestone *Stesura della Lista Terminata*. La chiusura di questo stage è determinata dal raggiungimento della milestone *Funzionalità Pianificata* da parte di

tutte le funzionalità. È questa una delle occasioni in cui il ciclo di vita del Progetto e quello della Funzionalità si influenzano.
Il primo sotto-stage prevede la formazione di un team che si occuperà della pianificazione del progetto. Lo stage si apre conseguentemente all'apertura di quello che lo contiene. Per raggiungere la milestone occorre che venga generato l'evento esterno di conferma.
Il raggiungimento di tale milestone provoca l'apertura del secondo, e ultimo sotto-stage, *Pianificazione delle Aree Tematiche*. Come si evince dal nome, il team deve determinare la scadenza per la realizzazione di ciascuna area tematica e deve assegnarla ad un capo-programmatore. Quando verrà lanciato l'apposito evento di conferma, il sotto-stage si chiuderà. Questo provocherà l'apertura del primo stage del ciclo di vita di ogni Funzionalità abbinata al Progetto.
Come nei casi precedenti, anche qui è prevista la possibilità di modificare il team.

- *Sviluppo*

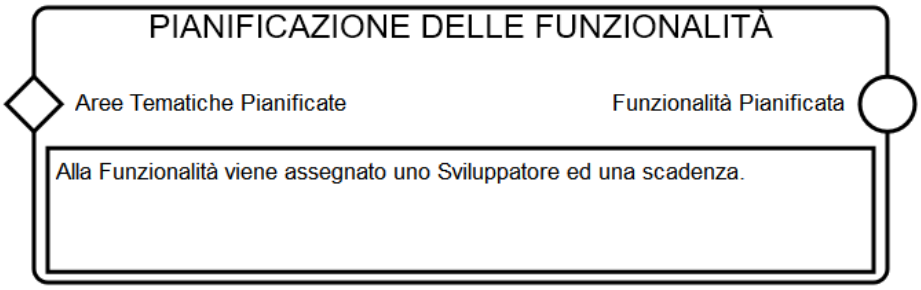


Questo è l'ultimo stage del ciclo di vita del Progetto. Si apre non appena una delle funzionalità completa il suo ciclo di vita, ovvero ne viene terminata la realizzazione. Questo stage, infatti, rappresenta la fase in cui le varie funzionalità vengono unite a formare il progetto completo. Per il raggiungimento della milestone è necessario un evento esterno di conferma e che, ovviamente, tutte le Funzionalità abbiano terminato il proprio ciclo di vita.

5.2.3 Entità Funzionalità

Per quanto riguarda il ciclo di vita della Funzionalità, ho riconosciuto 3 stages: *Pianificazione della Funzionalità*, *Progettazione* e *Realizzazione.*

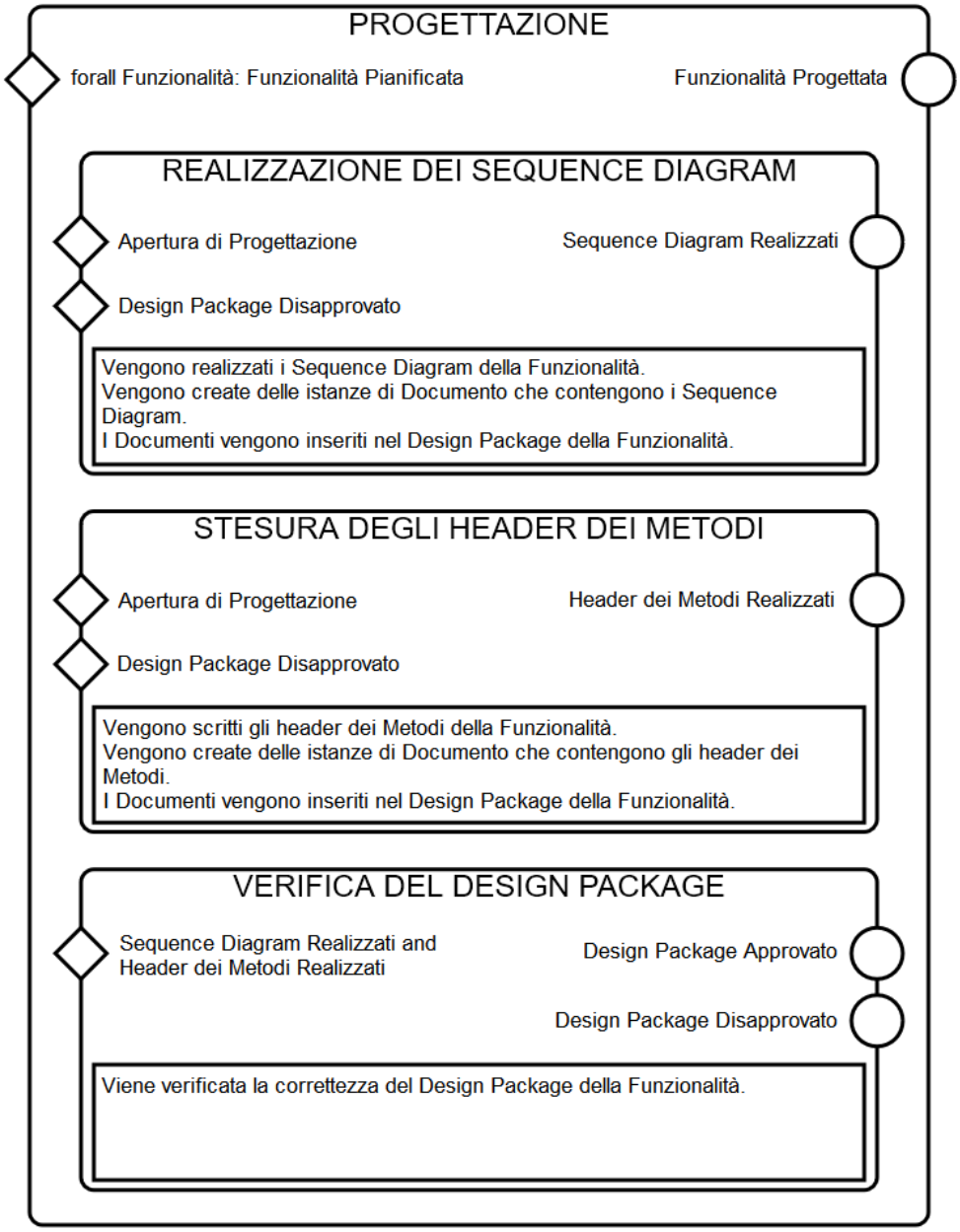
- *Pianificazione della Funzionalità*



In questo primo stage avviene la pianificazione della funzionalità. Quest'operazione consiste nello stabilire una data entro ci la funzionalità dovrà essere terminata e assegnarne lo sviluppo a un programmatore.

Come già accennato, questo stage viene aperto al raggiungimento della milestone Aree Tematiche Pianificate, che trova all'interno della terza parte del ciclo di vita del Progetto. Per raggiungere la milestone è necessario fornire una conferma tramite un evento esterno.

- *Progettazione*



Questo secondo Stage del ciclo di vita della Funzionalità si apre quando viene raggiunta la milestone dello stage Pianificazione Funzionalità.
Lo stage si divide in tre sotto-stage. Il raggiungimento della milestone avviene come conseguenza alla chiusura dell'ultimo sotto-stage con l'approvazione del Design Package.
Il primo dei sotto-stage prevede la realizzazione dei sequence diagram della funzionalità, che, insieme agli header dei metodi, andranno a formare il Design Package della Funzionalità.

Per terminare questa fase sarà necessario generare un evento esterno di conferma.

Il secondo sotto-stage consiste nella stesura degli header dei metodi che dovranno essere implementati per realizzare la funzionalità. Anche in questo caso è necessario in evento di conferma per raggiungere la milestone.

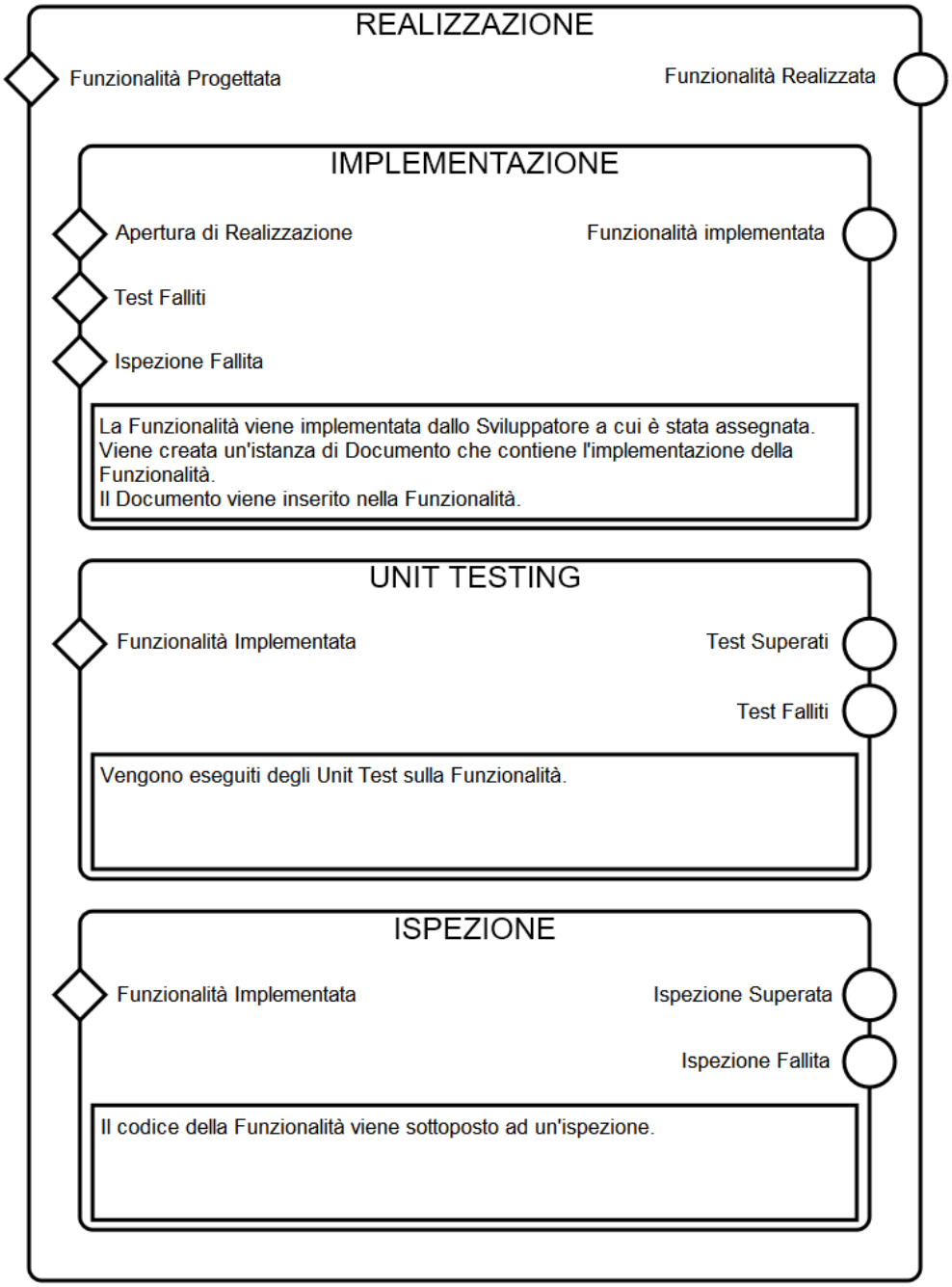
Ho ritenuto opportuno che questi due sotto-stage si aprissero nello stesso momento per dare la possibilità al programmatore di lavorare alternatamente ai sequence diagram agli header.

L'apertura del terzo sotto-stage avviene quando entrambi i precedenti hanno raggiunto le rispettive milestone. Durante questo stage deve essere effettuata una verifica della correttezza del Design Package appena prodotto.

Se il Design Package è corretto deve essere generato l'evento esterno di approvazione che farà validare la milestone Design Package Approvato e, come conseguenza di questo, farà validare la milestone dello stage Progettazione.

Se invece il Design Package non venisse approvato deve essere generato l'evento apposito che porterà al raggiungimento della milestone Design Package Disapprovato. Questa eventualità determina la riapertura dei due sotto-stage precedenti così da consentire la modifica dei documenti.

- *Realizzazione*



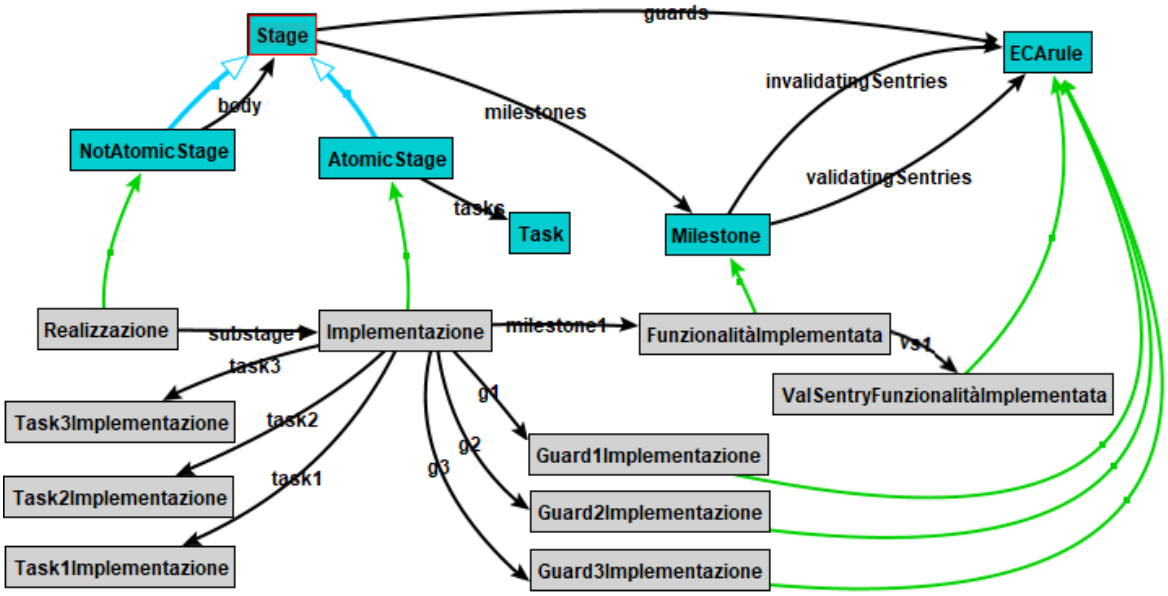
Il terzo stage del ciclo di vita della *Funzionalità* racchiude le fasi di vera e propria realizzazione del software. Questo stage si apre non appena la milestone *Funzionalità Progettata* viene raggiunta e si conclude solo quando saranno validate entrambe le milestone *Test Superati* e *Ispezione Superata*. I sotto-stage di cui si compone sono tre, descritti qui di seguito.

Il primo rappresenta la fase in cui il programmatore implementa la funzionalità. Questo sotto-stage si apre insieme allo stage che lo contiene, e la sua milestone viene raggiunta quando viene generato l'evento di conferma dell'implementazione.

Il secondo e il terzo sotto-stage si aprono al raggiungimento della milestone *Funzionalità Implementata*. Il secondo prevede che il programmatore verifichi la correttezza dell'implementazione tramite unit-testing. Il terzo prevede un'ispezione manuale del codice per valutarne correttezza e qualità. Ho ritenuto opportuno che questi due sotto-stage fosse aperti in contemporanea per permettere al programmatore di eseguire le due operazioni nell'ordine che ritiene più opportuno o alternarle. Entrambi i sotto-stage possiedono una milestone positiva e una negativa; a seconda dell'esito di ogni controllo dovrà essere generato il rispettivo evento esterno che determinerà il raggiungimento della milestone.

La validazione di una delle milestone negative porta alla riapertura del primo sotto-stage, per permettere la correzione del codice, e alla chiusura dell'altro sotto-stage, se aperto. La riconferma dell'implementazione porterà alla riapertura di entrambi i sotto-stage di verifica (il secondo e il terzo sotto-stage). Quando entrambe le verifiche avranno esito positivo verrà raggiunta la milestone Funzionalità Realizzata con cui si concluderà il ciclo di vita della Funzionalità.

A seguire riporto come esempio lo schema del sotto-stage "Implementazione", realizzato con il tool grafico fornito da ConceptBase.



"Implementazione", come già detto, è una delle fasi dello sviluppo FDD, che possiede delle pre-condizioni e delle post-condizioni, perciò ho deciso di implementarlo come istanza di Stage. Infatti, Stage possiede (frecce nere) due tipi di attributi: guards e milestones.

In base alla nostra implementazione del meta-modello però, ci sono due specializzazioni (frecce blu) di Stage: AtomicStage e NotAtomicStage.

Dal momento che durante la fase di "Implementazione" vengono svolti dei compiti (i Task), si è reso necessario renderla istanza (freccia verde) di AtomicStage, in quanto, questo, possiede l'attributo "tasks".

Per poter dichiarare conclusa questa fase, deve essere completata l'implementazione della funzionalità. Per tale motivo, ho assegnato una Milestone a "Implementazione" ovvero "FunzionalitàImplementata". Come si vede infatti dal grafo, "Implementazione" possiede una milestone, la quale è istanza di Milestone.

L'implementazione di una Milestone, necessita di almeno una "validatingSentries" per poter funzionare, e può avere delle "invalidatingSentries", entrambe espresse mediante ECA-rules.

Infatti, come si può notare dal grafo, "FunzionalitàImplementata" possiede un attributo vs1 (istanza di "validatingSentries") contenente "ValSentryFunzionalitàImplementata", istanza di ECArule.

A titolo esemplificativo, riporto di seguito il codice di una delle ECA-Rule implementate nel modello, ovvero "ValSentryFunzionalitàImplementata". L'intera implementazione del modello, comprese le ECA-Rule, si può trovare nell'appendice.

```
ValSentryFunzionalitàImplementata in ECArule with
   mode m: Deferred
   ecarule myecarule:
      $
        ev/ExternalEvent m/MilestoneType s/StageType
        f/Funzionalità doc/Documento d/Individual
        ON Tell (ev in EventoFunzionalitàImplementata)
        IFNEW (f in Funzionalità) and (ev funzionalità f)
               and (s in Implementazione)
               and (s parentEntity f)
               and (m in FunzionalitàImplementata)
               and (s [milestones] m)
               and (doc in Documento) and (f codice doc)
         DO Retell (m status TRUE),
            CALL CreateIndividual (MlAcEvent, d),
            Tell (d in MilestoneAchivedEvent),
            Tell (d milestone/ml m)
       $
end
```

Questa ECA-Rule, come mostrato in precedenza, è la validating sentry dello stage "Implementazione". Il suo scopo è quello di validare la milestone "FunzionalitàImplementata" quando viene generato l'evento che segnala il completamento dell'implementazione della funzionalità. Deve inoltre controllare che sia stato assegnato alla funzionalità un documento contenente il codice di quest'ultima.
Per realizzare tutto ciò ho creato questa ECA-Rule che viene innescata (ON-part) dall'evento esterno "EventoFunzionalitàImplementata"; all'interno della IF-part della regola, ho inserito una condizione che risulta verificata solo nel caso la funzionalità cui è riferito l'evento possieda il

documento contenente il codice. Se la condizione è verificata, la milestone "FunzionalitàImplementata" appartenente al ciclo di vita di quella funzionalità viene validata e viene creato un evento interno che notifica tale cambiamento di stato (DO-part).

CONCLUSIONI

La realizzazione di questo progetto era volta alla costruzione di un meta-modello che permettesse di definire modelli basati sul concetto di BEL e GSM.

Dopo aver approfondito il software di riferimento e le possibilità offerte dal linguaggio nella definizione di ECA rules, ci siamo accorti che il meta-modello generabile poteva essere utilizzato su una scala più ampia rispetto a creare modelli per attività di business.

Con una corretta implementazione del modello è infatti possibile generare modelli di metodologie di sviluppo software (come mostrato da me in questa relazione) ma non solo.

È infatti possibile modellare qualunque "sistema" in cui sia identificabile un ciclo di vita: addirittura un parsificatore di stringhe! E di conseguenza un compilatore. Ad esempio, ciò potrebbe essere generabile identificando con uno stage ogni elemento della grammatica e come guards i vincoli grammaticali per i quali diventa possibile aggiungere un certo costrutto.

BIBLIOGRAFIA E SITOGRAFIA

- Richard Hull, Elio Damaggio, Fabiana Fournier, Manmohan Gupta, Fenno (Terry) Heath III, Stacy Hobson, Mark Linehan, Sridhar Maradugu, Anil Nigam, Piyawadee Sukaviriya, Roman Vaculin, "*Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles",* International Workshop on Web Services and Formal Methods (WS-FM), 2010: Web Services and Formal Methods pp 1-24.
- Team di sviluppo: http://conceptbase.sourceforge.net/cbteam.html, "http://conceptbase.sourceforge.net/", ConceptBase.cc.
- Wikipedia, "https://it.wikipedia.org/wiki/Feature_Driven_Development", Feature Driven Development.
- Valesca, "http://metodologiafdd.blogspot.com/", Metodología FDD (Feature Driven Development / Desarrollo Basado en Funciones), 12 Giugno 2012.

APPENDICE

Implementazione del meta-modello

Le classi

```
Entity in Class with
  attribute, necessary, single
    name: String
  attribute, necessary
    lifeCycle: Stage
  attribute
    dataAttributes: Domain
end
```

```
EntityType in Class end
```

```
Milestone in Class with
  attribute, necessary, single
    name: String
  attribute, necessary
    validatingSentries: ECArule
  attribute
    invalidatingSentries: ECArule
end
```

```
MilestoneType in Class with
  attribute, single
    status: Boolean
end
```

```
Stage in Class with
  attribute, necessary, single
    name : String
  attribute, necessary
    milestones : Milestone;
    guards : ECArule
  constraint
    parentStageConstraint:
       $ forall s/Stage
         forall ns1,ns2/NotAtomicStage (ns1 body s)
         and (ns2 body s) ==> (ns1 = ns2)
       $
end
```

```
StageType in Class with
  attribute, single
    status : Boolean;
```

```
38.     parentEntity : EntityType
39. end
```

```
40. NotAtomicStage in Class isA Stage with
41.   attribute, necessary
42.     body: Stage
43. end
```

```
44. AtomicStage in Class isA Stage with
45.   attribute, necessary
46.     tasks: Task
47.   constraint
48.      taskTypeBConst:
49.        $ forall s/AtomicStage
50.          forall t,t2/Task (t type TaskTypeB)
51.          and (t2 type TaskTypeB) and (s tasks t)
52.          and (s tasks t2) ==> (t = t2)
53.        $;
54.      taskTypeCConst:
55.        $ forall s/AtomicStage
56.          forall t,t2/Task (t type TaskTypeC)
57.          and (t2 type TaskTypeC) and (s tasks t)
58.          and (s tasks t2) ==> (t = t2)
59.        $;
60.      taskTypeDConst:
61.        $ forall s/AtomicStage
62.          forall t,t2/Task (t type TaskTypeD)
63.          and (t2 type TaskTypeD) and (s tasks t)
64.          and (s tasks t2) ==> (t = t2)
65.        $;
66.      taskTypeEConst:
67.        $ forall s/AtomicStage
68.          forall t,t2/Task (t type TaskTypeE)
69.          and (t2 type TaskTypeE) and (s tasks t)
70.          and (s tasks t2) ==> (t = t2)
71.        $
72. end
```

```
73. Task in Class with
74.   attribute, necessary, single
75.     type: TaskType;
76.     description: String
77. end
```

```
78. TaskType in Class end
```

```
79. Event in Class with
80.   attribute
81.     dataAttributes : Domain
82. end
```

```
83. ExternalEvent in Class isA Event with
```

```
84.   attribute, necessary, single
85.     name : String
86.   attribute, necessary
87.     entity : Entity
88.   rule
89.     entityIsNecessary :
90.       $ forall a/ExternalEvent!entity
91.         In(a,Proposition!necessary)
92.       $
93. end
```

```
94. StatusChangeEvent in Class isA Event end
```

```
95. MilestoneAchivedEvent in StatusChangeEvent with
96.   attribute, single, necessary
97.     milestone: MilestoneType
98. end
```

```
99. MilestoneInvalidatedEvent in StatusChangeEvent with
100.   attribute, single, necessary
101.     milestone: MilestoneType
102. end
```

```
103. StageOpenedEvent in StatusChangeEvent with
104.   attribute, single, necessary
105.     stage: StageType
106. end
```

```
107. StageClosedEvent in StatusChangeEvent with
108.   attribute, single, necessary
109.     stage: StageType
110. end
```

```
111. Domain in Class with
112.   attribute
113.     fields : Domain
114. end
```

```
115. Integer in Domain end
```

```
116. String in Domain end
```

```
117. Date in Domain,Class with
118.   fields
119.     year: Integer;
120.     month: Integer;
121.     day: Integer
122.   constraint
123.     yearCostraint :
```

```
124.       $ forall d/Date i/Integer
125.         (d year i) ==> (i > 2017) $;
126.     monthCostraint :
127.       $ forall d/Date i/Integer
128.         (d month i) ==> (i > 0) and (i < 13) $;
129.     dayCostraint :
130.       $ forall d/Date i/Integer
131.         (d day i) ==> (i > 0) and (i < 32) $
132. end
```

```
133. Boolean in Domain end
```

```
134. Real in Domain end
```

```
135. TaskTypeA in TaskType end
```

```
136. TaskTypeB in TaskType end
```

```
137. TaskTypeC in TaskType end
```

```
138. TaskTypeD in TaskType end
```

```
139. TaskTypeE in TaskType end
```

```
140. Class with
141.   constraint
142.     necConstraint :
143.       $ forall c,d/Proposition p/Proposition!necessary
144.         x,m/VAR
145.         P(p,c,m,d) and (x in c) ==>
146.         exists y/VAR (y in d) and (x m y)
147.       $
148. end
```

```
149. Class with
150.   constraint
151.     singleConstraint :
152.       $ forall c,d/Proposition p/Proposition!single x,m/VAR
153.         P(p,c,m,d) and (x in c) ==>
154.         ( forall a1,a2/VAR
155.           (a1 in p) and (a2 in p) and Ai(x,m,a1)
156.           and Ai(x,m,a2) ==> (a1=a2)
157.         )
158.       $
159. end
```

```
160. StClEvent end
```

```
161. StOpEvent end
```

```
162. MlAcEvent end
```

```
163. MlInEvent end
```

```
164. StageInstance end
```

```
165. MilestoneInstance end
```

ECA-Rules

```
166. StageTypeRule in ECArule with
167.   mode m: Deferred
168.   ecarule myecarule:
169.     $ s/Stage
170.       ON Tell In(s, Stage)
171.       DO Tell (s isA StageType)
172.     $
173. end
```

```
174. StageStatusRule in ECArule with
175.   mode m: Deferred
176.   ecarule myecarule :
177.        $ s/StageType
178.          ON Tell In(s, StageType)
179.          DO Tell A(s, status, FALSE)
180.        $
181. end
```

```
182. MilestoneTypeRule in ECArule with
183.   mode m: Deferred
184.   ecarule myecarule:
185.            $ m/Milestone
186.              ON Tell In(m, Milestone)
187.              DO Tell (m isA MilestoneType)
188.            $
189. end
```

```
190. MilestoneStatusRule in ECArule with
191.   mode m: Deferred
192.   ecarule myecarule :
193.            $ m/MilestoneType
194.              ON Tell In(m, MilestoneType)
195.              DO Tell A(m, status , FALSE)
196.            $
```

```
197. end
```

```
198. EntityTypeRule in ECArule with
199.   mode m: Deferred
200.   ecarule myecarule:
201.            $ e/Entity
202.              ON Tell In(e, Entity)
203.              DO Tell (e isA EntityType)
204.            $
205. end
```

```
206. GenerateStageIstancesRule in ECArule with
207.   mode m: Deferred
208.   ecarule myecarule :
209.     $ e/EntityType s/Stage me/Entity d,a/Individual
210.       p/Proposition
211.       ON Tell In(e, EntityType)
212.       IFNEW (me in Entity) and (e in me) and (s in Stage)
213.             and (p in Entity!lifeCycle) and From(p,me)
214.             and To(p,s)
215.       DO CALL CreateIndividual (StageInstance, d),
216.          Tell (d in s),
217.          CALL CreateAttribute(Proposition!attribute,e,d,a),
218.          Tell (a in p),
219.          Tell (d parentEntity/parentE e)
220.     $
221. end
```

```
222. GenerateSubStageIstancesRule in ECArule with
223.   mode m: Deferred
224.   ecarule myecarule :
225.     $ ms,msb/Stage s/StageType e/EntityType d,a/Individual
226.       p/Proposition
227.       ON Tell In(s, StageType)
228.       IFNEW (ms in NotAtomicStage) and (s in ms)
229.             and (msb in Stage)
230.             and (p in NotAtomicStage!body) and From(p,ms)
231.             and To(p,msb) and (e in EntityType)
232.             and (s parentEntity e)
233.       DO CALL CreateIndividual (StageInstance, d),
234.          Tell (d in msb),
235.          CALL CreateAttribute(Proposition!attribute,s,d,a),
236.          Tell (a in p),
237.          Tell (d parentEntity/parentE e)
238.    $
239. end
```

```
240. GenerateMilestoneIstancesRule in ECArule with
241.   mode m: Deferred
242.   ecarule myecarule :
243.     $ ms/Stage s/StageType m/Milestone d,a/Individual
244.       p/Proposition
245.       ON Tell In(s, StageType)
246.       IFNEW (ms in Stage) and (s in ms)
```

```
247.             and (m in Milestone)
248.             and (p in Stage!milestones) and From(p,ms)
249.             and To(p,m)
250.       DO CALL CreateIndividual (MilestoneInstance, d),
251.          Tell (d in m),
252.          CALL CreateAttribute(Proposition!attribute,s,d,a),
253.          Tell (a in p)
254.    $
255. end
```

```
256. CloseStageRule in ECArule with
257.   mode m: Deferred
258.   ecarule myecarule:
259.     $ s/StageType m/MilestoneType e/MilestoneAchivedEvent
260.       d/Individual
261.       ON Tell In(e, MilestoneAchivedEvent)
262.       IFNEW (e in MilestoneAchivedEvent)
263.             and (m in MilestoneType) and (e milestone m)
264.             and (s in StageType) and (s [milestones] m)
265.             and (s status TRUE)
266.       DO Retell A(s, status , FALSE),
267.          CALL CreateIndividual (StClEvent, d),
268.          Tell (d in StageClosedEvent),
269.          Tell (d stage/st s)
270.       ELSE reject
271.    $
272. end
```

```
273. InvalidateMilestoneRule in ECArule with
274.   mode m: Deferred
275.   ecarule myecarule:
276.     $ s/StageType e/StageOpenedEvent m/MilestoneType
277.       d/Individual
278.       ON Tell In(e, StageOpenedEvent)
279.       IFNEW (s in StageType) and (e stage s)
280.             and (m in MilestoneType) and (s [milestones] m)
281.             and (m status TRUE)
282.       DO Retell A(m, status , FALSE),
283.          CALL CreateIndividual (MlInEvent, d),
284.          Tell (d in MilestoneInvalidatedEvent),
285.          Tell (d milestone/ml m)
286.    $
287. end
```

```
288. CloseSubStageRule in ECArule with
289.   mode m: Deferred
290.   ecarule myecarule:
291.     $ parent,sub/StageType e/StageClosedEvent d/Individual
292.       ON Tell In(e, StageClosedEvent)
293.       IFNEW (parent in StageType) and (sub in StageType)
294.             and (e stage parent) and (parent [body] sub)
295.       DO Retell A(sub, status , FALSE),
296.          CALL CreateIndividual (StClEvent, d),
297.          Tell (d in StageClosedEvent),
298.          Tell (d stage/st sub)
```

```
299.    $
300. end
```

```
301. SubStageType in QueryClass isA StageType with
302.   constraint
303.     c1: $ exists p/StageType (p [body] this) $
304. end
```

```
305. ParentIsOpenRule in ECArule with
306.   mode m: Deferred
307.   ecarule myecarule:
308.     $ s,p/StageType e/StageOpenedEvent
309.       ON Tell In(e, StageOpenedEvent)
310.       IFNEW (s in StageType) and (e stage s)
311.             and (s in SubStageType)
312.             and (p in StageType) and (p [body] s)
313.             and (p status FALSE)
314.       DO reject
315.     $
316. end
```

Implementazione del modello

Le classi

```
Progetto in Entity with
  name
    nome: "Progetto sorftware"
  dataAttributes, necessary, single
    nomeProgetto: String
  dataAttributes, single
    teamModellazione: Team;
    teamFunzionalità: Team;
    teamPianificazione: Team;
     resocontoContesto: Documento;
     modelloDominio: Documento
   dataAttributes
     sottoModelli: Documento;
     areeTematiche: AreaTematica;
     funzionalità: Funzionalità
   lifeCycle
     stage1: Modellazione;
     stage2: StesuraListaFunzionalità;
     stage3: Pianificazione;
     stage4: Sviluppo
end
```

```
Funzionalità in Entity with
  name
    nome: "Funzionalità del software"
  dataAttributes, necessary, single
    nomeFunzionalità: String;
    areaTematica: AreaTematica
  dataAttributes, single
    sviluppatore: Persona;
    designPackage: DesignPackage;
    codice: Documento;
    dataTermine: Date
  lifeCycle
    stage1: PianificazioneFunzionalità;
    stage2: Progettazione;
    stage3: Realizzazione
end
```

```
Funzionalità in Domain end
```

```
Team in Domain with
  attribute, necessary, single
    capo: Persona
  attribute, necessary
    membri: Persona
end
```

```
Persona in Domain with
  attribute, necessary, single
    nome: String;
```

```
    ruolo: String
end
```

```
Documento in Domain with
  attribute, necessary, single
    nome: String
  attribute, necessary
    file: String
end
```

```
AreaTematica in Domain with
  attribute, necessary, single
    nome: String
  attribute, single
    sviluppatoreCapo: Persona;
    dataTermine: Date
end
```

```
DesignPackage in Domain with
  attribute
    sequenceDiagrams: Documento;
    headerMetodi: Documento
end
```

```
Modellazione in NotAtomicStage with
  name
    nome: "Fase di Modellazione"
  milestones
    milestone1: ModellazioneTerminata
  guards
    guard1: Guard1Modellazione
  body
    substage1: FormazioneTeamModellazione;
    substage2: AnalisiContesto;
    substage3: CostruzioneSottoModelli;
    substage4: CostruzioneModello;
    substage5: VerificaModello
end
```

```
ModellazioneTerminata in Milestone with
    name
        nome: "Modellazione Terminata"
    validatingSentries
        val1: ValSentryModellazioneTerminata
end
```

```
StesuraListaFunzionalità in NotAtomicStage with
  name
    nome: "Fase di stesura della Lista delle Funzionalità"
  milestones
    milestone1: StesuraListaTerminata
  guards
    guard1: Guard1StesuraListaFunzionalità
  body
    substage1: FormazioneTeamFunzionalità;
    substage2: SuddivisioneDominio;
```

```
    substage3: CreazioneFunzionalità
end
```

```
StesuraListaTerminata in Milestone with
    name
      nome: "Stesura Lista Terminata"
    validatingSentries
      val1: ValSentryStesuraListaTerminata
end
```

```
Pianificazione in NotAtomicStage with
  name
    nome: "Fase di Pianificazione del Progetto"
  milestones
    milestone1: PianificazioneTerminata
  guards
    guard1: Guard1Pianificazione
  body
    substage1: FormazioneTeamPianificazione;
    substage2: PianificazioneAreeTematiche
end
```

```
PianificazioneTerminata in Milestone with
    name
        nome: "Pianificazione Terminata"
    validatingSentries
        val1: ValSentryPianificazioneTerminata
end
```

```
Sviluppo in AtomicStage with
  name
    nome: "Fase di Sviluppo del Software"
  milestones
    milestone1: SoftwareTerminato
  guards
    guard1: Guard1Sviluppo
  tasks
    task1: Task1Sviluppo
end
```

```
SoftwareTerminato in Milestone with
    name
        nome: "Software Terminato"
    validatingSentries
        val1: ValSentrySoftwareTerminato
end
```

```
PianificazioneFunzionalità in AtomicStage with
  name
    nome: "Fase di Pianificazione della Funzionalità"
  milestones
    milestone1: FunzionalitàPianificata
  guards
    guard1: Guard1PianificazioneFunzionalità
  tasks
    task1: Task1PianificazioneFunzionalità
```

```
148. end
```

```
149. FunzionalitàPianificata in Milestone with
150.     name
151.         nome: "Funzionalità Pianificata"
152.     validatingSentries
153.         val1: ValSentryFunzioalitàPianificata
154. end
```

```
155. Progettazione in NotAtomicStage with
156.   name
157.     nome: "Fase di Progettazione della Funzionalità"
158.   milestones
159.     milestone1: FunzionalitàProgettata
160.   guards
161.     guard1: Guard1Progettazione
162.   body
163.     substage1: RealizzazioneSequenceDiagram;
164.     substage2: StesuraHeaderMetodi;
165.     substage3: VerificaDesignPackage
166. end
```

```
167. FunzionalitàProgettata in Milestone with
168.     name
169.         nome: "Funzionalità Progettata"
170.     validatingSentries
171.         val1: ValSentryFunzionalitàProgettata
172. end
```

```
173. Realizzazione in NotAtomicStage with
174.   name
175.     nome: "Fase di Realizzazione della Funzionalità"
176.   milestones
177.     milestone1: FunzionalitàRealizzata
178.   guards
179.     guard1: Guard1Realizzazione
180.   body
181.     substage1: Implementazione;
182.     substage2: UnitTesting;
183.     substage3: Ispezione
184. end
```

```
185. FunzionalitàRealizzata in Milestone with
186.     name
187.         nome: "Funzionalità Realizzata"
188.     validatingSentries
189.         val1: ValSentryFunzionalitàRealizzata
190. end
```

```
191. FormazioneTeamModellazione in AtomicStage with
192.   name
193.     nome: "Fase di Formazione del Team di Modellazione"
194.   milestones
195.     milestone1: TeamModellazioneFormato
196.   guards
197.     guard1: Guard1FormazioneTeamModellazione;
```

```
198.       guard2: Guard2FormazioneTeamModellazione
199.   tasks
200.     task1: Task1FormazioneTeamModellazione;
201.     task2: Task2FormazioneTeamModellazione;
202.     task3: Task3FormazioneTeamModellazione
203. end
```

```
204. TeamModellazioneFormato in Milestone with
205.     name
206.         nome: "Team Modellazione Formato"
207.     validatingSentries
208.         val1: ValSentryTeamModellazioneFormato
209. end
```

```
210. AnalisiContesto in AtomicStage with
211.   name
212.     nome: "Fase di Analisi del Contesto applicativo"
213.   milestones
214.     milestone1: ContestoAnalizzato
215.   guards
216.     guard1: Guard1AnalisiContesto
217.   tasks
218.     task1: Task1AnalisiContesto;
219.     task2: Task2AnalisiContesto;
220.     task3: Task3AnalisiContesto
221. end
```

```
222. ContestoAnalizzato in Milestone with
223.     name
224.         nome: "Contesto Analizzato"
225.     validatingSentries
226.         val1: ValSentryContestoAnalizzato
227. end
```

```
228. CostruzioneSottoModelli in AtomicStage with
229.   name
230.     nome: "Fase di Costruzione dei Sotto-Modelli di dominio"
231.   milestones
232.     milestone1: SottoModelliCreati
233.   guards
234.     guard1: Guard1CostruzioneSottoModelli;
235.       guard2: Guard2CostruzioneSottoModelli
236.   tasks
237.     task1: Task1CostruzioneSottoModelli;
238.     task2: Task2CostruzioneSottoModelli;
239.     task3: Task3CostruzioneSottoModelli
240. end
```

```
241. SottoModelliCreati in Milestone with
242.     name
243.         nome: "Sotto Modelli Creati"
244.     validatingSentries
245.         val1: ValSentrySottoModelliCreati
246. end
```

```
247. CostruzioneModello in AtomicStage with
```

```
248.   name
249.     nome: "Fase di Costruzione del Modello di dominio"
250.   milestones
251.     milestone1: ModelloCreato
252.   guards
253.     guard1: Guard1CostruzioneModello;
254.       guard2: Guard2CostruzioneModello
255.   tasks
256.     task1: Task1CostruzioneModello;
257.     task2: Task2CostruzioneModello;
258.     task3: Task3CostruzioneModello
259. end
```

```
260. ModelloCreato in Milestone with
261.     name
262.         nome: "Modello Creato"
263.     validatingSentries
264.         val1: ValSentryModelloCreato
265. end
```

```
266. VerificaModello in AtomicStage with
267.   name
268.     nome: "Fase di Verifica del Modello di dominio"
269.   milestones
270.     milestone1: ModelloApprovato;
271.     milestone2: ModelloDisapprovato
272.   guards
273.     guard1: Guard1VerificaModello
274.   tasks
275.     task1: Task1VerificaModello
276. end
```

```
277. ModelloApprovato in Milestone with
278.     name
279.         nome: "Modello Approvato"
280.     validatingSentries
281.         val1: ValSentryModelloApprovato
282. end
```

```
283. ModelloDisapprovato in Milestone with
284.     name
285.         nome: "Modello Disapprovato"
286.     validatingSentries
287.         val1: ValSentryModelloDisapprovato
288. end
```

```
289. FormazioneTeamFunzionalità in AtomicStage with
290.   name
291.     nome: "Fase di Formazione del Team che incaricato della Lista
   delle Funzionalità"
292.   milestones
293.     milestone1: TeamFunzionalitàFormato
294.   guards
295.     guard1: Guard1FormazioneTeamFunzionalitr
296.   tasks
297.     task1: Task1FormazioneTeamFunzionalità;
298.     task2: Task2FormazioneTeamFunzionalità;
```

```
299.     task3: Task3FormazioneTeamFunzionalità
300. end
```

```
301. TeamFunzionalitàFormato in Milestone with
302.     name
303.         nome: "Team Funzionalità Formato"
304.     validatingSentries
305.         val1: ValSentryTeamFunzionalitàFormato
306. end
```

```
307. SuddivisioneDominio in AtomicStage with
308.   name
309.     nome: "Fase di Suddivisione del Dominio in Aree Tematiche"
310.   milestones
311.     milestone1: DominioSuddiviso
312.   guards
313.     guard1: Guard1SuddivisioneDominio
314.   tasks
315.     task1: Task1SuddivisioneDominio;
316.     task2: Task2SuddivisioneDominio;
317.     task3: Task3SuddivisioneDominio
318. end
```

```
319. DominioSuddiviso in Milestone with
320.     name
321.         nome: "Dominio Suddiviso"
322.     validatingSentries
323.         val1: ValSentryDominioSuddiviso
324. end
```

```
325. CreazioneFunzionalità in AtomicStage with
326.   name
327.     nome: "Fase di Creazione delle Funzionalità"
328.   milestones
329.     milestone1: FunzionalitàCreate
330.   guards
331.     guard1: Guard1CreazioneFunzionalitr
332.   tasks
333.     task1: Task1CreazioneFunzionalità;
334.     task2: Task2CreazioneFunzionalità;
335.     task3: Task3CreazioneFunzionalità
336. end
```

```
337. FunzionalitàCreate in Milestone with
338.     name
339.         nome: "Funzionalità Create"
340.     validatingSentries
341.         val1: ValSentryFunzionalitàCreate
342. end
```

```
343. FormazioneTeamPianificazione in AtomicStage with
344.   name
345.     nome: "Fase di Formazione del Team di Pianificazione"
346.   milestones
347.     milestone1: TeamPianificazioneFormato
348.   guards
```

```
    guard1: Guard1FormazioneTeamPianificazione
  tasks
    task1: Task1FormazioneTeamPianificazione;
    task2: Task2FormazioneTeamPianificazione;
    task3: Task3FormazioneTeamPianificazione
end
```

```
TeamPianificazioneFormato in Milestone with
    name
        nome: "Team Pianificazione Formato"
    validatingSentries
        val1: ValSentryTeamPianificazioneFormato
end
```

```
PianificazioneAreeTematiche in AtomicStage with
  name
    nome: "Fase di Pianificazione delle Aree Tematiche"
  milestones
    milestone1: AreeTematichePianificate
  guards
    guard1: Guard1PianificazioneAreeTematiche
  tasks
    task1: Task1PianificazioneAreeTematiche
end
```

```
AreeTematichePianificate in Milestone with
    name
        nome: "Aree Tematiche Pianificate"
    validatingSentries
        val1: ValSentryAreeTematichePianificate
end
```

```
RealizzazioneSequenceDiagram in AtomicStage with
  name
    nome: "Fase di Realizzazione dei Sequence Diagram"
  milestones
    milestone1: SequenceDiagramRealizzati
  guards
    guard1: Guard1RealizzazioneSequenceDiagram;
      guard2: Guard2RealizzazioneSequenceDiagram
  tasks
    task1: Task1RealizzazioneSequenceDiagram;
    task2: Task2RealizzazioneSequenceDiagram;
    task3: Task3RealizzazioneSequenceDiagram
end
```

```
SequenceDiagramRealizzati in Milestone with
    name
        nome: "Sequence Diagram Realizzati"
    validatingSentries
        val1: ValSentrySequenceDiagramRealizzati
end
```

```
StesuraHeaderMetodi in AtomicStage with
  name
    nome: "Fase di Stesura degli Header dei Metodi"
```

```
399.   milestones
400.     milestone1: HeaderMetodiRealizzati
401.   guards
402.     guard1: Guard1StesuraHeaderMetodi;
403.       guard2: Guard2StesuraHeaderMetodi
404.   tasks
405.     task1: Task1StesuraHeaderMetodi;
406.     task2: Task2StesuraHeaderMetodi;
407.     task3: Task3StesuraHeaderMetodi
408. end
```

```
409. HeaderMetodiRealizzati in Milestone with
410.     name
411.         nome: "Header Metodi Realizzati"
412.     validatingSentries
413.         val1: ValSentryHeaderMetodiRealizzati
414. end
```

```
415. VerificaDesignPackage in AtomicStage with
416.   name
417.     nome: "Fase di Verifica del Design Package"
418.   milestones
419.     milestone1: DesignPackageApprovato;
420.     milestone2: DesignPackageDisapprovato
421.   guards
422.     guard1: Guard1VerificaDesignPackage
423.   tasks
424.     task1: Task1VerificaDesignPackage
425. end
```

```
426. DesignPackageApprovato in Milestone with
427.     name
428.         nome: "Design Package Approvato"
429.     validatingSentries
430.         val1: ValSentryDesignPackageApprovato
431. end
```

```
432. DesignPackageDisapprovato in Milestone with
433.     name
434.         nome: "Design Package Disapprovato"
435.     validatingSentries
436.         val1: ValSentryDesignPackageDisapprovato
437. end
```

```
438. Implementazione in AtomicStage with
439.   name
440.     nome: "Fase di Implementazione della Funzionalità"
441.   milestones
442.     milestone1: FunzionalitàImplementata
443.   guards
444.     guard1: Guard1Implementazione;
445.       guard2: Guard2Implementazione;
446.       guard3: Guard3Implementazione
447.   tasks
448.     task1: Task1Implementazione;
449.     task2: Task2Implementazione;
450.     task3: Task3Implementazione
```

```
451. end
```

```
452. FunzionalitàImplementata in Milestone with
453.     name
454.         nome: "Funzionalità Implementata"
455.     validatingSentries
456.         val1: ValSentryFunzionalitàImplementata
457. end
```

```
458. UnitTesting in AtomicStage with
459.   name
460.     nome: "Fase di Unit Testing"
461.   milestones
462.     milestone1: TestSuperati;
463.     milestone2: TestFalliti
464.   guards
465.     guard1: Guard1UnitTesting
466.   tasks
467.     task1: Task1UnitTesting
468. end
```

```
469. TestSuperati in Milestone with
470.     name
471.         nome: "Test Superati"
472.     validatingSentries
473.         val1: ValSentryTestSuperati
474. end
```

```
475. TestFalliti in Milestone with
476.     name
477.         nome: "Test Falliti"
478.     validatingSentries
479.         val1: ValSentryTestFalliti
480. end
```

```
481. Ispezione in AtomicStage with
482.   name
483.     nome: "Fase di Ispezione del Codice"
484.   milestones
485.     milestone1: IspezioneSuperata;
486.     milestone2: IspezioneFallita
487.   guards
488.     guard1: Guard1Ispezione
489.   tasks
490.     task1: Task1Ispezione
491. end
```

```
492. IspezioneSuperata in Milestone with
493.     name
494.         nome: "Ispezione Superata"
495.     validatingSentries
496.         val1: ValSentryIspezioneSuperata
497. end
```

```
498. IspezioneFallita in Milestone with
499.     name
```

```
500.         nome: "Ispezione Fallita"
501.     validatingSentries
502.         val1: ValSentryIspezioneFallita
503. end
```

```
504. EventoNuovoSoftware in ExternalEvent with
505.   name
506.     nome: "Inizia lo sviluppo di un nuovo software"
507.   entity
508.     progetto: Progetto
509. end
```

```
510. EventoTeamModellazioneFormato in ExternalEvent with
511.   name
512.     nome: "Il team di modellazione è stato formato"
513.   entity
514.     progetto: Progetto
515. end
```

```
516. EventoContestoAnalizzato in ExternalEvent with
517.   name
518.     nome: "Il contesto è stato analizzato"
519.   entity
520.     progetto: Progetto
521. end
```

```
522. EventoSottoModelliCreati in ExternalEvent with
523.   name
524.     nome: "I sotto modelli sono stati completati"
525.   entity
526.     progetto: Progetto
527. end
```

```
528. EventoModelloCreato in ExternalEvent with
529.   name
530.     nome: "Il modello del progetto è stato creato"
531.   entity
532.     progetto: Progetto
533. end
```

```
534. EventoModelloApprovato in ExternalEvent with
535.   name
536.     nome: "Il modello del progetto è approvato"
537.   entity
538.     progetto: Progetto
539. end
```

```
540. EventoModelloDisapprovato in ExternalEvent with
541.   name
542.     nome: "Il modello del progetto non è approvato"
543.   entity
544.     progetto: Progetto
545. end
```

```
546. EventoTeamFunzionalitàFormato in ExternalEvent with
```

```
547.   name
548.     nome: "Il team che stilerà la lista delle funzionalità è
   formato"
549.   entity
550.     progetto: Progetto
551. end
```

```
552. EventoDominioSuddiviso in ExternalEvent with
553.   name
554.     nome: "Il dominio è stato suddiviso in aree tematiche"
555.   entity
556.     progetto: Progetto
557. end
```

```
558. EventoFunzionalitàCreate in ExternalEvent with
559.   name
560.     nome: "Le funzionalità sono state create"
561.   entity
562.     progetto: Progetto
563. end
```

```
564. EventoStesuraListaTerminata in ExternalEvent with
565.   name
566.     nome: "La stesura della lista delle funzionalità è terminata"
567.   entity
568.     progetto: Progetto
569. end
```

```
570. EventoTeamPianificazioneFormato in ExternalEvent with
571.   name
572.     nome: "Il team di pianificazione è stato formato"
573.   entity
574.     progetto: Progetto
575. end
```

```
576. EventoAreeTematichePianificate in ExternalEvent with
577.   name
578.     nome: "Le aree tematiche sono state pianificate"
579.   entity
580.     progetto: Progetto
581. end
```

```
582. EventoSoftwareTerminato in ExternalEvent with
583.   name
584.     nome: "Il software è terminato"
585.   entity
586.     progetto: Progetto
587. end
```

```
588. EventoFunzioalitàPianificata in ExternalEvent with
589.   name
590.     nome: "La funzionalità è stata pianificata"
591.   entity
592.     funzionalità: Funzionalità
593. end
```

```
EventoSequenceDiagramRealizzati in ExternalEvent with
  name
    nome: "I sequence diagram della funzionalità sono stati
  realizzati"
  entity
    funzionalità: Funzionalità
end
```

```
EventoHeaderMetodiRealizzati in ExternalEvent with
  name
    nome: "Header metodi realizzati"
  entity
    funzionalità: Funzionalità
end
```

```
EventoDesignPackageApprovato in ExternalEvent with
  name
    nome: "Design package approvato"
  entity
    funzionalità: Funzionalità
end
```

```
EventoDesignPackageDisapprovato in ExternalEvent with
  name
    nome: "Design package disapprovato"
  entity
    funzionalità: Funzionalità
end
```

```
EventoFunzionalitàImplementata in ExternalEvent with
  name
    nome: "funzioanlità implementata"
  entity
    funzionalità: Funzionalità
end
```

```
EventoTestSuperati in ExternalEvent with
  name
    nome: "da scrivere"
  entity
    funzionalità: Funzionalità
end
```

```
EventoTestFalliti in ExternalEvent with
  name
    nome: "da scrivere"
  entity
    funzionalità: Funzionalità
end
```

```
EventoIspezioneSuperata in ExternalEvent with
  name
    nome: "da scrivere"
  entity
```

```
    funzionalità: Funzionalità
end
```

```
EventoIspezioneFallita in ExternalEvent with
  name
    nome: "da scrivere"
  entity
    funzionalità: Funzionalità
end
```

```
Task1FormazioneTeamModellazione in Task with
  type
    tipo: TaskTypeB
  description
    descrizione: "Viene formato un Team che si occupererà della
  fase di Modellazione"
end
```

```
Task2FormazioneTeamModellazione in Task with
  type
    tipo: TaskTypeE
  description
    descrizione: "Viene creata un'istanza di Team che rappresenta
  il Team di Modellazione"
end
```

```
Task3FormazioneTeamModellazione in Task with
  type
    tipo: TaskTypeA
  description
    descrizione: "Il Team viene assegnato al Progetto"
end
```

```
Task1AnalisiContesto in Task with
  type
    tipo: TaskTypeB
  description
    descrizione: "Il Team di Modellazione scrive un' Analisi del
  Contesto applicativo"
end
```

```
Task2AnalisiContesto in Task with
  type
    tipo: TaskTypeE
  description
    descrizione: "Viene creata un'istanza di Documento che contiene
  l'analisi del contesto"
end
```

```
Task3AnalisiContesto in Task with
  type
    tipo: TaskTypeA
  description
    descrizione: "Il Documento viene inserito nel Progetto"
end
```

```
684. Task1CostruzioneSottoModelli in Task with
685.   type
686.     tipo: TaskTypeB
687.   description
688.     descrizione: "Il Team di Modellazione divide il Dominio in
    sezioni e crea un Sottomodello per ognuna"
689. end
```

```
690. Task2CostruzioneSottoModelli in Task with
691.   type
692.     tipo: TaskTypeE
693.   description
694.     descrizione: "Per ogni Sottomodello viene creata un'istanza di
    Documento che lo contiene"
695. end
```

```
696. Task3CostruzioneSottoModelli in Task with
697.   type
698.     tipo: TaskTypeA
699.   description
700.     descrizione: "I Documenti vengono inseriti nel Progetto"
701. end
```

```
702. Task1CostruzioneModello in Task with
703.   type
704.     tipo: TaskTypeB
705.   description
706.     descrizione: "Il Team di Modellazione unisce ed elabora i
    Sottomodelli creando il Modello di Dominio completo"
707. end
```

```
708. Task2CostruzioneModello in Task with
709.   type
710.     tipo: TaskTypeE
711.   description
712.     descrizione: "Viene creata in'istanza di Documento che contiene
    il Modello di Dominio"
713. end
```

```
714. Task3CostruzioneModello in Task with
715.   type
716.     tipo: TaskTypeA
717.   description
718.     descrizione: "Il Documento viene inserito nel Progetto"
719. end
```

```
720. Task1VerificaModello in Task with
721.   type
722.     tipo: TaskTypeB
723.   description
724.     descrizione: "Il Team di Modellazione analizza il Modello di
    Dominio per verificarne la correttezza"
725. end
```

```
Task1FormazioneTeamFunzionalità in Task with
  type
    tipo: TaskTypeB
  description
    descrizione: "Viene formato un Team che si occupererà della
fase di Stesura della Lista delle Funzionalità"
end
```

```
Task2FormazioneTeamFunzionalità in Task with
  type
    tipo: TaskTypeE
  description
    descrizione: "Viene creata un'istanza di Team che rappresenta
il Team delle Funzionalità"
end
```

```
Task3FormazioneTeamFunzionalità in Task with
  type
    tipo: TaskTypeA
  description
    descrizione: "Il Team viene assegnato al Progetto"
end
```

```
Task1SuddivisioneDominio in Task with
  type
    tipo: TaskTypeB
  description
    descrizione: "Il Team delle Funzionalità divide il Dominio in
Aree Tematiche"
end
```

```
Task2SuddivisioneDominio in Task with
  type
    tipo: TaskTypeE
  description
    descrizione: "Per ogni Area Tematica viene creata un'istanza di
AreaTematica che la rappresenta"
end
```

```
Task3SuddivisioneDominio in Task with
  type
    tipo: TaskTypeA
  description
    descrizione: "Ogni AreaTematica viene inserita nel Progetto"
end
```

```
Task1FormazioneTeamPianificazione in Task with
  type
    tipo: TaskTypeB
  description
    descrizione: "Viene formato un Team che si occupererà della
fase di Pianificazione del Progetto"
end
```

```
Task2FormazioneTeamPianificazione in Task with
```

```
769.   type
770.     tipo: TaskTypeE
771.   description
772.     descrizione: "Viene creata un'istanza di Team che rappresenta
   il Team di Pianificazione"
773. end
```

```
774. Task3FormazioneTeamPianificazione in Task with
775.   type
776.     tipo: TaskTypeA
777.   description
778.     descrizione: "Il Team viene assegnato al Progetto"
779. end
```

```
780. Task1PianificazioneAreeTematiche in Task with
781.   type
782.     tipo: TaskTypeA
783.   description
784.     descrizione: "Il Team di Pianificazione assegna ad ogni Area
   Tematica uno sviluppatore capo e una scadenza"
785. end
```

```
786. Task1Sviluppo in Task with
787.   type
788.     tipo: TaskTypeB
789.   description
790.     descrizione: "Con una procedura incrementale, vengono unite le
   Funzionalità implementate a formare il software completo"
791. end
```

```
792. Task1CreazioneFunzionalità in Task with
793.   type
794.     tipo: TaskTypeB
795.   description
796.     descrizione: "Il Team delle Funzionalità scrive una lista di
   Funzionalità ognuna appartenente ad un'AreaTematica"
797. end
```

```
798. Task2CreazioneFunzionalità in Task with
799.   type
800.     tipo: TaskTypeE
801.   description
802.     descrizione: "Vengono create le istanze di Funzionalità"
803. end
```

```
804. Task3CreazioneFunzionalità in Task with
805.   type
806.     tipo: TaskTypeA
807.   description
808.     descrizione: "Ogni Funzionalità viene inserita nel Progetto"
809. end
```

```
810. Task1PianificazioneFunzionalità in Task with
811.   type
812.     tipo: TaskTypeA
813.   description
```

```
814.     descrizione: "Alla Funzionalità viene assegnato uno
   sviluppatore ed una scadenza"
815. end
```

```
816. Task1RealizzazioneSequenceDiagram in Task with
817.   type
818.     tipo: TaskTypeB
819.   description
820.     descrizione: "Vengono realizzati i Sequence Diagram della
   Funzionalità"
821. end
```

```
822. Task2RealizzazioneSequenceDiagram in Task with
823.   type
824.     tipo: TaskTypeE
825.   description
826.     descrizione: "Vengono create delle istanze di Documento che
   contengono i Sequence Diagram"
827. end
```

```
828. Task3RealizzazioneSequenceDiagram in Task with
829.   type
830.     tipo: TaskTypeA
831.   description
832.     descrizione: "I Documenti vengono inseriti nel Design Package
   della Funzionalità"
833. end
```

```
834. Task1StesuraHeaderMetodi in Task with
835.   type
836.     tipo: TaskTypeB
837.   description
838.     descrizione: "Vengono scritti gli header dei Metodi della
   Funzionalità"
839. end
```

```
840. Task2StesuraHeaderMetodi in Task with
841.   type
842.     tipo: TaskTypeE
843.   description
844.     descrizione: "Vengono create delle istanze di Documento che
   contengono gli header dei Metodi"
845. end
```

```
846. Task3StesuraHeaderMetodi in Task with
847.   type
848.     tipo: TaskTypeA
849.   description
850.     descrizione: "I Documenti vengono inseriti nel Design Package
   della Funzionalità"
851. end
```

```
852. Task1VerificaDesignPackage in Task with
853.   type
854.     tipo: TaskTypeB
855.   description
```

```
856.     descrizione: "Viene verificata la correttezza del Design
   Package della Funzionalità "
857. end
```

```
858. Task1Implementazione in Task with
859.   type
860.     tipo: TaskTypeB
861.   description
862.     descrizione: "La Funzionalità viene implementata dallo
   sviluppatore a cui è stata assegnata"
863. end
```

```
864. Task2Implementazione in Task with
865.   type
866.     tipo: TaskTypeE
867.   description
868.     descrizione: "Viene creata un'istanza di Documento che contiene
   l'implementazione della Funzionalità"
869. end
```

```
870. Task3Implementazione in Task with
871.   type
872.     tipo: TaskTypeA
873.   description
874.     descrizione: "Il Documento viene inserito nella Funzionalità"
875. end
```

```
876. Task1UnitTesting in Task with
877.   type
878.     tipo: TaskTypeB
879.   description
880.     descrizione: "Vengono eseguiti degli Unit Test sulla
   Funzionalità"
881. end
```

```
882. Task1Ispezione in Task with
883.   type
884.     tipo: TaskTypeB
885.   description
886.     descrizione: "Il codice della Funzionalità viene sottoposto ad
   un'ispezione"
887. end
```

Query

```
888. ProgettiConSottoModelli in QueryClass isA Progetto with
889.   constraint
890.     c1: $ exists doc/Documento (this sottoModelli doc) $
891. end
```

```
892. ProgettiConAreeTematiche in QueryClass isA Progetto with
893.   constraint
894.     c1: $ exists a/AreaTematica (this areeTematiche a) $
895. end
```

```
ProgettiConFunzionalità in QueryClass isA Progetto with
  constraint
    c1: $ exists f/Funzionalità (this funzionalità f) $
end
```

```
AreeTematicheGiàPianificate in QueryClass isA AreaTematica with
  constraint
    c1: $ exists pers/Persona data/Date (this sviluppatoreCapo
  pers) and (this dataTermine data) $
end
```

```
ProgettiConAreePianificate in QueryClass isA Progetto with
  constraint
    c1: $ forall a/AreaTematica (this areeTematiche a) ==> (a in
  AreeTematicheGiàPianificate) $
end
```

```
FunzionalitàGiàPianificate in QueryClass isA Funzionalità with
  constraint
    c1: $ exists m/FunzionalitàPianificata
  s/PianificazioneFunzionalità (s parentEntity this) and (s
  [milestones] m) and (m status TRUE)$
end
```

```
ProgettiConFunzionalitàPianificate in QueryClass isA Progetto with
  constraint
    c1: $ forall f/Funzionalità (this funzionalità f) ==> (f in
  FunzionalitàGiàPianificate) $
end
```

```
FunzionalitàGiàRealizzate in QueryClass isA Funzionalità with
  constraint
    c1: $ exists m/FunzionalitàRealizzata s/Realizzazione (s
  parentEntity this) and (s [milestones] m) and (m status TRUE) $
end
```

```
ProgettiConFunzionalitàRealizzate in QueryClass isA Progetto with
  constraint
    c1: $ forall f/Funzionalità (this funzionalità f) ==> (f in
  FunzionalitàGiàRealizzate) $
end
```

```
DesignPackageConSequenceDiagram in QueryClass isA DesignPackage
  with
  constraint
    c1: $ exists doc/Documento (this sequenceDiagrams doc) $
end
```

```
DesignPackageConHeaderMetodi in QueryClass isA DesignPackage with
  constraint
    c1: $ exists doc/Documento (this headerMetodi doc) $
end
```

Guards

```
Guard1Modellazione in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/ExternalEvent p/Progetto s/StageType
m/MilestoneType d/Individual
             ON Tell (ev in EventoNuovoSoftware)
             IFNEW (p in Progetto) and (ev progetto p)
                   and (s in Modellazione) and (s parentEntity p)
                   and (m in ModellazioneTerminata)
                   and (s [milestones] m) and (m status FALSE)
             DO Retell (s status TRUE),
                CALL CreateIndividual (StOpEvent, d),
                Tell (d in StageOpenedEvent),
                Tell (d stage/st s)
             ELSE reject
          $
end
```

```
Guard1FormazioneTeamModellazione in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
             ON Tell (ev in StageOpenedEvent)
             IFNEW (s in Modellazione) and (ev stage s)
                   and (sb in FormazioneTeamModellazione)
                   and (s [body] sb)
             DO Retell (sb status TRUE),
                CALL CreateIndividual (StOpEvent, d),
                Tell (d in StageOpenedEvent),
                Tell (d stage/st sb)
          $
end
```

```
Guard2FormazioneTeamModellazione in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/ExternalEvent s,sm/StageType m/MilestoneType
d/Individual
             ON Tell (ev in ModificaTeamModellazione)
             IFNEW (p in Progetto) and (ev progetto p)
                   and (s in FormazioneTeamModellazione)
                   and (s parentEntity p)
                   and (s status FALSE)
                   and (sm in Modellazione) and (sm parentEntity p)
                   and (m in ModellazioneTerminata)
                   and (sm [milestones] m) and (m status FALSE)
             DO Retell (s status TRUE),
                CALL CreateIndividual (StOpEvent, d),
                Tell (d in StageOpenedEvent),
                Tell (d stage/st s)
          $
end
```

```
Guard1AnalisiContesto in ECArule with
```

```
981.    mode m: Deferred
982.    ecarule
983.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
984.              ON Tell (ev in MilestoneAchivedEvent)
985.              IFNEW (m in TeamModellazioneFormato)
986.                    and (ev milestone m)
987.                    and (s1 in FormazioneTeamModellazione)
988.                    and (s1 [milestones] m)
989.                    and (p in Progetto) and (s1 parentEntity p)
990.                    and (s2 in AnalisiContesto)
991.                    and (s2 parentEntity p)
992.              DO Retell (s2 status TRUE),
993.                 CALL CreateIndividual (StOpEvent, d),
994.                 Tell (d in StageOpenedEvent),
995.                 Tell (d stage/st s2)
996.           $
997. end
```

```
998. Guard1InvAnalisiContesto in ECArule with
999.    mode m: Deferred
1000.    ecarule
1001.     myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1002.              ON Tell (ev in MilestoneInvalidatedEvent)
1003.              IFNEW (m in TeamModellazioneFormato)
1004.                    and (ev milestone m)
1005.                    and (s1 in FormazioneTeamModellazione)
1006.                    and (s1 [milestones] m)
1007.                    and (p in Progetto) and (s1 parentEntity p)
1008.                    and (s2 in AnalisiContesto)
1009.                    and (s2 parentEntity p)
1010.                    and (s2 status TRUE)
1011.              DO Retell (s2 status FALSE),
1012.                 CALL CreateIndividual (StClEvent, d),
1013.                 Tell (d in StageClosedEvent),
1014.                 Tell (d stage/st s2)
1015.           $
1016. end
```

```
1017. Guard1CostruzioneSottoModelli in ECArule with
1018.    mode m: Deferred
1019.    ecarule
1020.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1021.              ON Tell (ev in MilestoneAchivedEvent)
1022.              IFNEW (m in ContestoAnalizzato) and (ev milestone m)
1023.                    and (s1 in AnalisiContesto)
1024.                    and (s1 [milestones] m)
1025.                    and (p in Progetto) and (s1 parentEntity p)
1026.                    and (s2 in CostruzioneSottoModelli) and (s2
   parentEntity p)
1027.              DO Retell (s2 status TRUE),
1028.                 CALL CreateIndividual (StOpEvent, d),
1029.                 Tell (d in StageOpenedEvent),
1030.                 Tell (d stage/st s2)
1031.           $
1032. end
```

```
1033. Guard2CostruzioneSottoModelli in ECArule with
1034.    mode m: Deferred
1035.    ecarule
1036.     myecarule : $ ev/MilestoneAchivedEvent m,m2,m3/MilestoneType
   s1,s2,s3/StageType p/Progetto d/Individual
1037.              ON Tell (ev in MilestoneAchivedEvent)
1038.              IFNEW (m in TeamModellazioneFormato)
1039.                    and (ev milestone m)
1040.                    and (s1 in FormazioneTeamModellazione)
1041.                    and (s1 [milestones] m)
1042.                    and (p in Progetto) and (s1 parentEntity p)
1043.                    and (s2 in CostruzioneSottoModelli)
1044.                    and (s2 parentEntity p)
1045.                    and (m2 in SottoModelliCreati)
1046.                    and (s2 [milestones] m2)
1047.                    and (m2 status FALSE)
1048.                    and (s3 in AnalisiContesto)
1049.                    and (s3 parentEntity p)
1050.                    and (m3 in ContestoAnalizzato)
1051.                    and (s3 [milestones] m3)
1052.                    and (m3 status TRUE)
1053.              DO Retell (s2 status TRUE),
1054.                 CALL CreateIndividual (StOpEvent, d),
1055.                 Tell (d in StageOpenedEvent),
1056.                 Tell (d stage/st s2)
1057.           $
1058. end
```

```
1059. Guard1InvCostruzioneSottoModelli in ECArule with
1060.    mode m: Deferred
1061.    ecarule
1062.     myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1063.              ON Tell (ev in MilestoneInvalidatedEvent)
1064.              IFNEW (m in TeamModellazioneFormato)
1065.                    and (ev milestone m)
1066.                    and (s1 in FormazioneTeamModellazione)
1067.                    and (s1 [milestones] m)
1068.                    and (p in Progetto) and (s1 parentEntity p)
1069.                    and (s2 in CostruzioneSottoModelli)
1070.                    and (s2 parentEntity p)
1071.                    and (s2 status TRUE)
1072.              DO Retell (s2 status FALSE),
1073.                 CALL CreateIndividual (StClEvent, d),
1074.                 Tell (d in StageClosedEvent),
1075.                 Tell (d stage/st s2)
1076.           $
1077. end
```

```
1078. Guard1CostruzioneModello in ECArule with
1079.    mode m: Deferred
1080.    ecarule
1081.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1082.              ON Tell (ev in MilestoneAchivedEvent)
1083.              IFNEW (m in SottoModelliCreati) and (ev milestone m)
1084.                    and (s1 in CostruzioneSottoModelli)
```

```
1085.                   and (s1 [milestones] m)
1086.                   and (p in Progetto) and (s1 parentEntity p)
1087.                   and (s2 in CostruzioneModello)
1088.                   and (s2 parentEntity p)
1089.             DO Retell (s2 status TRUE),
1090.                CALL CreateIndividual (StOpEvent, d),
1091.                Tell (d in StageOpenedEvent),
1092.                Tell (d stage/st s2)
1093.          $
1094. end
```

```
1095. Guard2CostruzioneModello in ECArule with
1096.    mode m: Deferred
1097.    ecarule
1098.     myecarule : $ ev/MilestoneAchivedEvent m,m2,m3/MilestoneType
   s1,s2,s3/StageType p/Progetto d/Individual
1099.             ON Tell (ev in MilestoneAchivedEvent)
1100.             IFNEW (m in TeamModellazioneFormato)
1101.                   and (ev milestone m)
1102.                   and (s1 in FormazioneTeamModellazione)
1103.                   and (s1 [milestones] m)
1104.                   and (p in Progetto) and (s1 parentEntity p)
1105.                   and (s2 in CostruzioneModello)
1106.                   and (s2 parentEntity p)
1107.                   and (m2 in ModelloCreato)
1108.                   and (s2 [milestones] m2)
1109.                   and (m2 status FALSE)
1110.                   and (s3 in CostruzioneSottoModelli)
1111.                   and (s3 parentEntity p)
1112.                   and (m3 in SottoModelliCreati)
1113.                   and (s3 [milestones] m3)
1114.                   and (m3 status TRUE)
1115.             DO Retell (s2 status TRUE),
1116.                CALL CreateIndividual (StOpEvent, d),
1117.                Tell (d in StageOpenedEvent),
1118.                Tell (d stage/st s2)
1119.          $
1120. end
```

```
1121. Guard1InvCostruzioneModello in ECArule with
1122.    mode m: Deferred
1123.    ecarule
1124.     myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1125.             ON Tell (ev in MilestoneInvalidatedEvent)
1126.             IFNEW (m in TeamModellazioneFormato)
1127.                   and (ev milestone m)
1128.                   and(s1 in FormazioneTeamModellazione)
1129.                   and (s1 [milestones] m)
1130.                   and (p in Progetto) and (s1 parentEntity p)
1131.                   and (s2 in CostruzioneModello)
1132.                   and (s2 parentEntity p)
1133.                   and (s2 status TRUE)
1134.             DO Retell (s2 status FALSE),
1135.                CALL CreateIndividual (StClEvent, d),
1136.                Tell (d in StageClosedEvent),
1137.                Tell (d stage/st s2)
1138.          $
```

```
end
```

```
Guard3CostruzioneModello in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
             ON Tell (ev in MilestoneAchivedEvent)
             IFNEW (m in ModelloDisapprovato) and (ev milestone m)
                   and (s1 in VerificaModello)
                   and (s1 [milestones] m)
                   and (p in Progetto) and (s1 parentEntity p)
                   and (s2 in CostruzioneModello)
                   and (s2 parentEntity p)
             DO Retell (s2 status TRUE),
                CALL CreateIndividual (StOpEvent, d),
                Tell (d in StageOpenedEvent),
                Tell (d stage/st s2)
          $
end
```

```
Guard1VerificaModello in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
             ON Tell (ev in MilestoneAchivedEvent)
             IFNEW (m in ModelloCreato) and (ev milestone m)
                   and (s1 in CostruzioneModello)
                   and (s1 [milestones] m)
                   and (p in Progetto) and (s1 parentEntity p)
                   and (s2 in VerificaModello)
                   and (s2 parentEntity p)
             DO Retell (s2 status TRUE),
                CALL CreateIndividual (StOpEvent, d),
                Tell (d in StageOpenedEvent),
                Tell (d stage/st s2)
          $
end
```

```
Guard1InvVerificaModello in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
             ON Tell (ev in MilestoneInvalidatedEvent)
             IFNEW (m in TeamModellazioneFormato)
                   and (ev milestone m)
                   and (s1 in FormazioneTeamModellazione)
                   and (s1 [milestones] m)
                   and (p in Progetto) and (s1 parentEntity p)
                   and (s2 in VerificaModello)
                   and (s2 parentEntity p)
                   and (s2 status TRUE)
             DO Retell (s2 status FALSE),
                CALL CreateIndividual (StClEvent, d),
                Tell (d in StageClosedEvent),
```

```
1190.                Tell (d stage/st s2)
1191.           $
1192. end
```

```
1193. Guard2VerificaModello in ECArule with
1194.    mode m: Deferred
1195.    ecarule
1196.     myecarule : $ ev/MilestoneAchivedEvent
   m,m2,m3,m4/MilestoneType s1,s2,s3/StageType p/Progetto d/Individual
1197.              ON Tell (ev in MilestoneAchivedEvent)
1198.              IFNEW (m in TeamModellazioneFormato)
1199.                    and (ev milestone m)
1200.                    and (s1 in FormazioneTeamModellazione)
1201.                    and (s1 [milestones] m)
1202.                    and (p in Progetto) and (s1 parentEntity p)
1203.                    and (s2 in VerificaModello)
1204.                    and (s2 parentEntity p)
1205.                    and (m2 in ModelloApprovato)
1206.                    and (s2 [milestones] m2)
1207.                    and (m2 status FALSE)
1208.                    and (m3 in ModelloDisapprovato)
1209.                    and (s2 [milestones] m3)
1210.                    and (m3 status FALSE)
1211.                    and (m4 in ModelloCreato)
1212.                    and (s3 in CostruzioneModello)
1213.                    and (s3 [milestones] m4)
1214.                    and (m4 status TRUE)
1215.              DO Retell (s2 status TRUE),
1216.                 CALL CreateIndividual (StOpEvent, d),
1217.                 Tell (d in StageOpenedEvent),
1218.                 Tell (d stage/st s2)
1219.           $
1220. end
```

```
1221. Guard1StesuraListaFunzionalità in ECArule with
1222.    mode m: Deferred
1223.    ecarule
1224.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1225.              ON Tell (ev in MilestoneAchivedEvent)
1226.              IFNEW (m in ModellazioneTerminata)
1227.                    and (ev milestone m)
1228.                    and (s1 in Modellazione)
1229.                    and (s1 [milestones] m)
1230.                    and (p in Progetto) and (s1 parentEntity p)
1231.                    and (s2 in StesuraListaFunzionalità)
1232.                    and (s2 parentEntity p)
1233.              DO Retell (s2 status TRUE),
1234.                 CALL CreateIndividual (StOpEvent, d),
1235.                 Tell (d in StageOpenedEvent),
1236.                 Tell (d stage/st s2)
1237.           $
1238. end
```

```
1239. Guard1FormazioneTeamFunzionalità in ECArule with
1240.    mode m: Deferred
1241.    ecarule
```

```
1242.     myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
1243.               ON Tell (ev in StageOpenedEvent)
1244.               IFNEW (s in StesuraListaFunzionalità)
1245.                     and (ev stage s)
1246.                     and (sb in FormazioneTeamFunzionalità)
1247.                     and (s [body] sb)
1248.               DO Retell (sb status TRUE),
1249.                  CALL CreateIndividual (StOpEvent, d),
1250.                  Tell (d in StageOpenedEvent),
1251.                  Tell (d stage/st sb)
1252.            $
1253. end
```

```
1254. Guard1SuddivisioneDominio in ECArule with
1255.    mode m: Deferred
1256.    ecarule
1257.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1258.               ON Tell (ev in MilestoneAchivedEvent)
1259.               IFNEW (m in TeamFunzionalitàFormato)
1260.                     and (ev milestone m)
1261.                     and (s1 in FormazioneTeamFunzionalità)
1262.                     and (s1 [milestones] m)
1263.                     and (p in Progetto) and (s1 parentEntity p)
1264.                     and (s2 in SuddivisioneDominio)
1265.                     and (s2 parentEntity p)
1266.               DO Retell (s2 status TRUE),
1267.                  CALL CreateIndividual (StOpEvent, d),
1268.                  Tell (d in StageOpenedEvent),
1269.                  Tell (d stage/st s2)
1270.            $
1271. end
```

```
1272. Guard1CreazioneFunzionalità in ECArule with
1273.    mode m: Deferred
1274.    ecarule
1275.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1276.               ON Tell (ev in MilestoneAchivedEvent)
1277.               IFNEW (m in DominioSuddiviso) and (ev milestone m)
1278.                     and (s1 in SuddivisioneDominio)
1279.                     and (s1 [milestones] m)
1280.                     and (p in Progetto) and (s1 parentEntity p)
1281.                     and (s2 in CreazioneFunzionalità)
1282.                     and (s2 parentEntity p)
1283.               DO Retell (s2 status TRUE),
1284.                  CALL CreateIndividual (StOpEvent, d),
1285.                  Tell (d in StageOpenedEvent),
1286.                  Tell (d stage/st s2)
1287.            $
1288. end
```

```
1289. Guard1Pianificazione in ECArule with
1290.    mode m: Deferred
1291.    ecarule
1292.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
```

```
1293.              ON Tell (ev in MilestoneAchivedEvent)
1294.              IFNEW (m in StesuraListaTerminata)
1295.                    and (ev milestone m)
1296.                    and (s1 in StesuraListaFunzionalità)
1297.                    and (s1 [milestones] m)
1298.                    and (p in Progetto) and (s1 parentEntity p)
1299.                    and (s2 in Pianificazione)
1300.                    and (s2 parentEntity p)
1301.              DO Retell (s2 status TRUE),
1302.                 CALL CreateIndividual (StOpEvent, d),
1303.                 Tell (d in StageOpenedEvent),
1304.                 Tell (d stage/st s2)
1305.           $
1306. end
```

```
1307. Guard1FormazioneTeamPianificazione in ECArule with
1308.    mode m: Deferred
1309.    ecarule
1310.     myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
1311.              ON Tell (ev in StageOpenedEvent)
1312.              IFNEW (s in Pianificazione) and (ev stage s)
1313.                    and (sb in FormazioneTeamPianificazione)
1314.                    and (s [body] sb)
1315.              DO Retell (sb status TRUE),
1316.                 CALL CreateIndividual (StOpEvent, d),
1317.                 Tell (d in StageOpenedEvent),
1318.                 Tell (d stage/st sb)
1319.           $
1320. end
```

```
1321. Guard1PianificazioneAreeTematiche in ECArule with
1322.    mode m: Deferred
1323.    ecarule
1324.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType p/Progetto d/Individual
1325.              ON Tell (ev in MilestoneAchivedEvent)
1326.              IFNEW (m in TeamPianificazioneFormato)
1327.                    and (ev milestone m)
1328.                    and (s1 in FormazioneTeamPianificazione)
1329.                    and (s1 [milestones] m)
1330.                    and (p in Progetto) and (s1 parentEntity p)
1331.                    and (s2 in PianificazioneAreeTematiche)
1332.                    and (s2 parentEntity p)
1333.              DO Retell (s2 status TRUE),
1334.                 CALL CreateIndividual (StOpEvent, d),
1335.                 Tell (d in StageOpenedEvent),
1336.                 Tell (d stage/st s2)
1337.           $
1338. end
```

```
1339. Guard1Sviluppo in ECArule with
1340.    mode m: Deferred
1341.    ecarule
1342.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità p/Progetto d/Individual
1343.              ON Tell (ev in MilestoneAchivedEvent)
1344.              IFNEW (m in FunzionalitàRealizzata)
```

```
                    and (ev milestone m)
                    and (s1 in Realizzazione)
                    and (s1 [milestones] m)
                    and (f in Funzionalità) and (s1 parentEntity f)
                    and (p in Progetto) and (p funzionalità f)
                    and (s2 in Sviluppo) and (s2 parentEntity p)
                    and (s2 status FALSE)
              DO Retell (s2 status TRUE),
                 CALL CreateIndividual (StOpEvent, d),
                 Tell (d in StageOpenedEvent),
                 Tell (d stage/st s2)
           $
end
```

```
Guard1PianificazioneFunzionalità in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
  s1,s2/StageType f/Funzionalità p/Progetto d/Individual
              ON Tell (ev in MilestoneAchivedEvent)
              IFNEW (m in AreeTematichePianificate)
                    and (ev milestone m)
                    and (s1 in PianificazioneAreeTematiche)
                    and (s1 [milestones] m)
                    and (p in Progetto) and (s1 parentEntity p)
                    and (f in Funzionalità) and (p funzionalità f)
                    and (s2 in PianificazioneFunzionalità)
                    and (s2 parentEntity f)
              DO Retell (s2 status TRUE),
                 CALL CreateIndividual (StOpEvent, d),
                 Tell (d in StageOpenedEvent),
                 Tell (d stage/st s2)
           $
end
```

```
Guard1Progettazione in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
  s1,s2/StageType f/Funzionalità p/Progetto d/Individual
              ON Tell (ev in MilestoneAchivedEvent)
              IFNEW (m in PianificazioneTerminata)
                   and (ev milestone m)
                    and (s1 in Pianificazione)
                    and (s1 [milestones] m)
                    and (p in Progetto) and (s1 parentEntity p)
                    and (f in Funzionalità) and (p funzionalità f)
                    and (s2 in Progettazione)
                    and (s2 parentEntity f)
              DO Retell (s2 status TRUE),
                 CALL CreateIndividual (StOpEvent, d),
                 Tell (d in StageOpenedEvent),
                 Tell (d stage/st s2)
           $
end
```

```
Guard1RealizzazioneSequenceDiagram in ECArule with
```

```
1397.    mode m: Deferred
1398.    ecarule
1399.     myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
1400.              ON Tell (ev in StageOpenedEvent)
1401.              IFNEW (s in Progettazione) and (ev stage s)
1402.                    and (sb in RealizzazioneSequenceDiagram)
1403.                    and (s [body] sb)
1404.              DO Retell (sb status TRUE),
1405.                 CALL CreateIndividual (StOpEvent, d),
1406.                 Tell (d in StageOpenedEvent),
1407.                 Tell (d stage/st sb)
1408.           $
1409. end
```

```
1410. Guard2RealizzazioneSequenceDiagram in ECArule with
1411.    mode m: Deferred
1412.    ecarule
1413.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1414.              ON Tell (ev in MilestoneAchivedEvent)
1415.              IFNEW (m in DesignPackageDisapprovato)
1416.                    and (ev milestone m)
1417.                    and (s1 in VerificaDesignPackage)
1418.                    and (s1 [milestones] m)
1419.                    and (f in Funzionalità) and (s1 parentEntity f)
1420.                    and (s2 in RealizzazioneSequenceDiagram)
1421.                    and (s2 parentEntity f)
1422.              DO Retell (s2 status TRUE),
1423.                 CALL CreateIndividual (StOpEvent, d),
1424.                 Tell (d in StageOpenedEvent),
1425.                 Tell (d stage/st s2)
1426.           $
1427. end
```

```
1428. Guard1StesuraHeaderMetodi in ECArule with
1429.    mode m: Deferred
1430.    ecarule
1431.     myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
1432.              ON Tell (ev in StageOpenedEvent)
1433.              IFNEW (s in Progettazione) and (ev stage s)
1434.                    and (sb in StesuraHeaderMetodi)
1435.                    and (s [body] sb)
1436.              DO Retell (sb status TRUE),
1437.                 CALL CreateIndividual (StOpEvent, d),
1438.                 Tell (d in StageOpenedEvent),
1439.                 Tell (d stage/st sb)
1440.           $
1441. end
```

```
1442. Guard2StesuraHeaderMetodi in ECArule with
1443.    mode m: Deferred
1444.    ecarule
1445.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1446.              ON Tell (ev in MilestoneAchivedEvent)
1447.              IFNEW (m in DesignPackageDisapprovato)
1448.                    and (ev milestone m)
```

```
1449.                   and (s1 in VerificaDesignPackage)
1450.                   and (s1 [milestones] m)
1451.                   and (f in Funzionalità) and (s1 parentEntity f)
1452.                   and (s2 in StesuraHeaderMetodi)
1453.                   and (s2 parentEntity f)
1454.             DO Retell (s2 status TRUE),
1455.                CALL CreateIndividual (StOpEvent, d),
1456.                Tell (d in StageOpenedEvent),
1457.                Tell (d stage/st s2)
1458.          $
1459. end
```

```
1460. Guard1VerificaDesignPackage in ECArule with
1461.    mode m: Deferred
1462.    ecarule
1463.     myecarule : $ ev/MilestoneAchivedEvent m,mr,ms/MilestoneType
   s,sr,ss,s2/StageType f/Funzionalità d/Individual
1464.             ON Tell (ev in MilestoneAchivedEvent)
1465.             IFNEW (ev milestone m) and (s in StageType)
1466.                   and (s [milestones] m) and (f in Funzionalità)
1467.                   and (s parentEntity f)
1468.                   and (sr in RealizzazioneSequenceDiagram)
1469.                   and (sr parentEntity f)
1470.                   and (mr in SequenceDiagramRealizzati)
1471.                   and (sr [milestones] mr) and (mr status TRUE)
1472.                   and (ss in StesuraHeaderMetodi)
1473.                   and (ss parentEntity f)
1474.                   and (ms in HeaderMetodiRealizzati)
1475.                   and (ss [milestones] ms) and (ms status TRUE)
1476.                   and (s2 in VerificaDesignPackage)
1477.                   and (s2 parentEntity f)
1478.                   and ((m in SequenceDiagramRealizzati)
1479.                         or (m in HeaderMetodiRealizzati))
1480.             DO Retell (s2 status TRUE),
1481.                CALL CreateIndividual (StOpEvent, d),
1482.                Tell (d in StageOpenedEvent),
1483.                Tell (d stage/st s2)
1484.          $
1485. end
```

```
1486. Guard1Realizzazione in ECArule with
1487.    mode m: Deferred
1488.    ecarule
1489.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1490.             ON Tell (ev in MilestoneAchivedEvent)
1491.             IFNEW (m in FunzionalitàProgettata)
1492.                   and (ev milestone m)
1493.                   and (s1 in Progettazione)
1494.                   and (s1 [milestones] m)
1495.                   and (f in Funzionalità) and (s1 parentEntity f)
1496.                   and (s2 in Realizzazione)
1497.                   and (s2 parentEntity f)
1498.             DO Retell (s2 status TRUE),
1499.                CALL CreateIndividual (StOpEvent, d),
1500.                Tell (d in StageOpenedEvent),
1501.                Tell (d stage/st s2)
1502.          $
```

```
1503. end
```

```
1504. Guard1Implementazione in ECArule with
1505.    mode m: Deferred
1506.    ecarule
1507.     myecarule : $ ev/StageOpenedEvent s,sb/StageType d/Individual
1508.              ON Tell (ev in StageOpenedEvent)
1509.              IFNEW (s in Realizzazione) and (ev stage s)
1510.                    and (sb in Implementazione) and (s [body] sb)
1511.              DO Retell (sb status TRUE),
1512.                 CALL CreateIndividual (StOpEvent, d),
1513.                 Tell (d in StageOpenedEvent),
1514.                 Tell (d stage/st sb)
1515.           $
1516. end
```

```
1517. Guard2Implementazione in ECArule with
1518.    mode m: Deferred
1519.    ecarule
1520.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1521.              ON Tell (ev in MilestoneAchivedEvent)
1522.              IFNEW (m in TestFalliti) and (ev milestone m)
1523.                    and (s1 in UnitTesting) and (s1 [milestones] m)
1524.                    and (f in Funzionalità) and (s1 parentEntity f)
1525.                    and (s2 in Implementazione)
1526.                    and (s2 parentEntity f)
1527.              DO Retell (s2 status TRUE),
1528.                 CALL CreateIndividual (StOpEvent, d),
1529.                 Tell (d in StageOpenedEvent),
1530.                 Tell (d stage/st s2)
1531.           $
1532. end
```

```
1533. Guard3Implementazione in ECArule with
1534.    mode m: Deferred
1535.    ecarule
1536.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1537.              ON Tell (ev in MilestoneAchivedEvent)
1538.              IFNEW (m in IspezioneFallita) and (ev milestone m)
1539.                    and (s1 in Ispezione) and (s1 [milestones] m)
1540.                    and (f in Funzionalità) and (s1 parentEntity f)
1541.                    and (s2 in Implementazione)
1542.                    and (s2 parentEntity f)
1543.              DO Retell (s2 status TRUE),
1544.                 CALL CreateIndividual (StOpEvent, d),
1545.                 Tell (d in StageOpenedEvent),
1546.                 Tell (d stage/st s2)
1547.           $
1548. end
```

```
1549. Guard1UnitTesting in ECArule with
1550.    mode m: Deferred
1551.    ecarule
1552.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
```

```
1553.              ON Tell (ev in MilestoneAchivedEvent)
1554.              IFNEW (m in FunzionalitàImplementata)
1555.                    and (ev milestone m)
1556.                    and (s1 in Implementazione)
1557.                    and (s1 [milestones] m)
1558.                    and (f in Funzionalità) and (s1 parentEntity f)
1559.                    and (s2 in UnitTesting) and (s2 parentEntity f)
1560.              DO Retell (s2 status TRUE),
1561.                 CALL CreateIndividual (StOpEvent, d),
1562.                 Tell (d in StageOpenedEvent),
1563.                 Tell (d stage/st s2)
1564.           $
1565. end
```

```
1566. GuardInvUnitTesting in ECArule with
1567.    mode m: Deferred
1568.    ecarule
1569.     myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1570.              ON Tell (ev in MilestoneInvalidatedEvent)
1571.              IFNEW (m in FunzionalitàImplementata)
1572.                    and (ev milestone m)
1573.                    and (s1 in Implementazione)
1574.                    and (s1 [milestones] m)
1575.                    and (f in Funzionalità) and (s1 parentEntity f)
1576.                    and (s2 in UnitTesting) and (s2 parentEntity f)
1577.                    and (s2 status TRUE)
1578.              DO Retell (s2 status FALSE),
1579.                 CALL CreateIndividual (StClEvent, d),
1580.                 Tell (d in StageClosedEvent),
1581.                 Tell (d stage/st s2)
1582.           $
1583. end
```

```
1584. Guard1Ispezione in ECArule with
1585.    mode m: Deferred
1586.    ecarule
1587.     myecarule : $ ev/MilestoneAchivedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1588.              ON Tell (ev in MilestoneAchivedEvent)
1589.              IFNEW (m in FunzionalitàImplementata)
1590.                    and (ev milestone m)
1591.                    and (s1 in Implementazione)
1592.                    and (s1 [milestones] m)
1593.                    and (f in Funzionalità) and (s1 parentEntity f)
1594.                    and (s2 in Ispezione) and (s2 parentEntity f)
1595.              DO Retell (s2 status TRUE),
1596.                 CALL CreateIndividual (StOpEvent, d),
1597.                 Tell (d in StageOpenedEvent),
1598.                 Tell (d stage/st s2)
1599.           $
1600. end
```

```
1601. GuardInvIspezione in ECArule with
1602.    mode m: Deferred
1603.    ecarule
```

```
1604.     myecarule : $ ev/MilestoneInvalidatedEvent m/MilestoneType
   s1,s2/StageType f/Funzionalità d/Individual
1605.              ON Tell (ev in MilestoneInvalidatedEvent)
1606.              IFNEW (m in FunzionalitàImplementata)
1607.                    and (ev milestone m)
1608.                    and (s1 in Implementazione)
1609.                    and (s1 [milestones] m)
1610.                    and (f in Funzionalità) and (s1 parentEntity f)
1611.                    and (s2 in Ispezione) and (s2 parentEntity f)
1612.                    and (s2 status TRUE)
1613.              DO Retell (s2 status FALSE),
1614.                 CALL CreateIndividual (StClEvent, d),
1615.                 Tell (d in StageClosedEvent),
1616.                 Tell (d stage/st s2)
1617.           $
1618. end
```

Sentries

```
1619. ValSentryTeamModellazioneFormato in ECArule with
1620.    mode m: Deferred
1621.    ecarule
1622.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto t/Team d/Individual
1623.              ON Tell (ev in EventoTeamModellazioneFormato)
1624.              IFNEW (p in Progetto) and (ev progetto p)
1625.                    and (s in FormazioneTeamModellazione)
1626.                    and (s parentEntity p)
1627.                    and (m in TeamModellazioneFormato)
1628.                    and (s [milestones] m)
1629.                    and (t in Team) and (p teamModellazione t)
1630.              DO Retell (m status TRUE),
1631.                 CALL CreateIndividual (MlAcEvent, d),
1632.                 Tell (d in MilestoneAchivedEvent),
1633.                 Tell (d milestone/ml m)
1634.              ELSE reject
1635.           $
1636. end
```

```
1637. ValSentryContestoAnalizzato in ECArule with
1638.    mode m: Deferred
1639.    ecarule
1640.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto doc/Documento d/Individual
1641.              ON Tell (ev in EventoContestoAnalizzato)
1642.              IFNEW (p in Progetto) and (ev progetto p)
1643.                    and (s in AnalisiContesto)
1644.                    and (s parentEntity p)
1645.                    and (m in ContestoAnalizzato)
1646.                    and (s [milestones] m)
1647.                    and (doc in Documento)
1648.                    and (p resocontoContesto doc)
1649.              DO Retell (m status TRUE),
1650.                 CALL CreateIndividual (MlAcEvent, d),
1651.                 Tell (d in MilestoneAchivedEvent),
1652.                 Tell (d milestone/ml m)
1653.              ELSE reject
```

```
1654.           $
1655. end
```

```
1656. ValSentrySottoModelliCreati in ECArule with
1657.    mode m: Deferred
1658.    ecarule
1659.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1660.              ON Tell (ev in EventoSottoModelliCreati)
1661.              IFNEW (p in Progetto) and (ev progetto p)
1662.                    and (s in CostruzioneSottoModelli)
1663.                    and (s parentEntity p)
1664.                    and (m in SottoModelliCreati)
1665.                    and (s [milestones] m)
1666.                    and (p in ProgettiConSottoModelli)
1667.              DO Retell (m status TRUE),
1668.                 CALL CreateIndividual (MlAcEvent, d),
1669.                 Tell (d in MilestoneAchivedEvent),
1670.                 Tell (d milestone/ml m)
1671.              ELSE reject
1672.           $
1673. end
```

```
1674. ValSentryModelloCreato in ECArule with
1675.    mode m: Deferred
1676.    ecarule
1677.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto doc/Documento d/Individual
1678.              ON Tell (ev in EventoModelloCreato)
1679.              IFNEW (p in Progetto) and (ev progetto p)
1680.                    and (s in CostruzioneModello)
1681.                    and (s parentEntity p)
1682.                    and (m in ModelloCreato) and (s [milestones] m)
1683.                    and (doc in Documento)
1684.                    and (p modelloDominio doc)
1685.              DO Retell (m status TRUE),
1686.                 CALL CreateIndividual (MlAcEvent, d),
1687.                 Tell (d in MilestoneAchivedEvent),
1688.                 Tell (d milestone/ml m)
1689.              ELSE reject
1690.           $
1691. end
```

```
1692. ValSentryModelloApprovato in ECArule with
1693.    mode m: Deferred
1694.    ecarule
1695.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1696.              ON Tell (ev in EventoModelloApprovato)
1697.              IFNEW (p in Progetto) and (ev progetto p)
1698.                    and (s in VerificaModello)
1699.                    and (s parentEntity p)
1700.                    and (m in ModelloApprovato)
1701.                    and (s [milestones] m)
1702.              DO Retell (m status TRUE),
1703.                 CALL CreateIndividual (MlAcEvent, d),
1704.                 Tell (d in MilestoneAchivedEvent),
```

```
1705.                Tell (d milestone/ml m)
1706.           $
1707. end
```

```
1708. ValSentryModelloDisapprovato in ECArule with
1709.    mode m: Deferred
1710.    ecarule
1711.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1712.              ON Tell (ev in EventoModelloDisapprovato)
1713.              IFNEW (p in Progetto) and (ev progetto p)
1714.                    and (s in VerificaModello)
1715.                    and (s parentEntity p)
1716.                    and (m in ModelloDisapprovato)
1717.                    and (s [milestones] m)
1718.              DO Retell (m status TRUE),
1719.                 CALL CreateIndividual (MlAcEvent, d),
1720.                 Tell (d in MilestoneAchivedEvent),
1721.                 Tell (d milestone/ml m)
1722.           $
1723. end
```

```
1724. ValSentryModellazioneTerminata in ECArule with
1725.    mode m: Deferred
1726.    ecarule
1727.     myecarule : $ ev/MilestoneAchivedEvent m,m2/MilestoneType
   p/Progetto s,s2/StageType d/Individual
1728.              ON Tell (ev in MilestoneAchivedEvent)
1729.              IFNEW (m in ModelloApprovato) and (ev milestone m)
1730.                    and (s in VerificaModello)
1731.                    and (s [milestones] m)
1732.                    and (p in Progetto) and (s parentEntity p)
1733.                    and (m2 in ModellazioneTerminata)
1734.                    and (s2 in Modellazione)
1735.                    and (s2 [milestones] m2)
1736.                    and (s2 parentEntity p)
1737.              DO Retell (m2 status TRUE),
1738.                 CALL CreateIndividual (MlAcEvent, d),
1739.                 Tell (d in MilestoneAchivedEvent),
1740.                 Tell (d milestone/ml m2)
1741.           $
1742. end
```

```
1743. ValSentryTeamFunzionalitàFormato in ECArule with
1744.    mode m: Deferred
1745.    ecarule
1746.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto t/Team d/Individual
1747.              ON Tell (ev in EventoTeamFunzionalitàFormato)
1748.              IFNEW (p in Progetto) and (ev progetto p)
1749.                    and (s in FormazioneTeamFunzionalità)
1750.                    and (s parentEntity p)
1751.                    and (m in TeamFunzionalitàFormato)
1752.                    and (s [milestones] m)
1753.                    and (t in Team) and (p teamFunzionalità t)
1754.              DO Retell (m status TRUE),
1755.                 CALL CreateIndividual (MlAcEvent, d),
```

```
1756.                Tell (d in MilestoneAchivedEvent),
1757.                Tell (d milestone/ml m)
1758.             ELSE reject
1759.          $
1760. end
```

```
1761. ValSentryDominioSuddiviso in ECArule with
1762.    mode m: Deferred
1763.    ecarule
1764.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1765.             ON Tell (ev in EventoDominioSuddiviso)
1766.             IFNEW (p in Progetto) and (ev progetto p)
1767.                   and (s in SuddivisioneDominio)
1768.                   and (s parentEntity p)
1769.                   and (m in DominioSuddiviso)
1770.                   and (s [milestones] m)
1771.                   and (p in ProgettiConAreeTematiche)
1772.             DO Retell (m status TRUE),
1773.                CALL CreateIndividual (MlAcEvent, d),
1774.                Tell (d in MilestoneAchivedEvent),
1775.                Tell (d milestone/ml m)
1776.             ELSE reject
1777.          $
1778. end
```

```
1779. ValSentryFunzionalitàCreate in ECArule with
1780.    mode m: Deferred
1781.    ecarule
1782.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1783.             ON Tell (ev in EventoFunzionalitàCreate)
1784.             IFNEW (p in Progetto) and (ev progetto p)
1785.                   and (s in CreazioneFunzionalità)
1786.                   and (s parentEntity p)
1787.                   and (m in FunzionalitàCreate)
1788.                   and (s [milestones] m)
1789.                   and (p in ProgettiConFunzionalità)
1790.             DO Retell (m status TRUE),
1791.                CALL CreateIndividual (MlAcEvent, d),
1792.                Tell (d in MilestoneAchivedEvent),
1793.                Tell (d milestone/ml m)
1794.             ELSE reject
1795.          $
1796. end
```

```
1797. ValSentryStesuraListaTerminata in ECArule with
1798.    mode m: Deferred
1799.    ecarule
1800.     myecarule : $ ev/MilestoneAchivedEvent m,m2/MilestoneType
   p/Progetto s,s2/StageType d/Individual
1801.             ON Tell (ev in MilestoneAchivedEvent)
1802.             IFNEW (m in FunzionalitàCreate) and (ev milestone m)
1803.                   and (s in CreazioneFunzionalità)
1804.                   and (s [milestones] m)
1805.                   and (p in Progetto) and (s parentEntity p)
1806.                   and (m2 in StesuraListaTerminata)
```

```
1807.                    and (s2 in StesuraListaFunzionalità)
1808.                    and (s2 [milestones] m2)
1809.                    and (s2 parentEntity p)
1810.              DO Retell (m2 status TRUE),
1811.                 CALL CreateIndividual (MlAcEvent, d),
1812.                 Tell (d in MilestoneAchivedEvent),
1813.                 Tell (d milestone/ml m2)
1814.           $
1815. end
```

```
1816. ValSentryTeamPianificazioneFormato in ECArule with
1817.    mode m: Deferred
1818.    ecarule
1819.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto t/Team d/Individual
1820.              ON Tell (ev in EventoTeamPianificazioneFormato)
1821.              IFNEW (p in Progetto) and (ev progetto p)
1822.                    and (s in FormazioneTeamPianificazione)
1823.                    and (s parentEntity p)
1824.                    and (m in TeamPianificazioneFormato)
1825.                    and (s [milestones] m)
1826.                    and (t in Team) and (p teamPianificazione t)
1827.              DO Retell (m status TRUE),
1828.                 CALL CreateIndividual (MlAcEvent, d),
1829.                 Tell (d in MilestoneAchivedEvent),
1830.                 Tell (d milestone/ml m)
1831.              ELSE reject
1832.           $
1833. end
```

```
1834. ValSentryAreeTematichePianificate in ECArule with
1835.    mode m: Deferred
1836.    ecarule
1837.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1838.              ON Tell (ev in EventoAreeTematichePianificate)
1839.              IFNEW (p in Progetto) and (ev progetto p)
1840.                    and (s in PianificazioneAreeTematiche)
1841.                    and (s parentEntity p)
1842.                    and (m in AreeTematichePianificate)
1843.                    and (s [milestones] m)
1844.                    and (p in ProgettiConAreePianificate)
1845.              DO Retell (m status TRUE),
1846.                 CALL CreateIndividual (MlAcEvent, d),
1847.                 Tell (d in MilestoneAchivedEvent),
1848.                 Tell (d milestone/ml m)
1849.              ELSE reject
1850.           $
1851. end
```

```
1852. ValSentryPianificazioneTerminata in ECArule with
1853.    mode m: Deferred
1854.    ecarule
1855.     myecarule : $ ev/MilestoneAchivedEvent m,m2/MilestoneType
   p/Progetto f/Funzionalità s,s2/StageType d/Individual
1856.              ON Tell (ev in MilestoneAchivedEvent)
1857.              IFNEW (m in FunzionalitàPianificata)
```

```
1858.                    and (ev milestone m)
1859.                    and (s in PianificazioneFunzionalità)
1860.                    and (s [milestones] m)
1861.                    and (f in Funzionalità)
1862.                    and (s parentEntity f)
1863.                    and (p in ProgettiConFunzionalitàPianificate)
1864.                    and (p funzionalità f)
1865.                    and (m2 in PianificazioneTerminata)
1866.                    and (s2 in Pianificazione)
1867.                    and (s2 [milestones] m2)
1868.                    and (s2 parentEntity p)
1869.              DO Retell (m2 status TRUE),
1870.                 CALL CreateIndividual (MlAcEvent, d),
1871.                 Tell (d in MilestoneAchivedEvent),
1872.                 Tell (d milestone/ml m2)
1873.           $
1874. end
```

```
1875. ValSentrySoftwareTerminato in ECArule with
1876.    mode m: Deferred
1877.    ecarule
1878.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   p/Progetto d/Individual
1879.              ON Tell (ev in EventoSoftwareTerminato)
1880.              IFNEW (p in Progetto) and (ev progetto p)
1881.                    and (s in Sviluppo) and (s parentEntity p)
1882.                    and (m in SoftwareTerminato)
1883.                    and (s [milestones] m)
1884.                    and (p in ProgettiConFunzionalitàRealizzate)
1885.              DO Retell (m status TRUE),
1886.                 CALL CreateIndividual (MlAcEvent, d),
1887.                 Tell (d in MilestoneAchivedEvent),
1888.                 Tell (d milestone/ml m)
1889.              ELSE reject
1890.           $
1891. end
```

```
1892. ValSentryFunzioalitàPianificata in ECArule with
1893.    mode m: Deferred
1894.    ecarule
1895.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità pers/Persona data/Date d/Individual
1896.              ON Tell (ev in EventoFunzioalitàPianificata)
1897.              IFNEW (f in Funzionalità) and (ev funzionalità f)
1898.                    and (s in PianificazioneFunzionalità)
1899.                    and (s parentEntity f)
1900.                    and (m in FunzionalitàPianificata)
1901.                    and (s [milestones] m)
1902.                    and (pers in Persona) and (data in Date)
1903.                    and (f sviluppatore pers)
1904.                    and (f dataTermine data)
1905.              DO Retell (m status TRUE),
1906.                 CALL CreateIndividual (MlAcEvent, d),
1907.                 Tell (d in MilestoneAchivedEvent),
1908.                 Tell (d milestone/ml m)
1909.              ELSE reject
1910.           $
1911. end
```

```
ValSentrySequenceDiagramRealizzati in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità dp/DesignPackage d/Individual
             ON Tell (ev in EventoSequenceDiagramRealizzati)
             IFNEW (f in Funzionalità) and (ev funzionalità f)
                   and (s in RealizzazioneSequenceDiagram)
                   and (s parentEntity f)
                   and (m in SequenceDiagramRealizzati)
                   and (s [milestones] m)
                   and (dp in DesignPackage)
                   and (f designPackage dp)
                   and (dp in DesignPackageConSequenceDiagram)
             DO Retell (m status TRUE),
                CALL CreateIndividual (MlAcEvent, d),
                Tell (d in MilestoneAchivedEvent),
                Tell (d milestone/ml m)
             ELSE reject
          $
end
```

```
ValSentryHeaderMetodiRealizzati in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità dp/DesignPackage d/Individual
             ON Tell (ev in EventoHeaderMetodiRealizzati)
             IFNEW (f in Funzionalità) and (ev funzionalità f)
                   and (s in StesuraHeaderMetodi)
                   and (s parentEntity f)
                   and (m in HeaderMetodiRealizzati)
                   and (s [milestones] m)
                   and (dp in DesignPackage)
                   and (f designPackage dp)
                   and (dp in DesignPackageConHeaderMetodi)
             DO Retell (m status TRUE),
                CALL CreateIndividual (MlAcEvent, d),
                Tell (d in MilestoneAchivedEvent),
                Tell (d milestone/ml m)
             ELSE reject
          $
end
```

```
ValSentryDesignPackageApprovato in ECArule with
   mode m: Deferred
   ecarule
    myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
             ON Tell (ev in EventoDesignPackageApprovato)
             IFNEW (f in Funzionalità) and (ev funzionalità f)
                   and (s in VerificaDesignPackage)
                   and (s parentEntity f)
                   and (m in DesignPackageApprovato)
                   and (s [milestones] m)
             DO Retell (m status TRUE),
                CALL CreateIndividual (MlAcEvent, d),
```

```
1964.                Tell (d in MilestoneAchivedEvent),
1965.                Tell (d milestone/ml m)
1966.           $
1967. end
```

```
1968. ValSentryDesignPackageDisapprovato in ECArule with
1969.    mode m: Deferred
1970.    ecarule
1971.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
1972.             ON Tell (ev in EventoDesignPackageDisapprovato)
1973.             IFNEW (f in Funzionalità) and (ev funzionalità f)
1974.                   and (s in VerificaDesignPackage)
1975.                   and (s parentEntity f)
1976.                   and (m in DesignPackageDisapprovato)
1977.                   and (s [milestones] m)
1978.             DO Retell (m status TRUE),
1979.                CALL CreateIndividual (MlAcEvent, d),
1980.                Tell (d in MilestoneAchivedEvent),
1981.                Tell (d milestone/ml m)
1982.           $
1983. end
```

```
1984. ValSentryFunzionalitàProgettata in ECArule with
1985.    mode m: Deferred
1986.    ecarule
1987.     myecarule : $ ev/MilestoneAchivedEvent m,m2/MilestoneType
   f/Funzionalità s,s2/StageType d/Individual
1988.             ON Tell (ev in MilestoneAchivedEvent)
1989.             IFNEW (m in DesignPackageApprovato)
1990.                   and (ev milestone m)
1991.                   and (s in VerificaDesignPackage)
1992.                   and (s [milestones] m)
1993.                   and (f in Funzionalità) and (s parentEntity f)
1994.                   and (m2 in FunzionalitàProgettata)
1995.                   and (s2 in Progettazione)
1996.                   and (s2 [milestones] m2)
1997.                   and (s2 parentEntity f)
1998.             DO Retell (m2 status TRUE),
1999.                CALL CreateIndividual (MlAcEvent, d),
2000.                Tell (d in MilestoneAchivedEvent),
2001.                Tell (d milestone/ml m2)
2002.           $
2003. end
```

```
2004. ValSentryFunzionalitàImplementata in ECArule with
2005.    mode m: Deferred
2006.    ecarule
2007.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità doc/Documento d/Individual
2008.             ON Tell (ev in EventoFunzionalitàImplementata)
2009.             IFNEW (f in Funzionalità) and (ev funzionalità f)
2010.                   and (s in Implementazione)
2011.                   and (s parentEntity f)
2012.                   and (m in FunzionalitàImplementata)
2013.                   and (s [milestones] m)
2014.                   and (doc in Documento) and (f codice doc)
```

```
2015.              DO Retell (m status TRUE),
2016.                 CALL CreateIndividual (MlAcEvent, d),
2017.                 Tell (d in MilestoneAchivedEvent),
2018.                 Tell (d milestone/ml m)
2019.           $
2020. end
```

```
2021. ValSentryTestSuperati in ECArule with
2022.    mode m: Deferred
2023.    ecarule
2024.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
2025.              ON Tell (ev in EventoTestSuperati)
2026.              IFNEW (f in Funzionalità) and (ev funzionalità f)
2027.                    and (s in UnitTesting) and (s parentEntity f)
2028.                    and (m in TestSuperati) and (s [milestones] m)
2029.              DO Retell (m status TRUE),
2030.                 CALL CreateIndividual (MlAcEvent, d),
2031.                 Tell (d in MilestoneAchivedEvent),
2032.                 Tell (d milestone/ml m)
2033.           $
2034. end
```

```
2035. ValSentryTestFalliti in ECArule with
2036.    mode m: Deferred
2037.    ecarule
2038.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
2039.              ON Tell (ev in EventoTestFalliti)
2040.              IFNEW (f in Funzionalità) and (ev funzionalità f)
2041.                    and (s in UnitTesting) and (s parentEntity f)
2042.                    and (m in TestFalliti) and (s [milestones] m)
2043.              DO Retell (m status TRUE),
2044.                 CALL CreateIndividual (MlAcEvent, d),
2045.                 Tell (d in MilestoneAchivedEvent),
2046.                 Tell (d milestone/ml m)
2047.           $
2048. end
```

```
2049. ValSentryIspezioneSuperata in ECArule with
2050.    mode m: Deferred
2051.    ecarule
2052.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
2053.              ON Tell (ev in EventoIspezioneSuperata)
2054.              IFNEW (f in Funzionalità) and (ev funzionalità f)
2055.                    and (s in Ispezione) and (s parentEntity f)
2056.                    and (m in IspezioneSuperata)
2057.                    and (s [milestones] m)
2058.              DO Retell (m status TRUE),
2059.                 CALL CreateIndividual (MlAcEvent, d),
2060.                 Tell (d in MilestoneAchivedEvent),
2061.                 Tell (d milestone/ml m)
2062.           $
2063. end
```

```
2064. ValSentryIspezioneFallita in ECArule with
```

```
2065.    mode m: Deferred
2066.    ecarule
2067.     myecarule : $ ev/ExternalEvent m/MilestoneType s/StageType
   f/Funzionalità d/Individual
2068.              ON Tell (ev in EventoIspezioneFallita)
2069.              IFNEW (f in Funzionalità) and (ev funzionalità f)
2070.                    and (s in Ispezione) and (s parentEntity f)
2071.                    and (m in IspezioneFallita)
2072.                    and (s [milestones] m)
2073.              DO Retell (m status TRUE),
2074.                 CALL CreateIndividual (MlAcEvent, d),
2075.                 Tell (d in MilestoneAchivedEvent),
2076.                 Tell (d milestone/ml m)
2077.           $
2078. end
```

```
2079. ValSentryFunzionalitàRealizzata in ECArule with
2080.    mode m: Deferred
2081.    ecarule
2082.     myecarule : $ ev/MilestoneAchivedEvent
   m,mu,mi,m2/MilestoneType s,su,si,sb/StageType f/Funzionalità
   d/Individual
2083.              ON Tell (ev in MilestoneAchivedEvent)
2084.              IFNEW (ev milestone m) and (s in StageType)
2085.                    and (s [milestones] m) and (f in Funzionalità)
2086.                    and (s parentEntity f)
2087.                    and (su in UnitTesting) and (su parentEntity f)
2088.                    and (mu in TestSuperati)
2089.                    and (su [milestones] mu) and (mu status TRUE)
2090.                    and (si in Ispezione) and (si parentEntity f)
2091.                    and (mi in IspezioneSuperata)
2092.                    and (si [milestones] mi) and (mi status TRUE)
2093.                    and (sb in Realizzazione)
2094.                    and (sb parentEntity f)
2095.                    and (m2 in FunzionalitàRealizzata)
2096.                    and (sb [milestones] m2)
2097.                    and ((m in TestSuperati)
2098.                         or (m in IspezioneSuperata))
2099.              DO Retell (m2 status TRUE),
2100.                 CALL CreateIndividual (MlAcEvent, d),
2101.                 Tell (d in MilestoneAchivedEvent),
2102.                 Tell (d milestone/ml m2)
2103.           $
2104. end
```