

Modello di espressione delle regole di adattamento

Survey

Nel presente documento si riassumono i contenuti del deliverable "**D2.3_1 Modello di espressione delle regole di adattamento**" relativo all'attività *RI 2.3 Definizione di modelli per la descrizione di regole di adattamento di contenuti multimediali e servizi interattivi*, nell'ambito del secondo Obiettivo Realizzativo (OR 2) "Studio di servizi e contenuti intelligenti".

Nell'ambito dell'attività *RI 2.3* è stato definito un modello astratto e generico di espressione delle regole di adattamento dal quale è stato istanziato quello di SAPI. Nel definire tale modello si è partiti dai risultati provenienti da due aree di ricerca: *Adaptive Hypermedia Systems (AHS)* e *Active Database*;

Brusilovsky nel 1996 propose di classificare i possibili metodi e tecniche di adattamento sulla base dei principali concetti caratterizzanti una *Regola di Adattamento*, che sono:

- i suoi obiettivi (*Adaptation Goals*);
- cosa (*What*) adattare;
- a che cosa (*to What*) adattarla;
- come (*How*) adattarla.

A partire da questi concetti, si è cercato di definire un modello astratto che descrivesse una generica regola di adattamento, vale a dire una regola in base alla quale si adatta *qualcosa* (dell'*Universo*) ad una data *situazione* (dell'*Universo*) e/o ai cambiamenti di una data situazione a cui essa è sensibile (*situation-awareness*). D'ora in poi, per maggiore eleganza espositiva, verrà denominata *Entità* il *qualcosa* di cui prima.

Prima di iniziare ad elaborare il modello si è ritenuto utile fare una netta distinzione tra capacità di trasformazione assoluta e capacità di trasformazione relativa di una Entità.

Definizione (*capacità di trasformazione assoluta di una Entità*) – Per **capacità di trasformazione assoluta** di una Entità si intende l'insieme dei **metodi di trasformazione** dell'Entità nella sua struttura, contenuto o forma, indipendenti dalle *situazioni*. Tali metodi non hanno consapevolezza del concetto di *situazione* e perciò sono indipendenti dalla data Applicazione.

Definizione (*capacità di trasformazione relativa di una Entità*) – Per capacità di trasformazione relativa di una Entità si intende l'insieme dei metodi di trasformazione dell'Entità nella sua struttura, contenuto o forma, in grado di produrre una versione dell'

Entità che sia la più congeniale possibile ad una data *situazione*. Tali metodi devono necessariamente avere consapevolezza del concetto di *situazione* e conoscerne la codifica. Ne consegue che essi sono dipendenti dalla data Applicazione.

D'ora in poi per maggiore chiarezza espositiva verrà chiamata **capacità di adattamento** di una Entità la sua *capacità di trasformazione relativa* e **metodi di adattamento** i suoi metodi.

Circa l'**attivazione** di una Regola di adattamento si è assunto che in SAPI una Regola di adattamento possa essere attivata nei seguenti casi (event/triggers):

- come richiesta esplicita da parte di un attore (**Adattore**);
- all'occorrenza di un cambiamento della *Situazione*;
- come *propagazione* di un'altra regola.

se e solo se sono soddisfatte alcune *condizioni* (**Pre-Conditions**).

In Figura 1 è riportata la rappresentazione formale in UML di una Regola di Adattamento SAPI

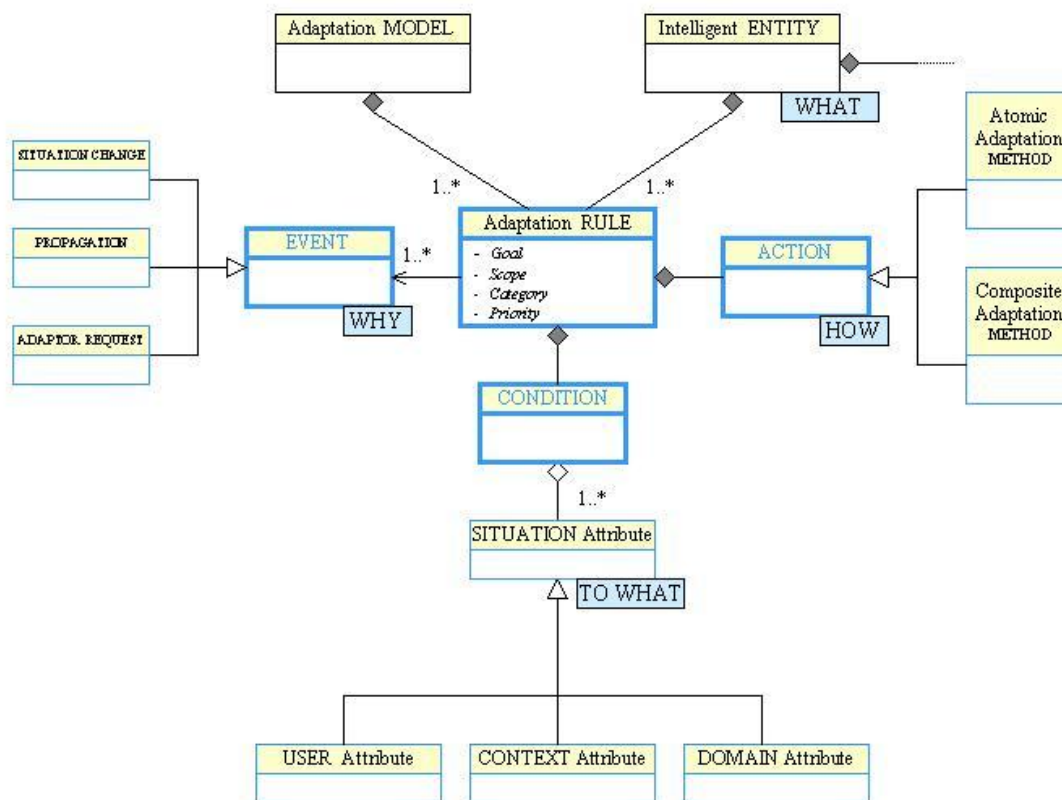


Figura 1 - Rappresentazione formale in UML di una Regola di Adattamento SAPI.

Definizione (*Regola di Adattamento*) – Una Regola di Adattamento \mathcal{A}_r di una Entità \mathcal{E}_n ad una data Situazione \mathcal{S} , attivabile dall'occorrenza di un evento \mathcal{E}_v , è una particolare Regola ECA, in cui l'*implicazione* è una Regola di Horn, così definita:

$$\mathcal{A}_r: \text{on } \mathcal{E}_v, \mathcal{C}_d \Rightarrow \mathcal{A}_d(\mathcal{E}_n, \mathcal{S})$$

dove:

- \mathcal{C}_d è una condizione costituita da una congiunzione di *atomi* della forma $C_i \gamma C_j$ o $C_i K$ con $K =$ costante, $\gamma =$ operatore di comparazione e C_i e C_j caratteristiche di \mathcal{S} ;
- $\mathcal{A}_d(\mathcal{E}_n, \mathcal{S}) =$ Adattamento di \mathcal{E}_n alla situazione \mathcal{S} .

Il significato intuitivo di una regola di adattamento è il seguente: all'occorrenza dell'evento \mathcal{E}_v in una data situazione \mathcal{S} se la condizione \mathcal{C}_d è verificata allora esegui l'adattamento dell'Entità \mathcal{E}_n alla situazione \mathcal{S} . Più in generale il paradigma ECA può così essere espresso:

On Event if Condition then do Action(s)

Livelli di astrazione delle Regole

Generalmente le Regole rappresentano frammenti di conoscenza *self-contained* che tipicamente vengono utilizzati in una qualche forma di ragionamento. Esse possono specificare ad esempio:

- vincoli di integrità statica o dinamica di un sistema (*integrity rules*);
- derivazioni di nuovi concetti (*derivation rules*);
- il comportamento reattivo di un sistema in risposta a determinati eventi (*reaction rules*).

Come illustrato in Figura 2 le regole possono essere descritte a tre livelli di astrazione:

1. *Computation Independent Business Domain Level* (CIM nella MDA dello OMG); a questo livello di astrazione le regole vengono espresse in modo dichiarativo utilizzando un linguaggio naturale o un linguaggio visivo.

2. *Platform Independent Operational Design Level* (PIM nella MDA dello OMG); a questo livello di astrazione le regole vengono espresse utilizzando un linguaggio formale o un paradigma computazionale che può essere facilmente tradotto in istruzioni eseguibili di un sistema software. Esempi di linguaggi di espressione di regole a questo livello sono R2ML e OCL 2.0.
3. *Platform Specific Implementation Level* (PSM nella MDA dello OMG); a questo livello di astrazione le regole sono espresse in un linguaggio di uno specifico ambiente di esecuzione come le viste di Oracle 10g, Jess 3.4, XSB 2.6 Prolog, or the Microsoft Outlook 6 Rule Wizard.

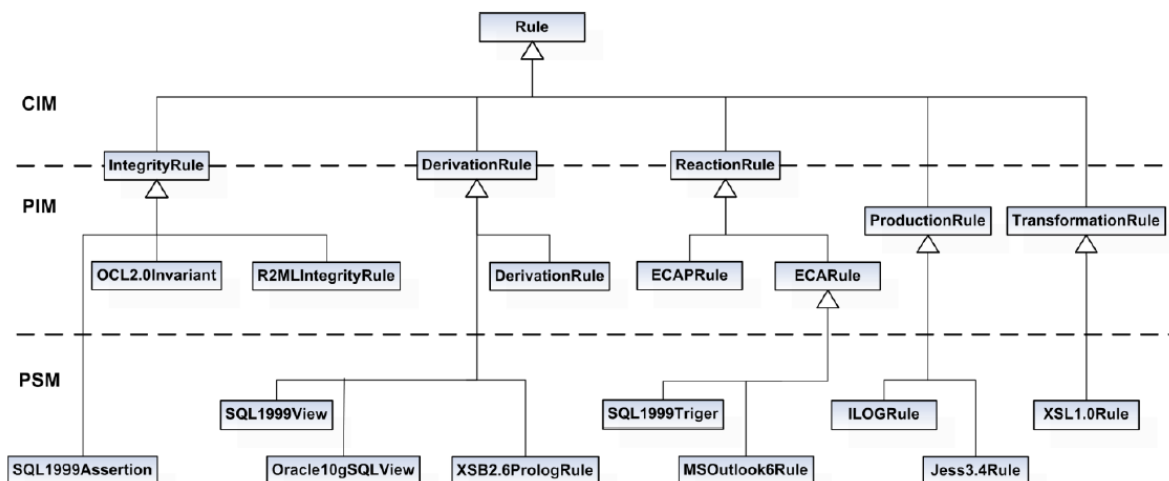


Figura 2 - La descrizione delle Regole a tre livelli di astrazione: CIM (Computation Independent Modeling), PIM (Platform Independent Modeling) e PSM (Platform Specific Modeling).

Scelta del linguaggio di espressione delle regole di adattamento

Lo scopo principale di un linguaggio di markup per le regole è quello di consentire la pubblicazione, il deploying, il riutilizzo e la comunicazione delle regole nel Web e in altri sistemi distribuiti. In altre parole i *rule markup language* consentono di specificare le regole in modo dichiarativo come unità modulari autonome interscambiabili tra differenti sistemi e tools. Essi ad esempio giocano un importante ruolo nel facilitare le interazioni B2C (Business-to-Customer) e B2B (Business-to-Business) in internet. Data la natura di SAPI e il suo *scope* applicativo, nel selezionare il linguaggio di regole più appropriato per la descrizione formale delle regole di adattamento abbiamo adottato un criterio di comparazione basato sostanzialmente sui seguenti requisiti (e "non-requisiti") di SAPI:

1. SAPI non richiede un linguaggio di regole che consenta l'interscambio delle stesse tra sistemi eterogenei (rule interoperability);

2. Il linguaggio di regole deve essere indipendente dalla piattaforma computazionale. Ciò al fine di contribuire a migliorare la flessibilità di SAPI;
3. Il linguaggio sia supportato da tool di editing e di validazione a larga diffusione e "maturi".
4. Il linguaggio di regole CPI (Computational Platform Independent) deve essere facilmente traducibile in un linguaggio di regole eseguibile che disponga di API (Application Programming Interface) richiamabili direttamente o indirettamente nel linguaggio di programmazione (ad es., Java) dei moduli software che fanno uso delle regole di adattamento;
5. Nelle regole di adattamento di SAPI il conseguente (head) è rappresentato da una "procedural call" alla corrispondente *primitiva di adattamento*;
6. È auspicabile che la soluzione scelta offra la possibilità di annotare semanticamente le regole con informazioni descrittive quali : *adaptation goal, category, priority, scope e parametri di qualità* (costo, tempo di esecuzione, etc.). Tali informazioni associate alla regola "as a whole" (*rule profile*) sono molto utili sia per la fase di *rule discovery* sia nel realizzare piani di adattamento (*adaptation plan*) che richiedono l'utilizzo di più regole (*rule adaptation path*).
7. Tra le soluzioni candidate equivalenti va privilegiata quella più simile se non addirittura coincidente con altre già presenti (ad es. SWRL) nell'attuale architettura tecnologica SAPI. Ciò ovviamente sia a vantaggio della semplificazione e omogeneità architetturale sia per ridurre l'effort formativo delle risorse umane.

Rational a supporto della scelta del linguaggio di regole

In SAPI si è scelto SWRL come linguaggio CPI per la descrizione formale delle regole di adattamento integrato con un sottoinsieme dell'ontologia OWL-S. Questa soluzione a primo acchito può sorprendere. È noto infatti che OWL-S è un'ontologia per la descrizione semantica dei *Servizi* che già prevede l'integrazione con vari linguaggi di regole tra cui SWRL. E allora, perchè OWL-S?

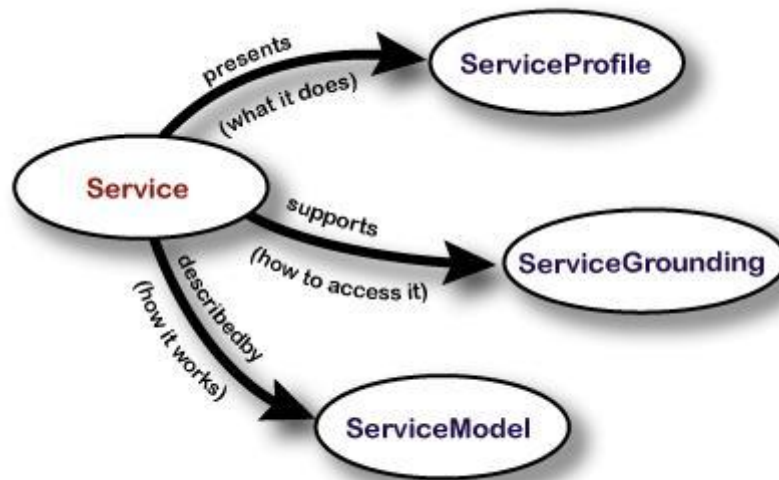


Figura 3 - OWL-S Top level ontology.

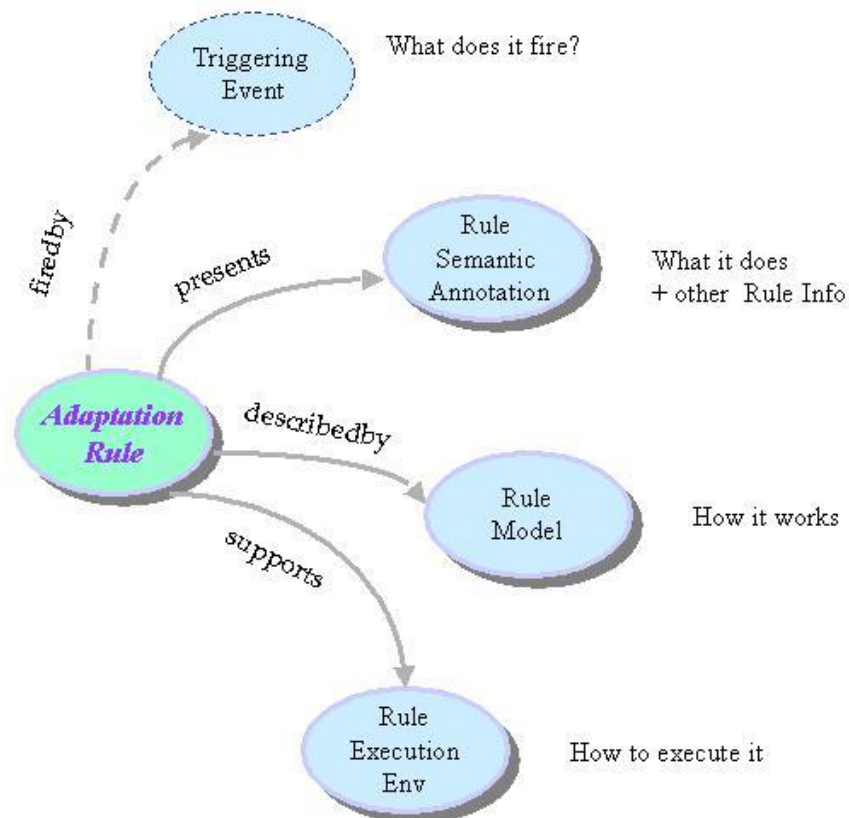


Figura 4 - Ipotesi di una Top level ontology relativa ad una (E)CA Adaptation Rule.

Tale scelta è maturata sulla base delle seguenti considerazioni:

1. se si pensasse di implementare una *Regola di adattamento* come un *Servizio di adattamento*, allora OWL-S + SWRL sarebbe la scelta elettiva per le stesse ragioni per cui lo è stata per i Servizi applicativi;
2. si noti che, se si trascurasse la "classe" *triggering event*, la somiglianza tra la top level ontology OWL-S relativa ai Servizi (Figura 3) e quella ipotizzata per le Regole di adattamento (Figura 4), nomi a parte, sarebbe davvero notevole:
 - tra le proprietà dell'ontologia Profile sono incluse le seguenti:

- ❖ *serviceCategory*;
- ❖ *serviceParameter*;
- ❖ *haseffect*;
- ❖ *hasPrecondition* (le istanze sono definite secondo lo schema fornito dalla Process Ontology).

Le prime tre proprietà sono sufficienti a soddisfare il requisito numero 6 relativo all'annotazione semantica delle regole di adattamento.

- Lo schema relativo alle *Condition* incluso nell'ontologia Process di OWL-S Integrato con le ontologie SAPI descrittive le situazioni, consente di descrivere il *body* di una regola di adattamento;
 - OWL-S *Perform* e OWL-S *Process* della Process ontology consentono di descrivere la *head* di una regola di adattamento specificando l'identificativo della relativa primitiva di adattamento;
3. Analogamente ai Servizi Web, anche le Regole di adattamento potrebbero essere corredate da un'ontologia Grounding completamente da definire ex-novo al fine di creare un isomorfismo tra la descrizione semantica PIM delle regole e quella PSM (vedi Figura 2) nel linguaggio eseguibile scelto (ad es. Jess);
 4. I linguaggi di regole indipendenti dalla piattaforma computazionale diversi da SWRL (Reaction RuleML, R2ML, etc.) e/o i relativi ambienti di sviluppo non sono ancora affidabili; per contro:
 - le SWRL rules sono facilmente trasformabili in Jess rules usando, ad esempio, XSLT;

- il rule engine Jess è implementato in Java e supporta sia il forward che il backward chaining;
- la sua implementazione è stabile e ben supportata;
- è un software largamente diffuso negli ambienti accademici, di ricerca e industriali;
- non esistono ad oggi stabili implementazioni di rule engine che supportano direttamente SWRL;
- Jess può essere facilmente integrato in Protégé;
- Una regola Jess ha la seguente sintassi:

```
(defrule ruleName
  (predicate1 constant1 ?boundVariable1)
  (test (jMethod1 constant2 ?boundVariable1))
  =>
  (jMethod2 (symbol3 ?boundVariable1)) )
```

La parte LHS di una regola Jess consiste generalmente di una lista di *pattern* base (atomi) che sono interpretati implicitamente come una congiunzione.

La parte RHS di una regola Jess è rappresentata da un'azione. *Pattern* e *azione* fanno parte della terminologia Jess. Un'azione corrisponde semplicemente ad una call ad una procedura Java di Jess ("JMethod" nell'esempio di cui sopra) che nella terminologia di Jess viene chiamata "function".

- la soluzione SWRL + Jess soddisfa pienamente i requisiti 3 4 e 5.

Essendo Jess un prodotto commerciale durante la fase di prototipazione di SAPI verrà considerata la possibilità di utilizzare come valida alternativa, sostanzialmente equivalente a Jess rispetto ai requisiti progettuali, il rule engine Open Source *Jboss Rules* (aka *Drools*).

Come considerazioni finali giova sottolineare che come un Web Service, una Regola necessita di una sua annotazione semantica (Profile nella terminologia OWL-S) attraverso cui specificare attributi come *adaptation goal, priority, scope, category, QoS parameter, precondition, effects, etc.*, che possono essere utilizzati per la *rule discovery*, e nell'ambito della strategia di composizione delle regole per risolvere conflitti e ricercare il percorso di adattamento ottimo.

Inoltre val la pena notare che, per la parte *Grounding* della descrizione semantica delle regole, si potrebbe integrare OWL-S con la descrizione semantica del mapping ad un linguaggio di regole eseguibile, ovvero CPD (Computational Platform Dependent) come Jess o Drools. Si otterrebbe in questo modo una versione di OWL-S specifica per le Regole, vale a dire ... OWL-R.