YAWL Workflow Management System

Gabriele Pozzani Barbara Oliboni

Sistemi informativi aziendali
Laurea magistrale in Ingegneria e scienze informatiche

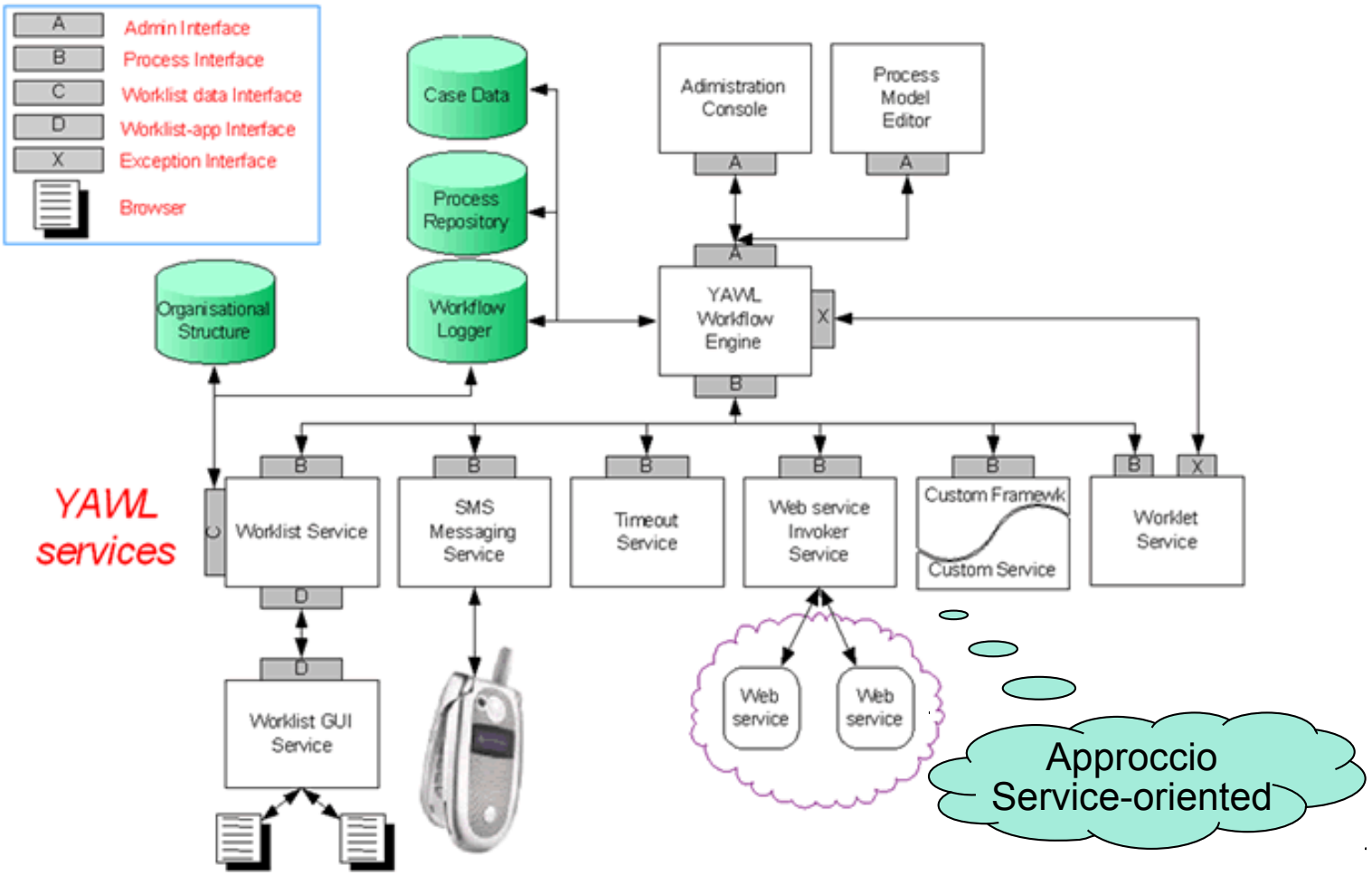
http://www.yawlfoundation.org/

Materiale prodotto da: Marco Bazzoni, Simone Marchesini, Giovanni Zorzato, Matteo Gozzi

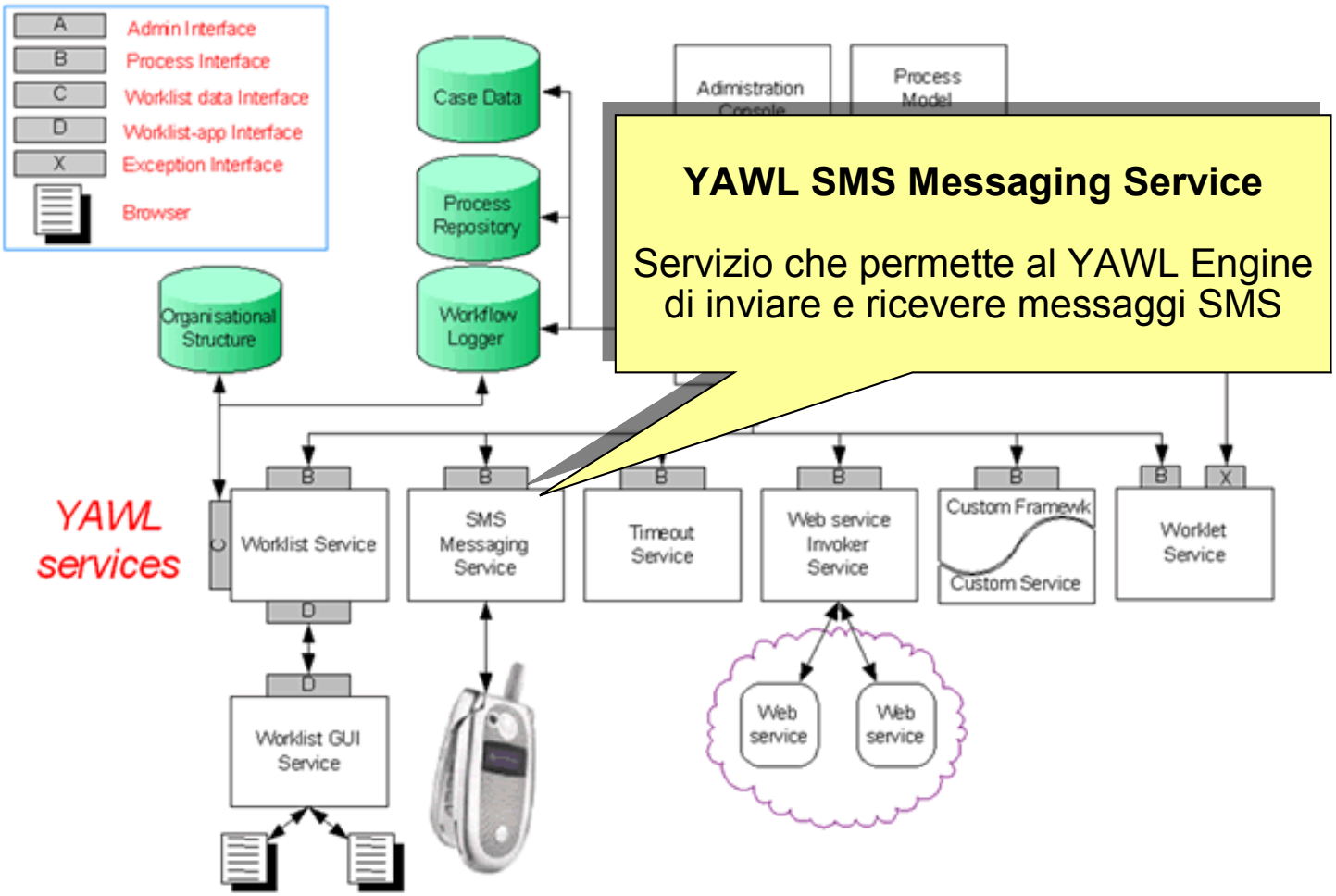
Sommario

Sommario

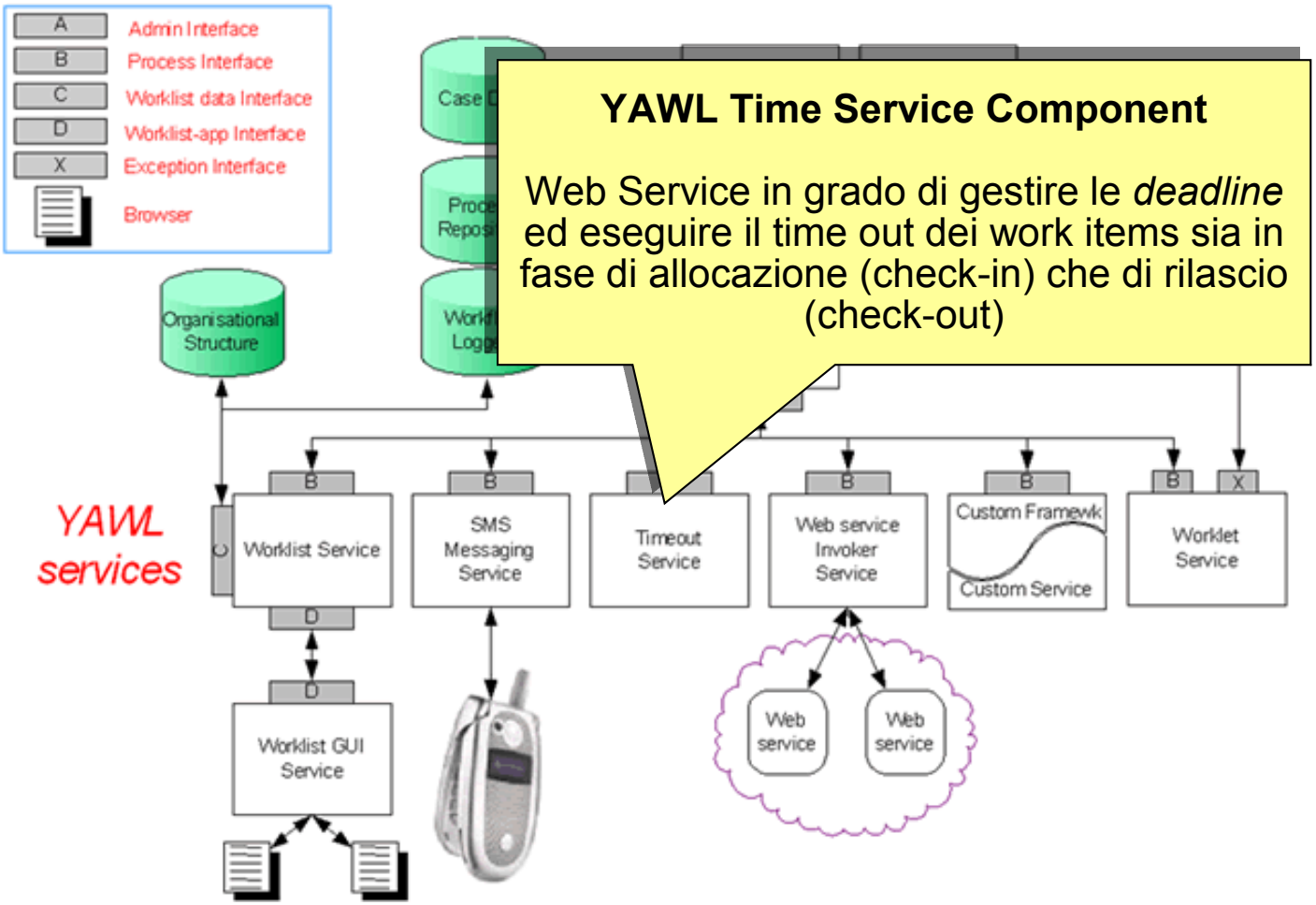
Architettura di YAWL e servizi



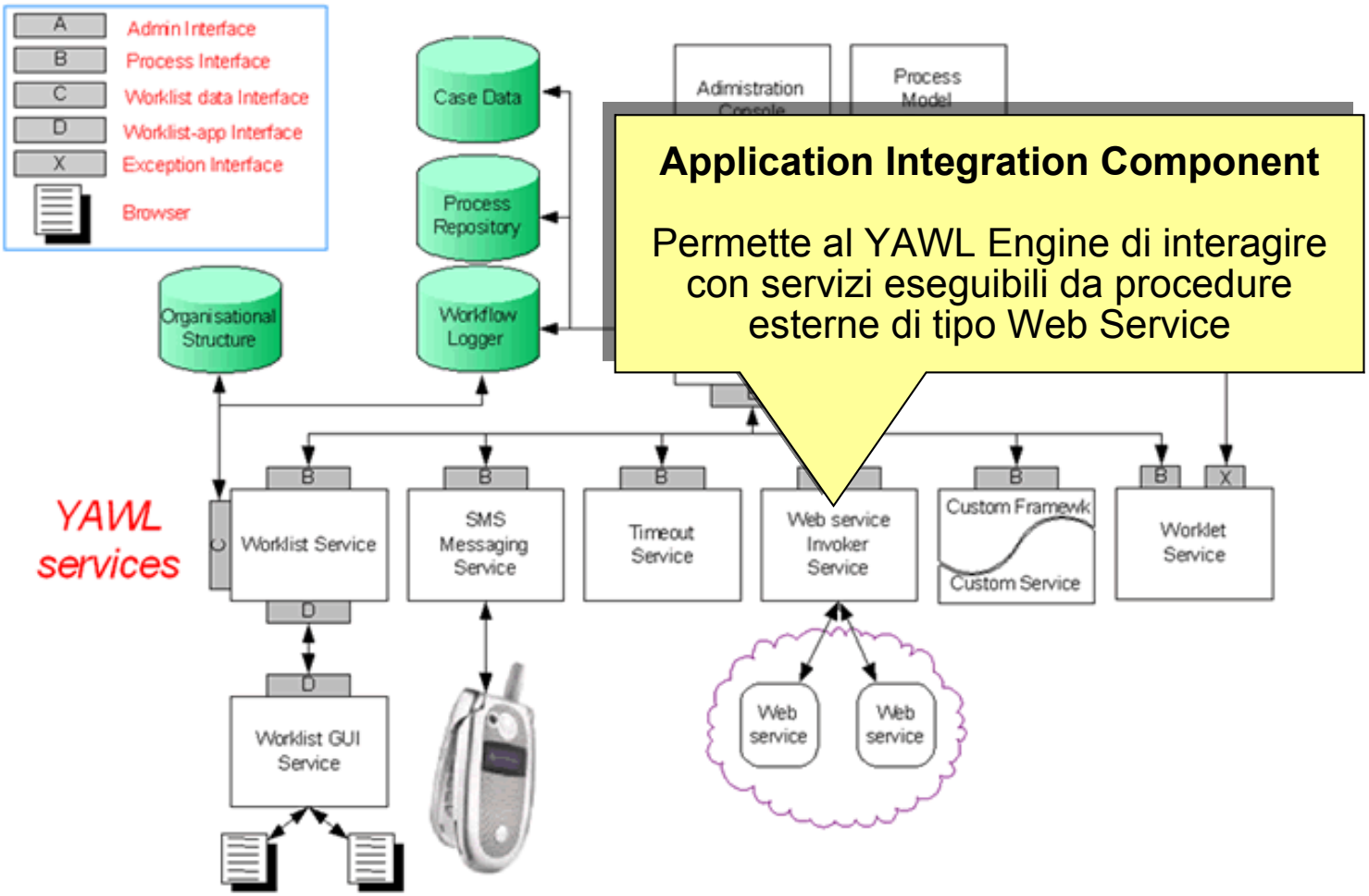
Architettura di YAWL e servizi



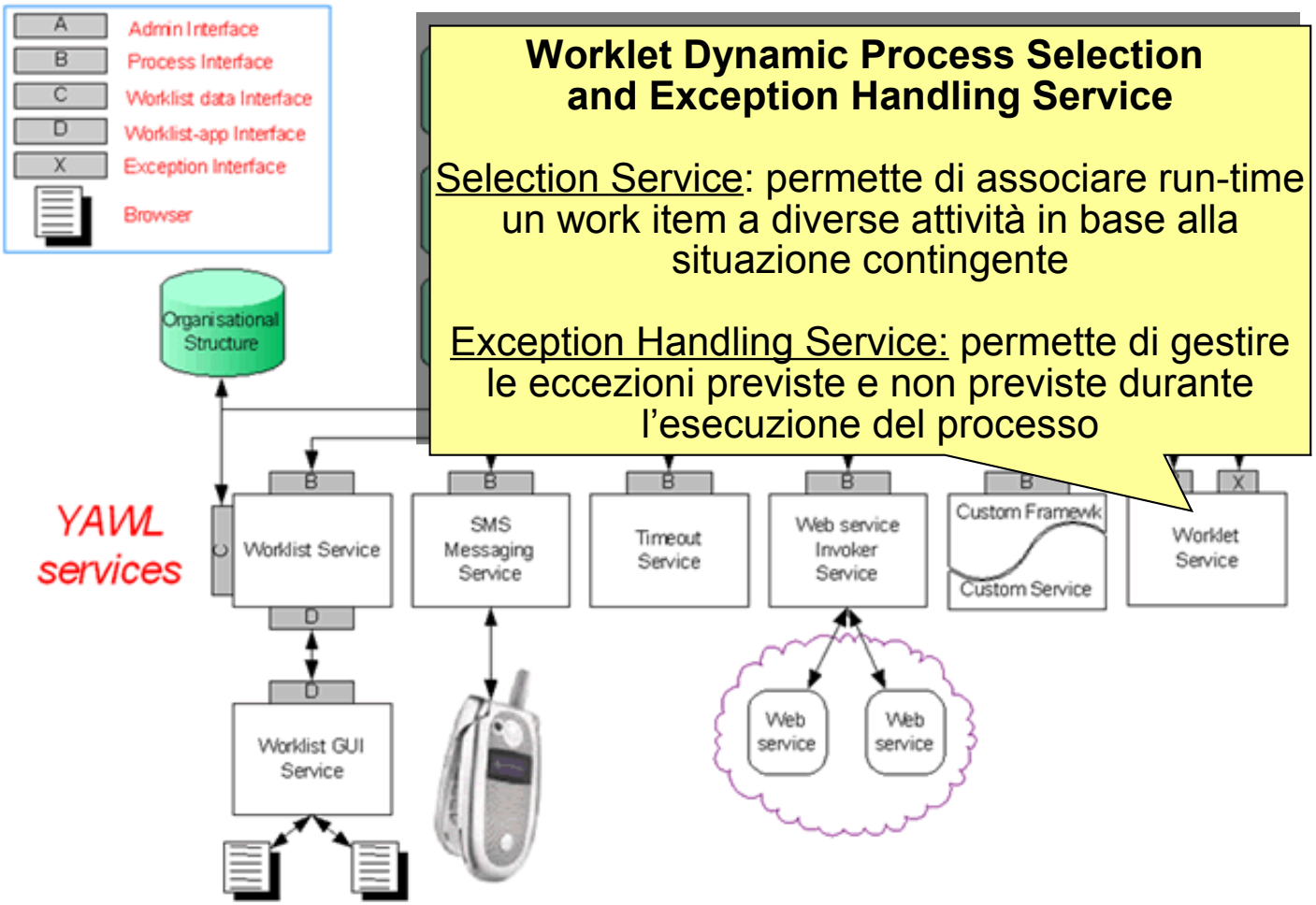
Architettura di YAWL e servizi



Architettura di YAWL e servizi



Architettura di YAWL e servizi



Worklet Dynamic Process Selection and Exception Handling Service

Selection Service: permette di associare run-time un work item a diverse attività in base alla situazione contingente

Exception Handling Service: permette di gestire le eccezioni previste e non previste durante l'esecuzione del processo

Sommario

Custom Services

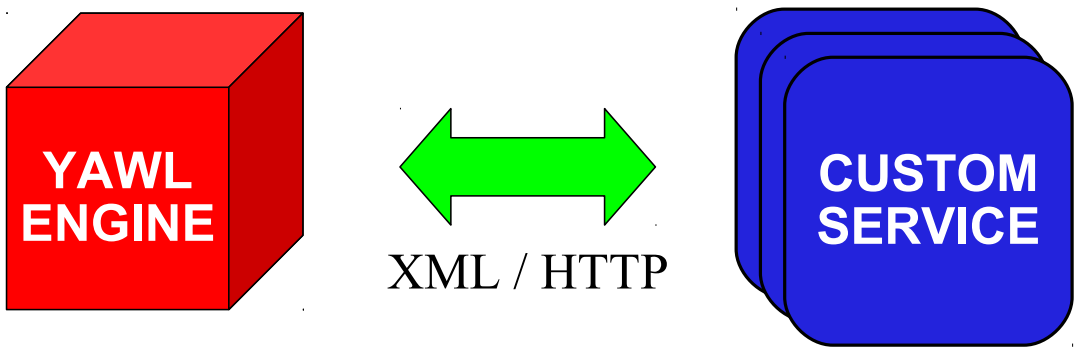
Le applicazioni esterne possono interagire con il YAWL Engine nei seguenti modi:

- Direttamente, utilizzando messaggi XML su protocollo HTTP
  - i servizi vengono definiti Custom YAWL Services
- Indirettamente, sfruttando lo YAWL Web Services Invoker
  - l'Engine YAWL comunica con lo YAWLWSInvoker che a sua volta inoltra le richieste ad un servizio esterno.

Custom Services

Questi servizi possono essere sviluppati in qualsiasi linguaggio di programmazione ed essere eseguiti su qualsiasi piattaforma capace di gestire messaggi XML via HTTP.

Vengono registrati nel YAWL Engine specificando l'URL al quale risponde il servizio.



YAWL WSInvoker

YAWL WSInvoker è necessario quando un'applicazione esterna deve essere invocata dall'Engine per eseguire il lavoro corrispondente ad un'istanza di task.

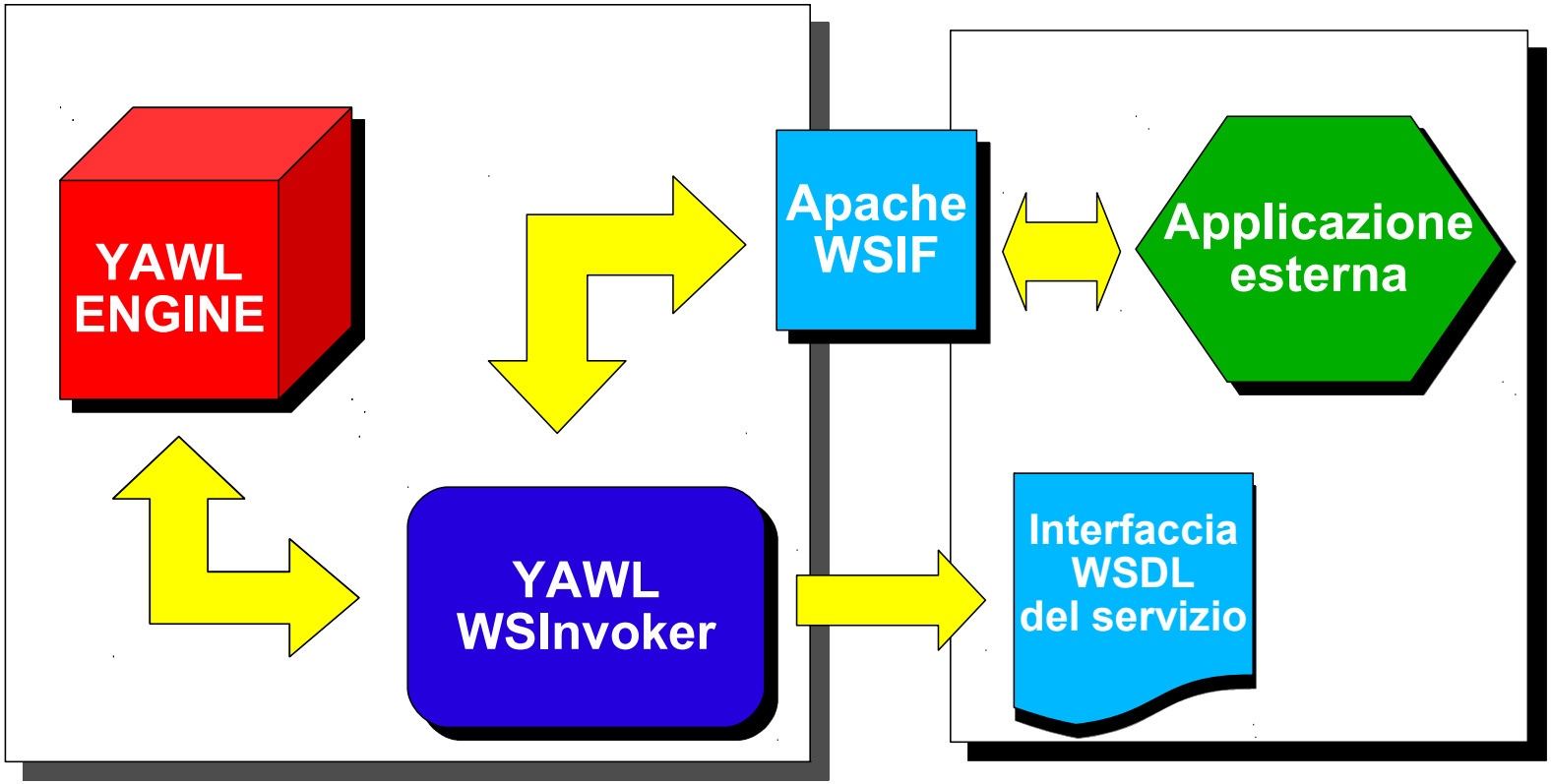
Il task viene "decomposto" in un'invocazione al servizio esterno (l'invocazione è espressa utilizzando l'interfaccia WSDL presente nella descrizione del processo).

YAWL WSInvoker

Esecuzione del task via Web Service:

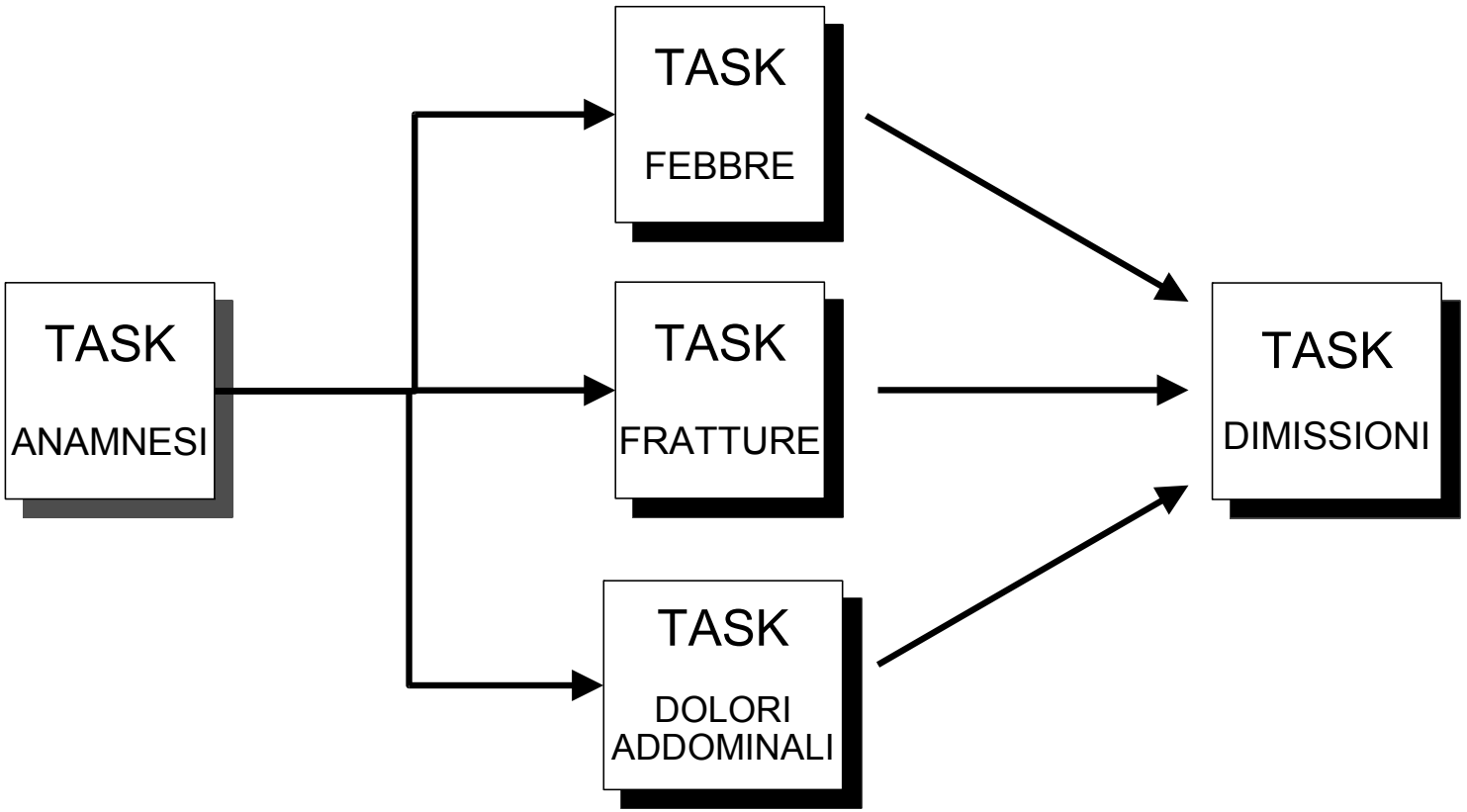
1. Quando viene attivata un'istanza del task, lo YAWL WSInvoker recupera l'interfaccia WSDL del task e, utilizzando l'Apache Web Services Invocation Framework (WSIF), ricava le informazioni necessarie per invocare il servizio;
2. il passaggio all'applicazione esterna avviene immediatamente e lo stato del task evolve da "enabled" a "processing";
3. quando il servizio termina di eseguire l'operazione richiesta, il task riprende il controllo passando in modalità "completed".

YAWL WSInvoker



Sommario

Modello senza worklet



Senza le worklet, OGNI caso possibile deve essere previsto nella specifica del workflow.

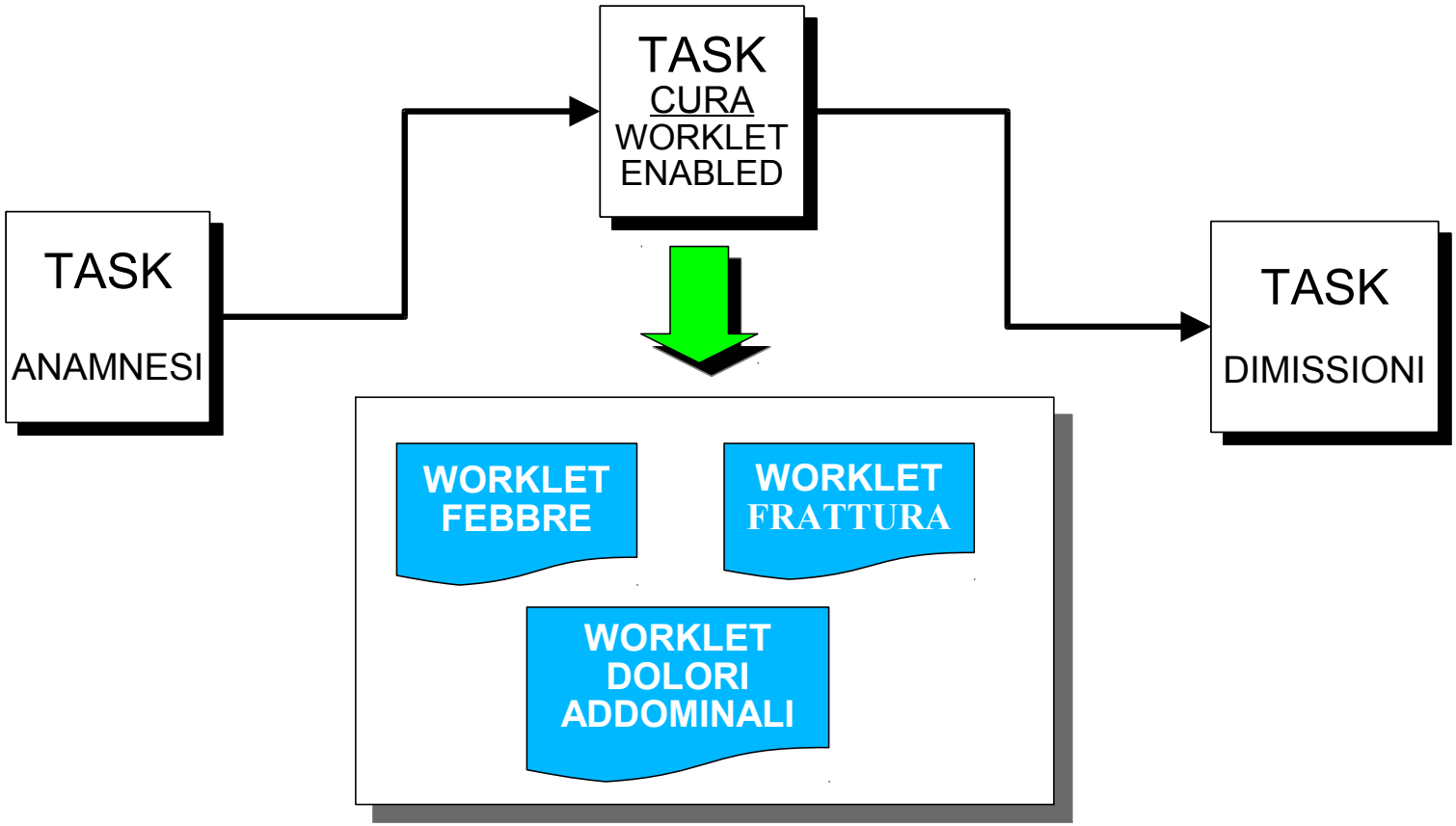
Modello con worklet

Non sempre è possibile definire A PRIORI tutti i casi possibili nella specifica del workflow.

In quei casi si usano le worklet.

Una worklet è un processo a sè stante, definito secondo il formalismo di YAWL, il quale agisce come una subnet per un workitem, permettendo così di gestire in modo più completo e flessibile uno specifico task.

Modello con worklet



Modello con worklet

Vantaggi del modello con worklet:

- permette di semplificare la struttura delle specifiche di workflow per i casi complessi;
- le specifiche possono così essere definite più semplicemente e poi evolvere dinamicamente;
- si possono aggiungere worklet per un task IN MOMENTI SUCCESSIVI per gestire nuove casistiche.

Modello con worklet

WORKLET

- è implementata come una specifica di workflow (workflow specification);
- dichiara come variabili di rete un sottoinsieme di quelle del task per cui svolge il servizio;
  - NB: può possederne anche di proprie
- risiede nel repository (repository / worklets).

Modello con worklet

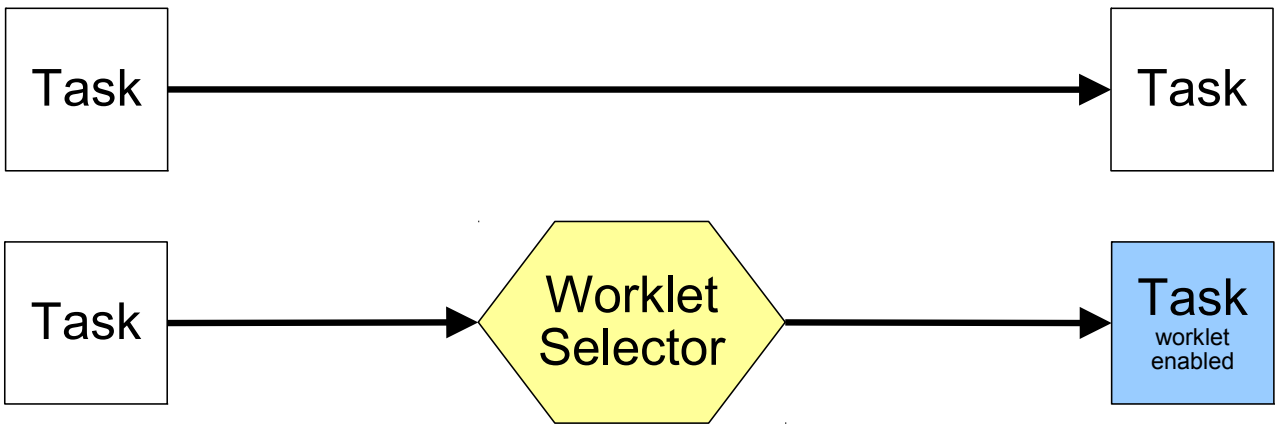
Un modello di workflow con worklet è composto di:

- una o più specifiche di workflow;
- una o più worklet che verranno sostituite a particolari task a runtime;
- un file di regole (.xrs) per ogni specifica che utilizzi il servizio.

Worklet Selection Service

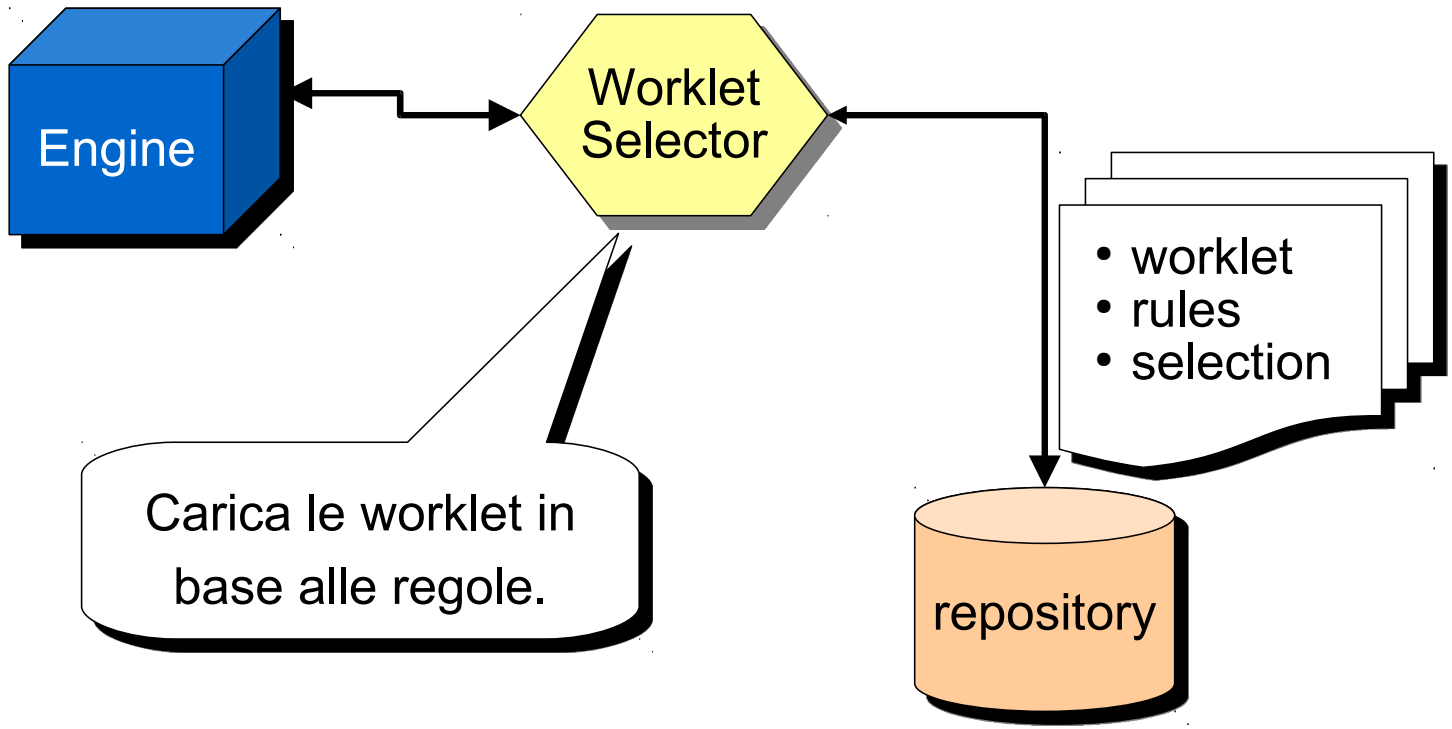
È un servizio esterno, integrato con il YAWL Engine.

Permette di sostituire dinamicamente un oggetto del workflow (workitem) con una worklet.

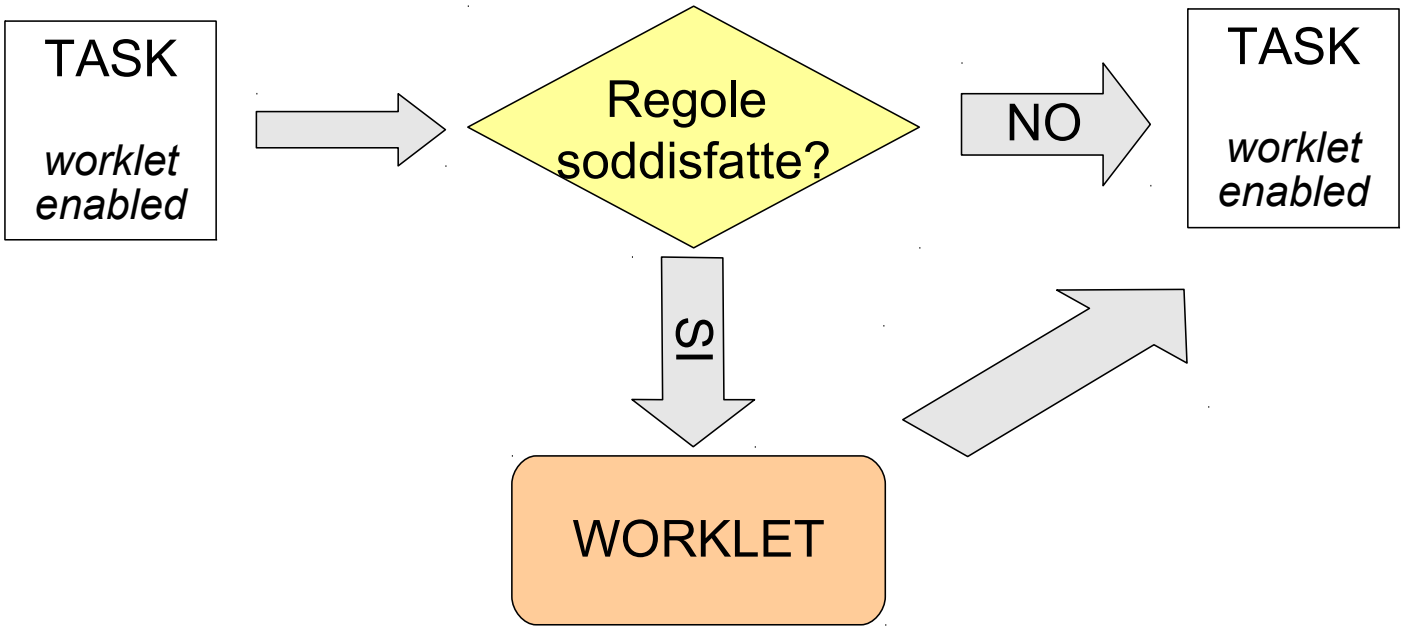


Flusso verso un workitem abilitato al servizio.

Worklet Selection Service



Worklet Selection Service



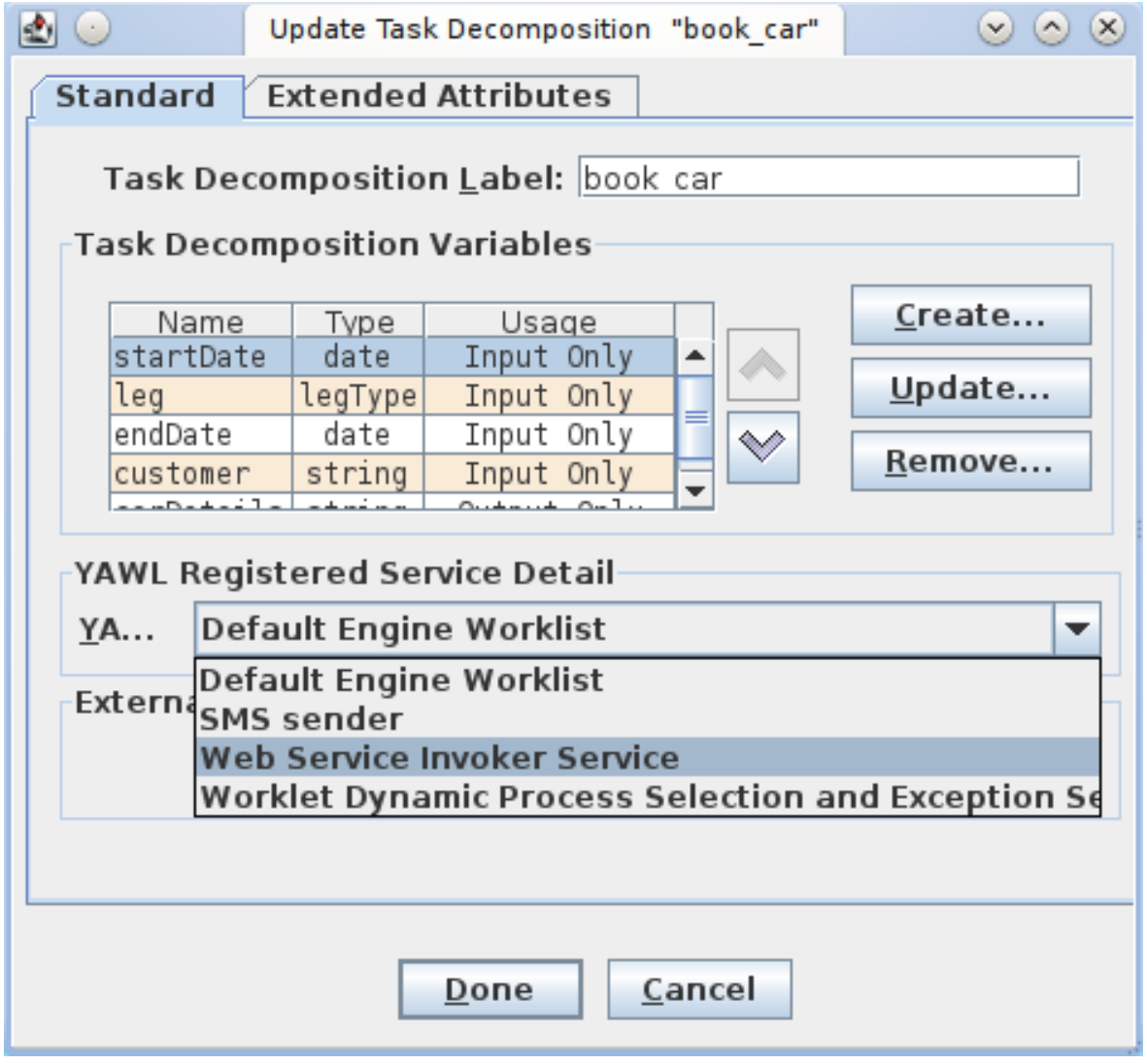
Worklet Selection Service

Se le regole sono soddisfatte il task viene temporaneamente “eliminato” e la worklet selezionata è inserita al suo posto.

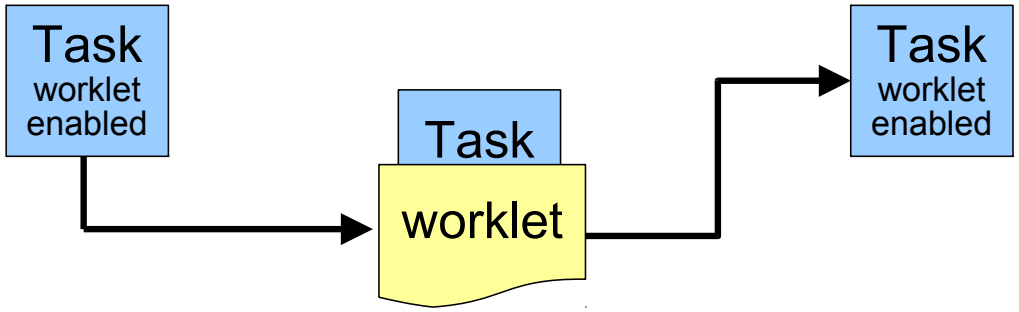
In quel caso, alla worklet vengono passate le variabili di input del task.

Quando la worklet ha terminato, le sue variabili di output vengono restituite al task, dopo che questo è stato nuovamente inserito nel motore.

Abilitare il servizio per un task



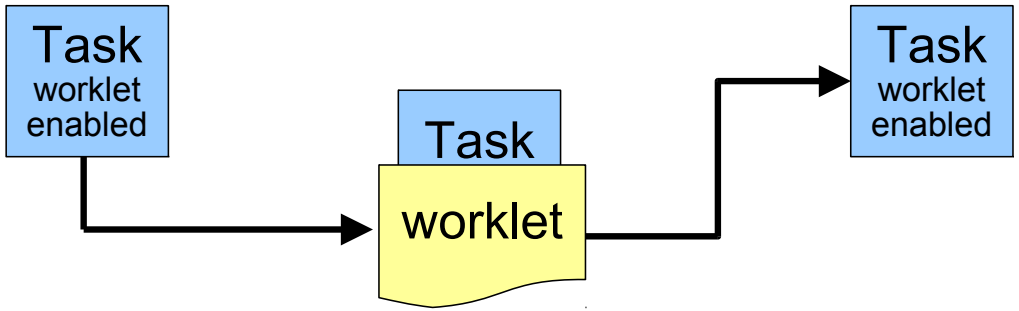
Passaggio di dati



Le variabili di tipo InputOnly e Input&Output ricevono il valore delle corrispondenti variabili nel task

Le variabili di tipo OutputOnly e Input&Output restituiscono il valore alle corrispondenti variabili nel task

Passaggio di dati



Le variabili della rete con lo stesso nome di quelle della decomposizione del task vengono mappate sulla worklet.

Se dichiarate “Input Only” il valore verrà letto dalla worklet, se “Output Only” il valore verrà restituito dalla worklet al task, se “Input & Output” il valore verrà sia letto all’avvio della worklet che restituito al task al termine dell’elaborazione della worklet.

Le regole

Il Worklet Selector basa le proprie decisioni per la sostituzione delle worklet su un insieme di regole specificato in formato XML in un file (.xrs).

Il file delle regole deve essere sempre collocato nel repository (/rules) e avere lo stesso nome della specifica contenente i task abilitati al servizio.

Le regole

```
<?xml version="1.0" encoding="utf-8"?>
<spec>
  <task name="Treat">
    <ruleNode>
      <id>0</id>
      <parent>-1</parent>
      <trueChild>1</trueChild>
      <falseChild>-1</falseChild>
      <condition>True</condition>
      <conclusion>null</conclusion>
      <cornerstone></cornerstone>
      <description>root level default node</description>
    </ruleNode>
    <ruleNode>
      <id>1</id>
      <parent>0</parent>
      <trueChild>-1</trueChild>
      <falseChild>2</falseChild>
      <condition>Fever = true</condition>
      <conclusion>TreatFever</conclusion>
      <cornerstone>
```

Le regole

Il file di regole contiene un elemento complesso (<task>) per ogni task abilitato al servizio.

Ogni elemento task è come un albero composto da più nodi a formare una struttura binaria.

```
<spec>
    <task name="nome_task">
        <ruleNode>
            [...]
        </ruleNode>
        <ruleNode>
            [...]
        </ruleNode>
    </task>
    <task name="nome_task2">
        [...]
    </task>
    [...]
</spec>
```

Le regole

Ogni nodo possiede:

- Un identificativo;
- Un padre;
- Una condizione;
- Una conclusione.

```
<ruleNode>
    <id>1</id>
    <parent>0</parent>
    <trueChild>2</trueChild>
    <falseChild>-1</falseChild>
    <condition>pippo = true</condition>
    <conclusion>nome_worklet</conclusion>
    [...]
</ruleNode>
```

Inoltre può avere:

- Un nodo figlio per il ramo falso (false);
- Un nodo figlio per il ramo vero (true).

Un nodo foglia è specificato mettendo a -1 sia trueChild che falseChild

Le regole

L'albero viene attraversato dalla radice alle foglie valutando tutte le condizioni dei nodi attraversati.

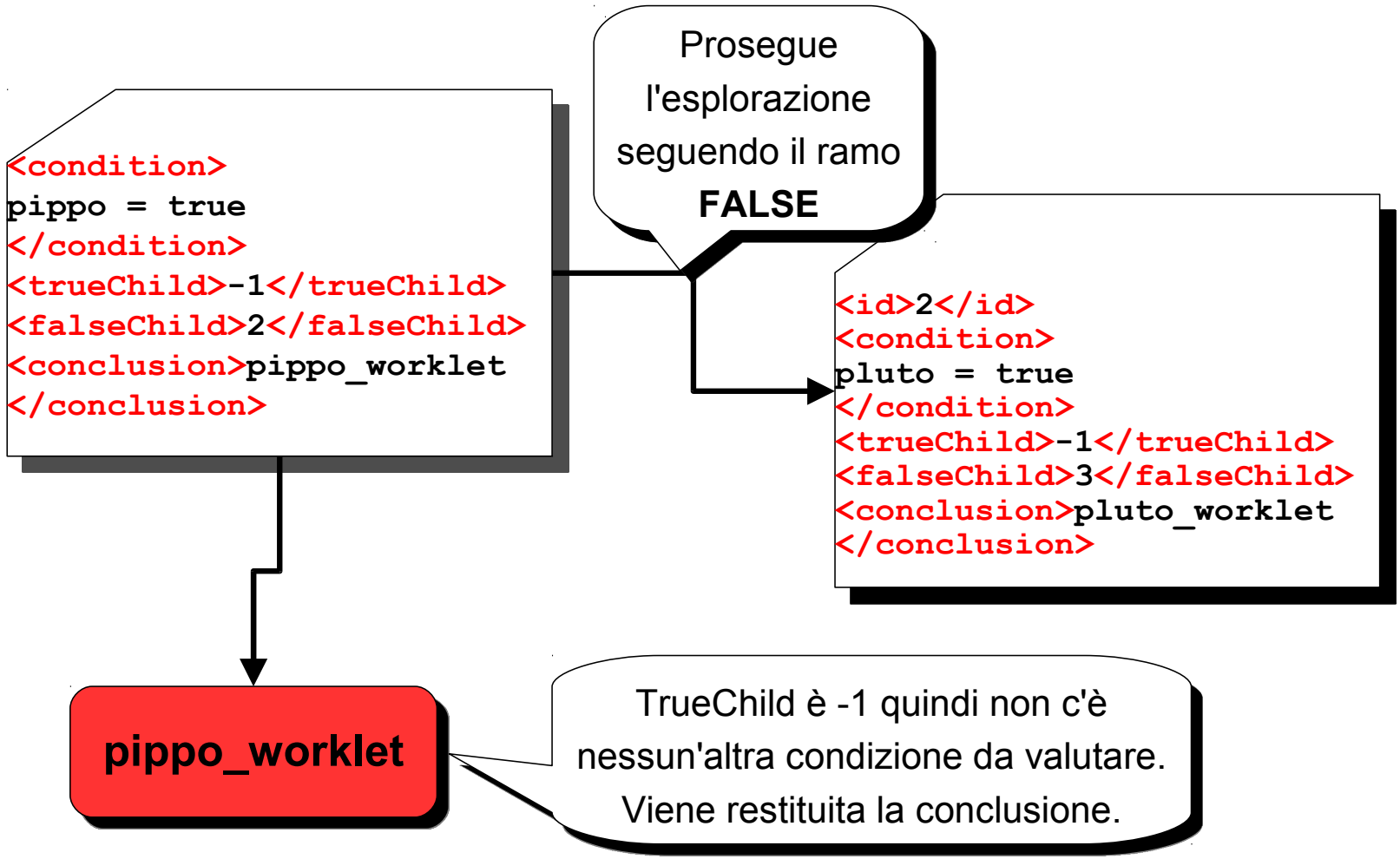
Se un nodo viene valutato vero e possiede un "trueChild" verrà valutata anche la condizione di quest'ultimo.

Analogamente se un nodo è valutato falso e possiede un "falseChild" si prosegue l'analisi verso tale figlio.

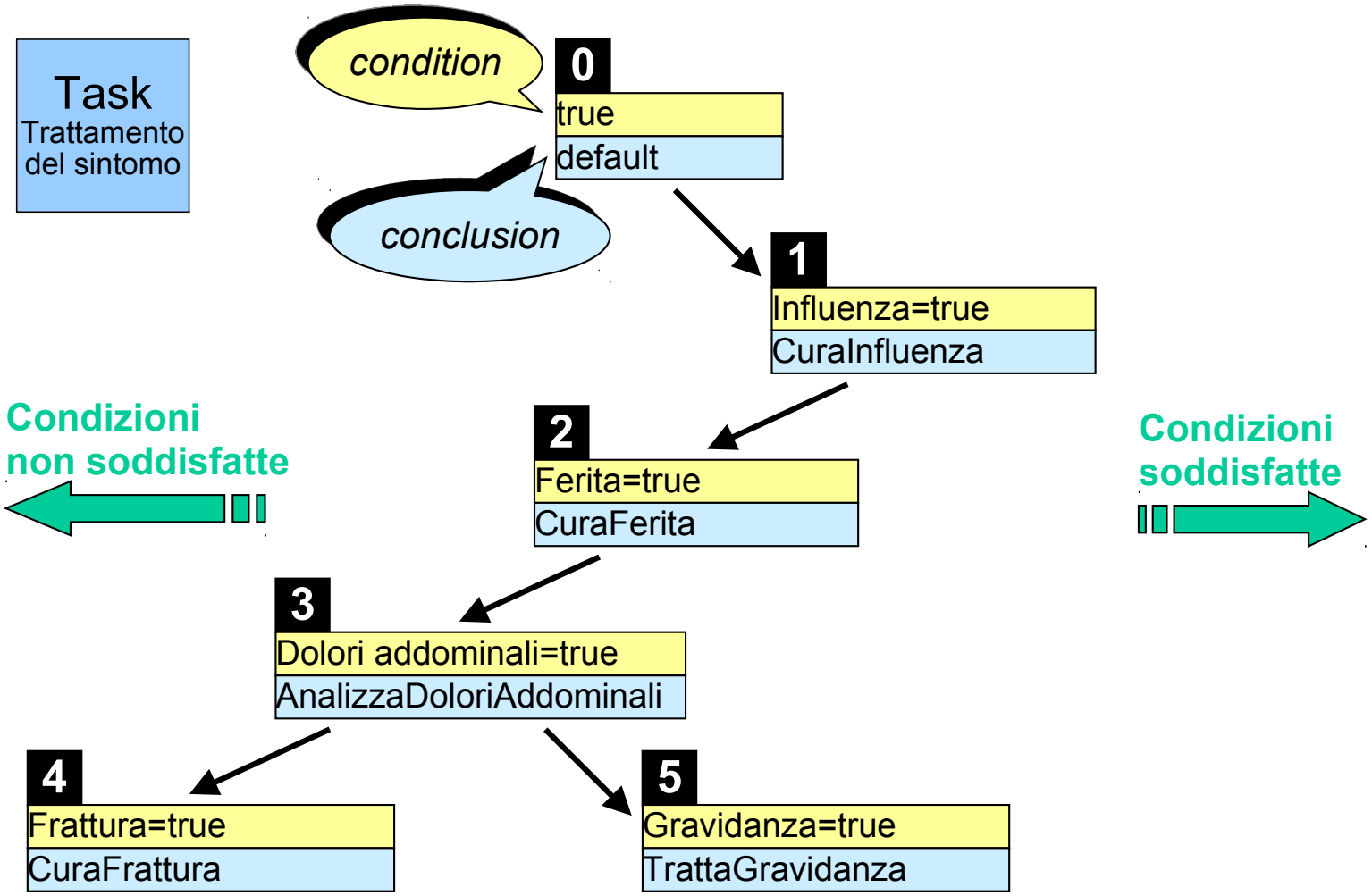
Le regole

Quando viene raggiunto un nodo foglia, se la sua condizione è valutata vera, viene restituita la sua conclusione, altrimenti viene restituita la conclusione dell'ultimo nodo incontrato, attraversando la struttura dell'albero, la cui condizione era stata valutata vera.

Le regole


<condition>
pippo = true
</condition>
<trueChild>-1</trueChild>
<falseChild>2</falseChild>
<conclusion>pippo_worklet
</conclusion>
Prosegue l'esplorazione seguendo il ramo FALSE
<id>2</id>
<condition>
pluto = true
</condition>
<trueChild>-1</trueChild>
<falseChild>3</falseChild>
<conclusion>pluto_worklet
</conclusion>
pippo_worklet
TrueChild è -1 quindi non c'è nessun'altra condizione da valutare. Viene restituita la conclusione.

L'albero delle regole



L'albero delle regole

Nell'esempio, se il controllo delle condizioni dell'albero delle regole porta ad una foglia, allora il task "Trattamento del sintomo" viene sostituito dalla worklet corrispondente.

Altrimenti il task viene eseguito normalmente come tutti gli altri.

Worklet Rules Editor

YAWL mette a disposizione un editor di regole.

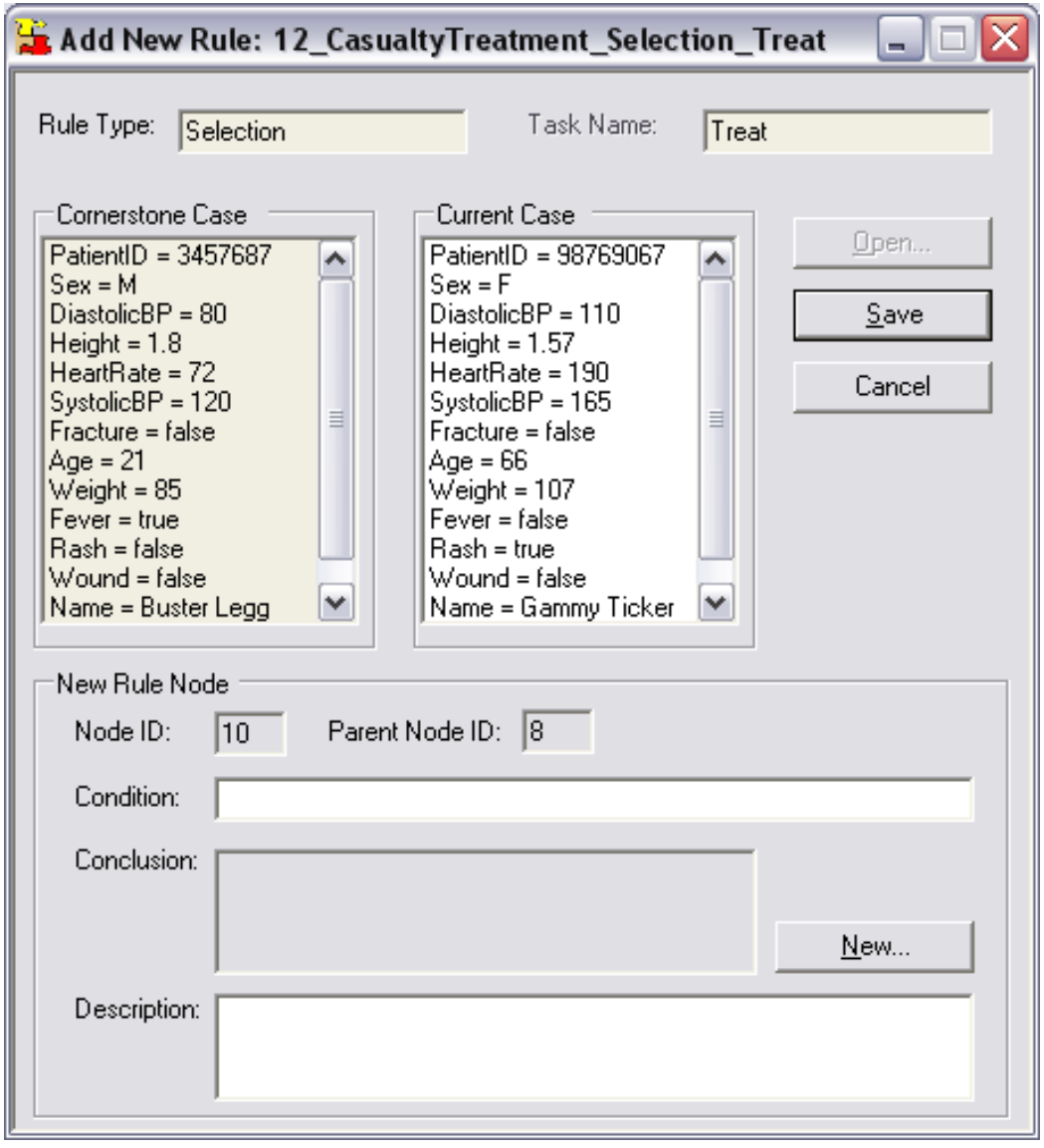
Tale editor è però al momento disponibile solo per Microsoft Windows.

Vediamolo comunque velocemente.

Creare una nuova regola

La nuova regola può essere definita sfruttando i valori che divergono tra il Case standard e il Case corrente.

Dettagli della nuova regola: costruisco la Condition in base ai valori che vedo nella Current Case e che divergono da quelli del Case standard.



Creare una nuova regola

La nuova Condition che creo, deve solo contenere le variabili (una o più), tra quelle che divergono rispetto al Case standard.

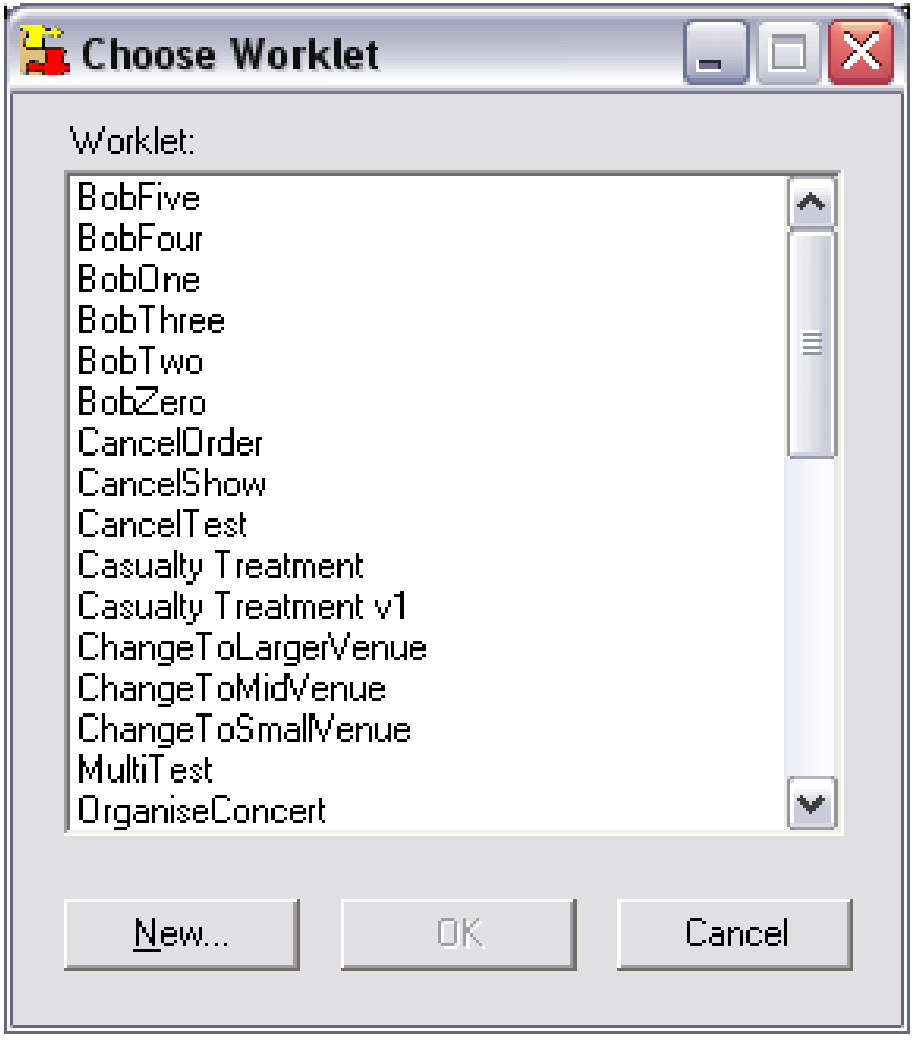
Le Condition sono stringhe di operandi (variabili) e operatori logici, con le parentesi per delimitare le sotto-espressioni.

Precedenza	Operatore logico	Tipo di operatore
1	$*$ $/$	aritmetici
2	$+$ $-$	
3	$=$ $>$ $<$ $!=$ $>=$ $<=$	comparazione
4	&	AND logico
	|	OR logico
	!	NOT logico

Creare una nuova regola

Conclusion è la sezione della regola nella quale si definisce quale worklet debba essere eseguita se il valore della Condition risulterà vera.

NB: Quando si definisce una nuova worklet ricordare che per passare i dati dal workitem alla worklet e viceversa, è necessario che i nomi e i tipi delle variabili coincidano!



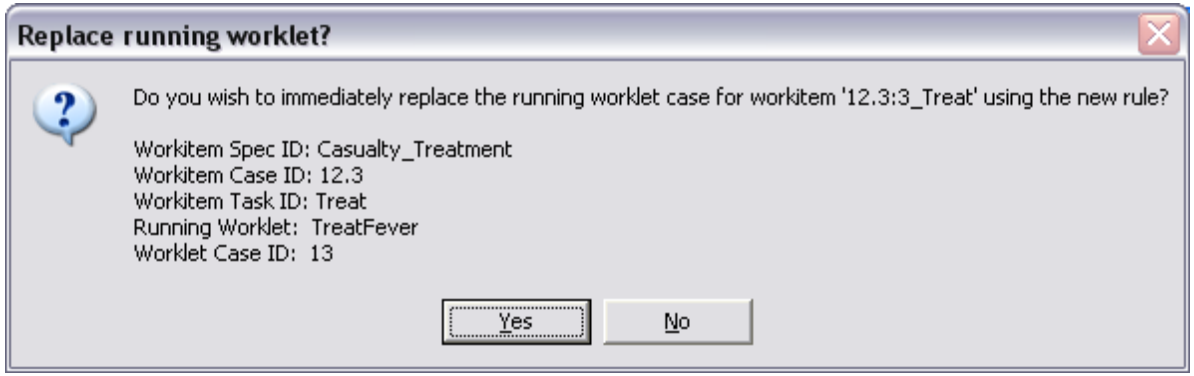
Creare una nuova regola

È possibile anche selezionare più worklet per la stessa Condition.

In quel caso, quando la regola è invocata in un processo, TUTTE le worklet presenti nella Conclusion partono simultaneamente e il processo continua solo quando TUTTE le worklet lanciate sono state terminate.

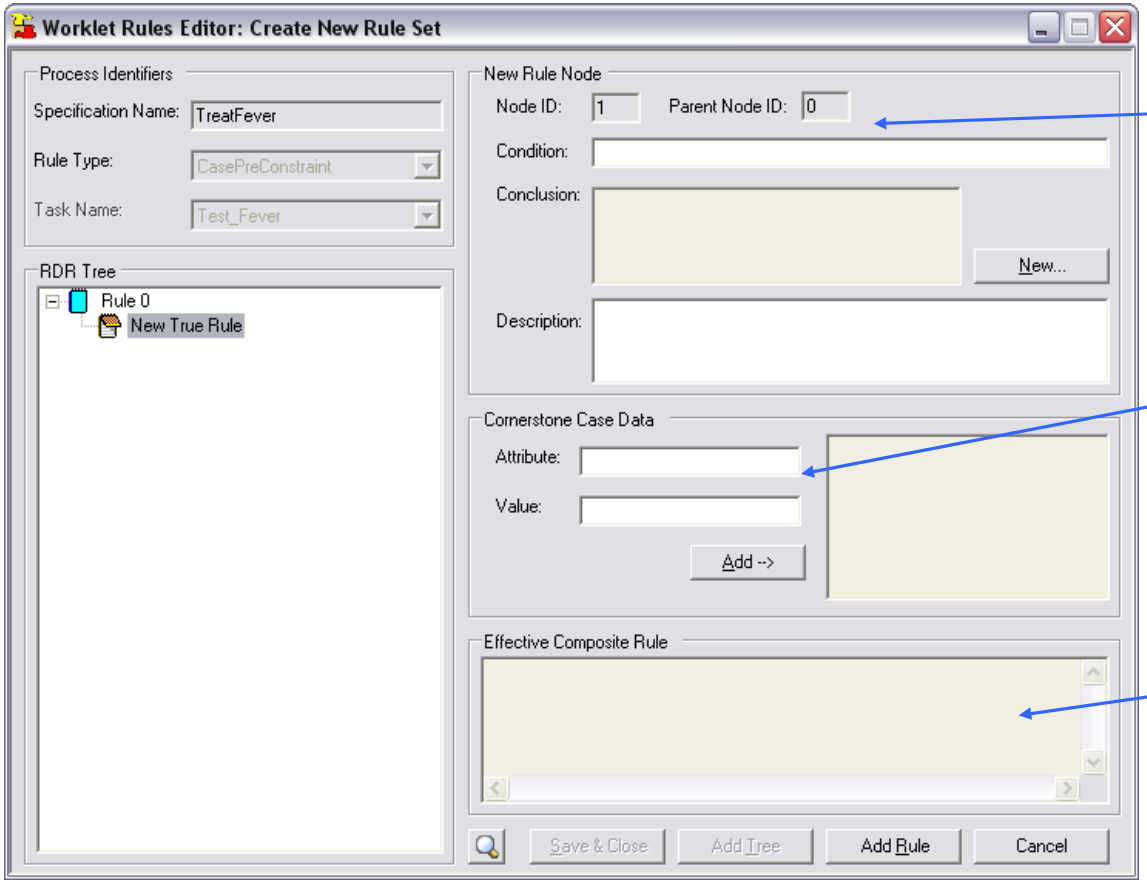
Applicazione istantanea di una nuova regola

Quando viene creata una nuova regola, il sistema chiede se si vuole applicare tale regola anche alle istanze del workflow che sono attualmente in esecuzione.



In tal caso, il Rules Editor contatta direttamente il Worklet Service andando ad aggiornare la modifica richiesta

Creare un nuovo albero di regole



Inserisco le regole nello stesso modo visto prima

Definisco le variabili e i valori di input relativi al Case

Definisco la condizione globale che indica se il nodo selezionato può essere considerato concluso

Esercizio 5

Implementare un workflow per gestisce l'assistenza sanitaria in un pronto soccorso.

Un paziente viene ricoverato nel reparto appropriato dopo averne esaminato i sintomi e stabilito la priorità di intervento.

Utilizzare la funzionalità delle worklet per modellare il meccanismo di priorità nella scelta del reparto di degenza.

Esercizio 6

Implementare un workflow per la gestione del servizio tecnico per il sistema informatico di un ateneo.

Il SIA riceve segnalazioni di mal funzionamento da parte degli utenti. In base alla priorità stabilita in termini di importanza del servizio (in ordine dal più importante: noRete, mailServer, laboratorio, ftpServer), viene attivato il processo di risoluzione appropriata.
Alla fine viene generato il report dell'intervento eseguito.

Utilizzare la funzionalità delle worklet per modellare il workflow.

Sommario

Cos'è un'eccezione

Eccezione:
Nell'ambito dei workflow, un'eccezione non è necessariamente un problema o un errore, ma semplicemente è un'attività non prevista in fase di progettazione del workflow.

Il Worklet Exception Service sfrutta i vantaggi di flessibilità dinamica ottenuta grazie all'uso delle worklet, per dare supporto alla miriade di eccezioni che possono accadere durante l'esecuzione di un processo.

Cos'è un'eccezione

Le worklet sono usate per modellare quelle attività che devono essere lanciate in caso di eccezioni.

Come nel Worklet Selection Service, è necessario definire le regole per permettere al YAWL Engine di far corrispondere la corretta worklet alla situazione contingente in atto.

In più, per ogni regola è possibile stabilire altre caratteristiche utili per la gestione avanzata delle eccezioni (ferma, cancella, riparte il task o l'intero workflow, ecc. . .).

Tipi di eccezioni

- Eccezioni causate da vincoli disattesi:

  I vincoli (constraint) sono regole applicate al workitem o al case PRIMA o DOPO della loro esecuzione

  1. **CasePreConstraint**: regola controllata prima dell'inizio dell'esecuzione dell'istanza del workflow
  2. **ItemPreConstraint**: regola controllata prima dell'inizio dell'esecuzione del task
  3. **ItemPostConstraint**: regola controllata dopo la conclusione dell'esecuzione del task
  4. **CasePostConstraint**: regola controllata dopo la conclusione dell'esecuzione dell'istanza del workflow

Tipi di eccezioni

- Eccezioni causate da eventi esterni:

  Quando accade qualcosa di esterno al processo ma che ne influenza l'esecuzione si usano le seguenti eccezioni:

  5. **CaseExternalTrigger**: a livello di istanza del workflow;
  6. **ItemExternalTrigger**: a livello di singolo task.

  Solitamente queste eccezioni sono scatenate dagli utenti, per avvisare il sistema che qualcosa di imprevisto è accaduto.

Tipi di eccezioni

7. Eccezione causata da **TimeOut**: un evento di timeout accade quando un task è collegato al YAWL Time Service, e una deadline relativa al task è stata violata.
8. Eccezione causata da annullamento di un task (**ItemAbort**): quando un task è eseguito da un servizio esterno e tale servizio deve annullare l'esecuzione in corso del task.
9. Eccezione causata da risorsa non disponibile (**ResourceUnavailable**): eccezione causata quando non si riesce ad allocare un task ad alcuna risorsa disponibile, tra quelle ammesse per quel task.
10. Eccezione causata da violazione di vincolo (**ConstraintViolation**): eccezione che deve gestire il caso in cui un vincolo venga violato DURANTE l'esecuzione del task (non prima o dopo come per i ItemPreContraint e ItemPostContraint)

Tipi di eccezioni

NB: Non è necessario prevedere delle attività che gestiscano tutti i tipi di eccezioni.

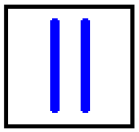
Ad ogni evento di eccezione, il YAWL Engine controlla nelle regole quali attività devono essere avviate e, se non ne trova, semplicemente ignora l'eccezione.

Primitive per modellare eccezioni

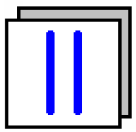
Per modellare le attività che devono gestire le eccezioni si deve definire la sequenza di passi (primitive) che devono essere svolti per superare la criticità in corso.

La sequenza dei passi può essere creata graficamente grazie al Worklet Rules Editor.

Primitive per modellare eccezioni



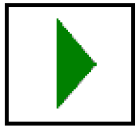
Sospensione del workitem: mette in pausa l'esecuzione del task finchè non saranno richiamate altre azioni sul task (annulla, riparti, ricomincia, forza a completato ecc. . .) o azioni sul suo workflow (annullalo, forzalo a completato ecc. . .).



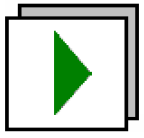
Sospensione del case: sospende tutti i task in corso (quelli già allocati dalla risorsa e quelli in esecuzione) e poi sospende l'istanza del workflow.

Sospensione di tutti i case: sospende tutti i task in corso di tutte le istanze di workflow in esecuzione che contengono il task in questione.

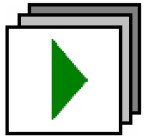
Primitive per modellare eccezioni



Continua il workitem: fa ripartire un task che era stato precedentemente sospeso.

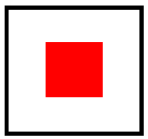


Continua il case: fa ripartire tutti i task sospesi di un'istanza del workflow che era stata precedentemente sospesa.

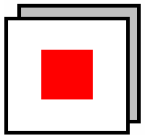


Continua tutti i case: fa ripartire tutti i task sospesi di tutte le istanze del workflow, precedentemente sospese, nelle quali è presente il task in questione.

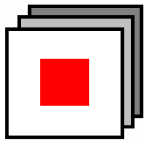
Primitive per modellare eccezioni



Rimuovi il workitem: annulla il task; l'esecuzione finisce e al workitem è assegnato lo stato di Cancelled. Non viene più richiamata l'esecuzione di tale task nel proseguo dell'esecuzione del workflow.



Rimuovi il case: conclude l'esecuzione dell'istanza del workflow e la elimina.



Rimuovi tutti i case: conclude l'esecuzione di tutte le istanze del workflow nelle quali il task è definito e le elimina.

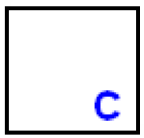
Primitive per modellare eccezioni

Riavvia il workitem: riavvia dall'inizio l'esecuzione del task. Azzera ai valori di default tutti i dati del workitem.

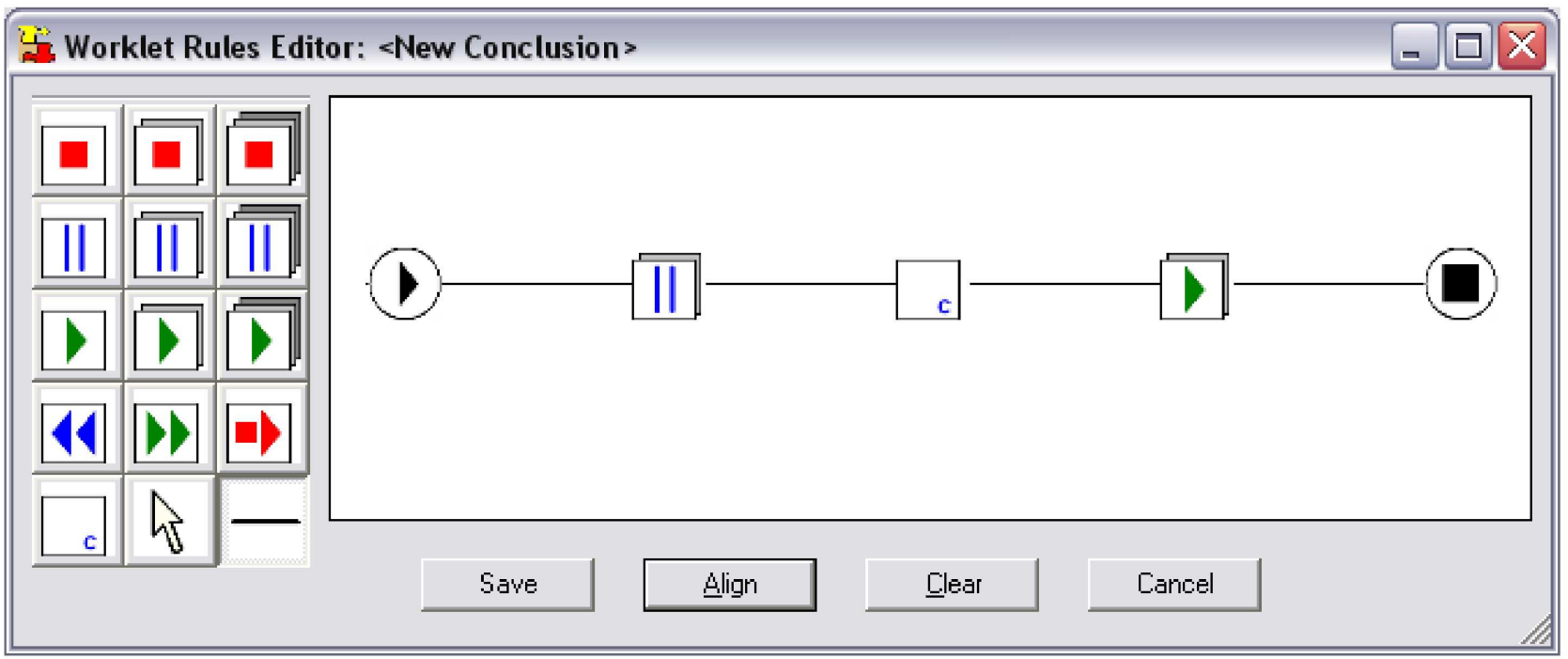
Forza il completamento del workitem: termina l'esecuzione del task e al workitem è assegnato lo stato di ForcedComplete. L'esecuzione prosegue con il task successivo nel workflow.

Forza il fallimento del workitem: termina l'esecuzione del task e al workitem è assegnato lo stato di Failed. L'esecuzione prosegue con il task successivo nel workflow.

Primitive per modellare eccezioni



Compensazione: esegue un processo di supporto (i.e. worklet). In base alle primitive precedenti, la worklet può essere eseguita simultaneamente alle altre attività del workflow, oppure svolgersi finchè il case è messo in pausa.



Esercizio 7

Implementare un workflow che gestisca in modo efficiente le prenotazioni allo stadio per un concerto.

In base al prezzo del biglietto e al numero dei biglietti venduti, il workflow deve assistere l'utente nel cercare una sede non troppo capiente (e costosa). Solo quando la sede trovata ha una capienza non troppo elevata, il workflow abilita l'esecuzione dello spettacolo.

L'utente imposta il costo per l'affitto dello stadio e la sua capacità di posti a sedere. Poi deve impostare il prezzo del biglietto e il numero di biglietti venduti. Se i biglietti venduti sono meno del 75% della capacità dello stadio, si cerca un palazzetto. Se, invece, i biglietti venduti sono meno del 50% della capacità dello stadio, si cerca un teatro. Se i biglietti venduti sono meno del 20% della capacità dello stadio si annulla lo spettacolo.