Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# Prototipo di un workflow complesso per l'esecuzione di ensemble di simulazioni in Sunflower

Giuseppe Papuzzo, Danilo Cistaro

**RT-ICAR-CS-11-08** **Dicembre 2011**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

# LABORATORIO PUBBLICO PRIVATO OPENKNOWTECH

Progetto di Ricerca**: DM21301**

**OpenKnowTech "*Laboratorio di Tecnologie per la Integrazione, Gestione e Distribuzione di Dati, Processi e Conoscenze*"**

## Obiettivo Realizzativo OR2

***Open Grid***

**Deliverable D2.3**

# ALLEGATO 1 D2.3_ALL_AT2.4.a

Prototipo di un workflow complesso per l'esecuzione di ensemble di simulazioni in Sunflower

## Informazioni sul documento

| | |
| --- | --- |
| Versione: | **1.0** |
| Data completamento: | |
| Preparato da: | |
| Approvato da: | |
| Stato | **FINALE** |

# Sommario

# 1 Introduzione

La necessità di ottenere in tempo reale i risultati della simulazione di un modello cip one di fronte al dilemma di utilizzare ono sistemi distribuiti ad alte prestazioni. Tali sistemi sono complessi da utilizzare e spesso necessitano di esperti. Inoltre l'elevata latenza delle comunicazioni fra i nodi, distribuiti geograficamente, può portare ad un'esecuzione inefficiente del programma. Con l'adozione di modelli basati su un *ensemble di simulazioni* eseguiti su una piattaforma distribuita attraverso *workflow scientifici* è possibile superare questi limiti.

L'adozione ormai comunemente accettata dell'architettura service-oriented per piattaforme distribuite, come ad esempio la Griglia o i sistemi per il Cloud computing rende ormai possibile strutturare una computazione distribuita in maniera semplice ed efficiente. L'architettura a servizi non è rivolta a supportare singole applicazioni che richiedono elevate potenze di calcolo, ma piuttosto a supportare un processo scientifico attraverso la cooperazione di servizi indipendenti. L'uso di workflow scientifici come strumento per supportare il processo scientifico è ormai diventato una necessità. Un workflow scientifico a servizi consente di definire un modello di elaborazione distribuita in cui sono presenti un insieme di servizi, definire l'ordine in cui i servizi devono essere eseguiti, gestire il trasferimento dei dati fra i servizi, etc. Una volta predisposto il workflow è necessario provvedere a fare il suo deployment e monitorarne il comportamento. Attraverso il workflow è quindi possibile ottenere una descrizione del processo che un ricercatore deve eseguire per ottenere il risultato. Inoltre, queste operazioni vengono eseguite automaticamente, senza la necessità di alcun intervento umano.

Una caratteristica fondamentale degli attuali motori per la gestione di workflow è che essi sono molto rigidi. Infatti, essendo basati su un modello cliente-servente non sono in grado di supportare il modello a coreografia per l'esecuzione di workflow. La mancata esecuzione del workflow secondo un modello a coreografia, che è basato sull'adozione di un pardigma P2P, non consente di supportare né il parallelismo a *livello dei servizi,* che utilizza algoritmi di scheduling basati sul rispetto della QoS, né il parallelismo *flow-level* che prevede l'esecuzione distribuita su multipli peer delle parti in cui il workflow può essere suddiviso adottando il modello a coreografia.

Per simulare il modello per la distribuzione delle auto mobili nel piazzale definito nell'OR3 abbiamo adottato uno schema di workflow scientifico che consente di eseguire esperimenti di sensitività attraverso l'esecuzione di un *ensemble di simulazioni* con parametri variabili. L'idea è quella di costruire un workflow complesso che ci permetta, in parallelo, di instanziare molteplici simulazioni del modello a cui sono forniti gli stessi dati di input riguardanti il piazzale e il lotto delle automobili da allocare ma differenti parametri che riguardano i valori con cui inizializzare l'algoritmo di simulazione. L'esecuzione del workflow sulla piattaforma distribuità avrà il vantaggio di svincolare l'utente finale dalla esecuzione manuale degli esperimenti permettendogli di generare in maniera efficiente un insieme di soluzioni fra cui scegliere la più appropriata.

Per superare i limiti degli attuali motori di workflow il nostro workflow utilizza il framework SunFLOWer, messo a punto in un'attività di ricerca precedente del progetto. L'adozione del sistema SunFLOWer, per le sue caratteristiche di autonomicità permette di soddisfare i requisiti necessari per un'esecuzione efficiente, affidabile e dinamica di

workflow complessi necessari per supportare l'esecuzione di un ensemble di simulazioni. Secondo questo modello è richiesta l'esecuzione di simulazioni multiple costituite da algoritmi esatti o euristici che vengono eseguiti con differenti condizioni iniziali.

Attraverso SunFLOWer, il modello ensemble sarà eseguito come un workflow scientifico sulla Griglia garantendo i seguenti benefici:

- esecuzione parallela, su differenti peer, dell'ensemble di simulazioni per la distribuzione delle automobili nel piazzale;
- scheduling e allocazione dei servizi sui vari nodi della piattaforma tenendo in conto dei parametri di QoS in modo da bilanciare il carico sui nodi;
- utilizzo delle capacità di failure-aware di SunFLOWer per gestire la dinamicità dei workflow.

Il modello per la distribuzione delle automobili nel piazzale eseguito attraverso un workflow scientifico consentirà di eseguire *esperimenti di sensitività* attraverso un'esecuzione iterativa dell'ensemble. Nel workflow ogni servizio per la simulazione è avviato adottando un insieme di parametri variabili appartenenti a un range prespecificato dall'utente che consente di avviare un'elaborazione del tipo *trial and error*. L'idea è quella di costruire un workflow complesso che ci permetta di eseguire in parallelo differenti versioni del modello e sia in grado di fornire le migliori soluzioni sfruttando efficientemente la piattaforma distribuita tenendo in conto la sua dinamicità. Inoltre, è necessario considerare la possibilità che potendosi verificare guasti nei nodi è necessario ripristinare on-line l'elaborazione in modo da non perdere il lavoro svolto ma possibilmente continuare l'elaborazione usando gli altri nodi disponibili della piattaforma.

L'esecuzione del workflow su una piattaforma distribuità avrà il vantaggio di svincolare l'utente finale dall'esecuzione manuale degli esperimenti permettendogli di concentrarsi sulla fase di analisi dei risultati di allocazione.

Il resto del documento è così organizzato. Nella sezione 2 viene definito il problema della distribuzione delle automobili nel piazzale. Successivamente sono illustrati i requisiti necessari per l'esecuzione del workflow e come essi sono soddisfatti dal framework SunFLOWer. Infine, è presentato uno schema di workflow per supportare l'ensemble e la sua realizzazione in SunFLOWer e i risultati sperimentali ottenuti.

# 2 Il problema della distribuzione di automobili

I terminali marittimi per autovetture sono caratterizzati da tematiche di gestione che solo in parte coincidono con quelle tipiche dei terminali per contenitori.

In particolare, il fatto che le autovetture, sia in fase di imbarco che di sbarco, richiedano per la loro movimentazione la presenza della figura professionale del guidatore (driver) introduce una significativa differenza, in quanto è necessaria la predisposizione di un servizio navetta, che affianchi le attività dei guidatori. Tale servizio, infatti, deve occuparsi di garantire la mobilità dei guidatori stessi nelle fasi di rientro, dopo aver eseguito un'operazione di spostamento di una vettura da o verso il piazzale.

Un'efficiente gestione delle operazioni di movimentazione è, quindi, strettamente connessa alla modalità di stoccaggio delle autovetture in piazzale. In particolare è considerata appropriata e utile una distribuzione delle stesse che sia di tipo concentrato, piuttosto che distribuito. E' infatti abbastanza intuitiva l'idea che sia più facile gestire i movimenti a vuoto dei driver se le auto non sono disposte in piazzale in maniera troppo sparsa.

Le attività di housekeeping hanno quindi un ruolo importante, sono tipicamente eseguite durante gli intervalli di tempo lasciati liberi dalle operazioni di carico-scarico e sono finalizzate proprio a garantire una configurazione di piazzale compatta. Diversamente dai container, le automobili sono oggetti "fragili" che richiedono maggior attenzione nella loro gestione. Per esempio le automobili non possono essere messe una sull'altra, sono quindi necessarie aree più estese in confronto a quelle utilizzate nei terminali per container. Le automobili sono organizzate in gruppi omogenei per modello e marca che arrivano e partono con la stessa copia di navi. Il compito del *planner* è di assegnare dinamicamente i gruppi di automobili in arrivo a righe del piazzale. Una volta assegnate, le automobili non sono più spostate all'interno del piazzale, quindi la posizione assegnata inizialmente non cambia per tutto il periodo di permanenza. Questo perché ogni spostamento aumenta il rischio di danneggiare l'automobile e questo rischio deve essere mantenuto più basso possibile. La gestione del piazzale è quindi un'attività di cruciale importanza e la distanza totale che le automobili devono percorrere ne rappresenta un aspetto critico. Per facilitare la gestione del piazzale un gruppo di automobili è allocato su un insieme di righe adiacenti. Il numero di righe richieste dipende dalla lunghezza delle automobili e dalla lunghezza della riga dato che le righe possono essere di lunghezza diversa. E' possibile quindi, la presenza di righe occupate parzialmente.

# 3 Requisiti e gestione del workflow

I modelli per la distribuzione delle automobili nel piazzale sono basati su complesse tecniche di ottimizzazione e sono programmi per computer complicati che richiedono una grande potenza di calcolo. Alcuni di questi potrebbero essere parallelizzati. Tuttavia, in questa sede non siamo interessati a questo aspetto, che è comunque estremamente interessante, ma all'utilizzo di un modello distribuito, conosciuto come "ensemble", che facendo uso di un grande numero di servizi di simulazione sia in grado di valutare l'incertezza intrinseca della simulazione individuando quali sono i parametri che forniscono le soluzioni migliori.

Modelli basati su ensemble di simulazioni con parametri variabili sono usati principalmente per eseguire esperimenti volti a studiare la *sensitività* del modello. Nell'ensemble ogni simulazione è indipendente dalle altre e può essere eseguita asincronicamente. Questo modello essendo costituito da job parametrici è particolarmente adatto per essere eseguito su piattaforme distribuite, come ad esempio la Griglia, poiché ogni simulazione può essere eseguita su differenti nodi e i risultati possono essere resi disponibili con un insieme di data set uniformi in un catalogo accessibile attraverso una interfaccia Web pronti per essere analizzati.

Il modello può essere facilmente supportato da un workflow a servizi ed eseguito su una Griglia. A causa della dinamicità della Griglia, ci sono però parecchie ragioni che possono causare il fallimento di una o più simulazioni durante un test di prova. Questo comportamento, induce i ricercatori ad utilizzare il modello con qualche cautela poiché in caso di guasto si perderebbero parecchie ore di lavoro in quanto i motori attuali per la gestione di workflow non sono dotati di meccanismi di recovery della computazione.

La necessità di disporre di motori per la gestione di workflow che siano capaci di far ripartire l'elaborazione anche in caso di guasto necessita di gestori di workflow *goal-oriented* e *failure-aware* che siano in grado di eseguire gli esperimenti con un minimo intervento umano lasciando all'utente solo il compito di analizzare i risultati. Pertanto, per effettuare la nostra sperimentazione ci avvarremo del framework SunFLOWer per la gestione di workflow autonomici le cui funzionalità sono brevemente richiamate nel successivo paragrafo.

## 3.1 *SunFLOWer*

I sistemi per la gestione di workflow (WMS) combinando le tecnologie di Griglia e di workflow offrono un sofisticato supporto per la costruzione di workflow scientifici dove è definita la descrizione del processo che un ricercatore deve eseguire per creare un "output scientifico".

La principale idea dell'approccio workflow è quella di separare la descrizione del processo di workflow o specifica dal sistema responsabile della sua esecuzione. L'esecuzione di workflow scientifici include l'elaborazione della specifica del workflow e l'uso coordinato delle risorse appropriate. Tre livelli sono comunemente identificati per la descrizione dei workflow:

1. **workflow astratti**: A questo livello di astrazione il workflow contiene le informazioni circa cosa deve essere fatto da ogni task e le informazioni su come i task sono interconnessi. A questo livello non è specificato come i task/servizi sono implementati.
2. **workflow concreti:** Un workflow concreto annota ogni task con informazioni sull'implementazione e/o le risorse usate. Le informazioni riguardano il metodo di invocazione e il formato dei dati reali che sono scambiati.
3. **istanze di workflow:** riguarda il mapping reale del workflow concreto sulle risorse computazionali, definendo i dati di input specifici e i corrispondenti risultati.

Un WMS deve tenere conto della *complessità*, *eterogeneità* e *dinamismo* della Griglia e adottare nella progettazione paradigmi innovativi come quelli della computazione autonomica e dei sistemi multi-agente per definire un sistema scalabile, adattativo e capace di auto-gestirsi. Incorporando le proprietà della computazione autonomica, un WMS è in grado di riconfigurarsi dinamicamente per adattarsi ai cambiamenti dell'ambiente di Griglia e soddisfare i requisiti di performance, affidabilità, sicurezza, qualità del servizio (QoS), senza interventi manuali.

Il framework SunFLOWer, sviluppato nell'ambito dell'OR3 del progetto AUTOMA, fornisce una piattaforma per supportare la configurazione, l'enactment, la gestione e l'adattività di Grid workflow capaci di auto-configurarsi dinamicamente in risposta a eventi generati dall'ambiente di Griglia e garantire il mantenimento di determinati livelli di prestazioni in contesti elaborativi caratterizzati da dinamicità nell'uso delle risorse e delle richieste di servizio.

Sunflower è basato sull'utilizzo di una piattaforma P2P su cui sono attivi un insieme di *workflow engine* che sono enacted come una federazione di agenti auto-organizzanti che interagiscono fra loro e con il sistema informativo di Griglia progettato con l'ausilio di algoritmi bio-ispirati per riorganizzare dinamicamente i descrittori che descrivono le risorse di Griglia che sono disponibili in modo da facilitare le operazioni di gestione e scoperta delle risorse e la loro allocazione.

Sunflower si propone quindi come una piattaforma innovativa per supportare la composizione dinamica di servizi su Griglia in grado di :

- gestire in maniera decentralizzata l'esecuzione di Grid-workflow con evidenti vantaggi per l'esplicitazione automatica delle forme di parallelismo presenti nel workflow e la fault tolerance;
- effettuare automaticamente il deployment dinamico del workflow tenendo conto dei parametri di QoS specificati dagli utenti;
- utilizzare tecniche basate sull'auto-organizzazione dei sistemi multi-agente per garantire l'autonomicità dei Grid workflow. Sunflower aggiunge capacità di self-tuning e self-configuration a un motore distribuito di workflow in modo da determinare la configurazione automaticamente prendendo in considerazione misure delle performance basate sul workload reale.

- usare, in maniera ottimizzata, i Grid/ Web service attraverso tecniche di selezione QoS-aware. I servizi sono estesi con requisiti non-funzionali quali quelli della performance, della scalabilità e della fault-tolerance.

le funzionalità caratterizzanti SunFLOWer sono:

- **Enactment decentralizzato di workflow**: il framework adotta un approccio basato sul modello a coreografia che sostituisce l'engine centralizzato e monolitico di BPEL con un insieme di agenti (engine) distribuiti e cooperanti in ambiente P2P capaci di auto-coordinarsi.
- **Adattabilità**: il workflow si adatta automaticamente per rispondere alle variazioni di workload del nodo di griglia o all'indisponibilità di uno o più servizi.
- **Deployment dinamico**: Il WMS è in grado di riorganizzare dinamicamente l'allocazione dei servizi al workflow (*binding dinamico*) in modo da consentire un'esecuzione quasi ottimale (*self-tuning*) del workflow senza dover riavviare l'intero sistema;
- **Self-managing e autonomicità:** Il WMS oltre che essere decentralizzato è dotato di proprietà autonomiche, che gli consentono di auto-configurarsi, auto-ripararsi, etc. senza che vi sia necessità di interventi manuali.

Una delle caratteristiche interessanti di SunFLOWer per la gestione di workflow scientifici riguarda la sua capacità di gestire workflow dinamici in grado di essere eseguiti e riconfigurati attraverso un processo adattativo.
Pensiamo ad esempio al metodo scientifico con cui i ricercatori eseguono le loro ricerche. Il metodo può essere descritto come un processo iterativo che implica i seguenti passi: identificazione del problema, definizione di un'ipotesi, conduzione di un esperimento, valutazione dei risultati e raggiungimento di una conclusione. Un punto chiave di questo metodo è il testing delle ipotesi attraverso la sperimentazione e l'osservazione. Formulando gli esperimenti come workflow scientifici è possibile definire un primo workflow per la descrizione di un esperimento e successivamente è possibile testarlo con differenti combinazioni di parametri continuando finché il processo adattativo non abbia trovato una soluzione adeguata. In questo modo, è possibile partire da un workflow standard per convergere verso un workflow finale attraverso un processo iterativo di esplorazione, spesso in maniera *trial and error*. Questo metodo porta a sviluppare differenti versioni dello stesso workflow che si modificano come la ricerca procede.
La dinamicità è quindi un requisito fondamentale per la realizzazione dello schema di workflow necessario per supportare il modello distribuito di tipo ensemble che coordina l'esecuzione di multiple simulazioni.

## 3.2 *La struttura del modello per l'ensemble di simulazioni*

In questa sezione introduciamo brevemente i differenti componenti coinvolti in una tipica simulazione per la distribuzione di automobili. Definiamo un **Esperimento** un ensemble di simulazioni (job parametrici) progettati per fornire l'allocazione delle automobili sul piazzale (una singola esecuzione è l'esperimento più semplice): ognuna di queste esecuzioni è chiamata **Realizzazione** e richiede un insieme di dati in ingresso per eseguire il modello nel periodo di simulazione prescritto (tipicamente un giorno). Un tipo particolare di esperimenti sono quelli relativi agli studi di

sensitività del modello. In questo caso i differenti parametri di ingresso sono ottenuti da un set di base a cui sono apportate perturbazioni secondo una tecnica definita dall'utente che consenta di esplorare in maniera intelligente lo spazio dei parametri.

L'elemento necessario per la definizione di un workflow del modello ensemble è un Web/Grid service che racchiude l'algoritmo necessario per l'esecuzione di una simulazione. In generale una *Realizzazione* richiede diversi web service che devono essere completati. Un Web service esegue il suo algoritmo finché il suo criterio di terminazione non è verificato o una soluzione è trovata. La soluzione trovata è memorizzata in un file. Quando tutti i Web service sono terminati producendo o no la soluzione, attraverso l'interfaccia grafica è possibile analizzare le soluzioni in modo da selezionare la migliore. Nel caso si verifichi un guasto in uno dei nodi della Griglia, SunFLOWer provvede a far continuare la simulazione del nodo guasto su un altro nodo dove è istanziata una copia equivalente del servizio necessario per la simulazione. E' possibile anche considerare la possibilità di definire parametri di QoS per ogni servizio che rappresenta una simulazione, in modo da ottenere un bilanciamento del carico automaticamente. SunFLOWer, provvederà a sostituire un servizio che non rispetti la QoS stabilità con uno equivalente disponibile su un altro facendo migrare anche la parte di workflow che utilizza il servizio.

Lo schema di workflow implementato per il progetto Automa è mostrato nella figura 1. In esso si prevede il wrapping come Web Service degli algoritmi euristici descritti precedentemente. Tali servizi sono invocati (invoke) ed eseguiti in parallelo (flow) su macchine distinte, il blocco parallelo è rieseguito (while) fino a che il tempo di esecuzione imposto dall'operatore umano non si esaurisce. Dopo la prima iterazione, ogni volta che il while è eseguito, vengono tarati i parametri degli algoritmi euristici in base a un processo selettivo che premia i parametri delle esecuzioni che si sono comportate meglio. Questa strategia aumenta la probabilità delle euristiche di individuare una soluzione, infatti, dopo alcuni cicli di while in cui si tenta in parallelo una serie di strategie, gli algoritmi si concentrano sempre in parallelo su un determinato tipo si soluzione. Naturalmente, questo tipo di strategia porta a individuare un insieme di soluzioni ammissibili, che a ogni ciclo si avvicinano più o meno rapidamente alla soluzione ottima. Se il workflow fosse libero di iterare portereb be sicuram ente alla so luzione ottima, ma ciò richiederebbe cert amente un t empo molto el evato. Il numero di cicli di while da eseguire è deciso indirettamente dell'operatore umano, che indica al workflow il tempo che ha disposizione per l'esecuzione. Durante tale periodo, dall'insieme di tutte le soluzioni trovate, sono eliminate a ogni ciclo le so luzioni peggiori. Al termin e d ell'esecuzione il wo rkflow fornirà co me o utput l'in sieme d elle migliori soluzioni individuate.

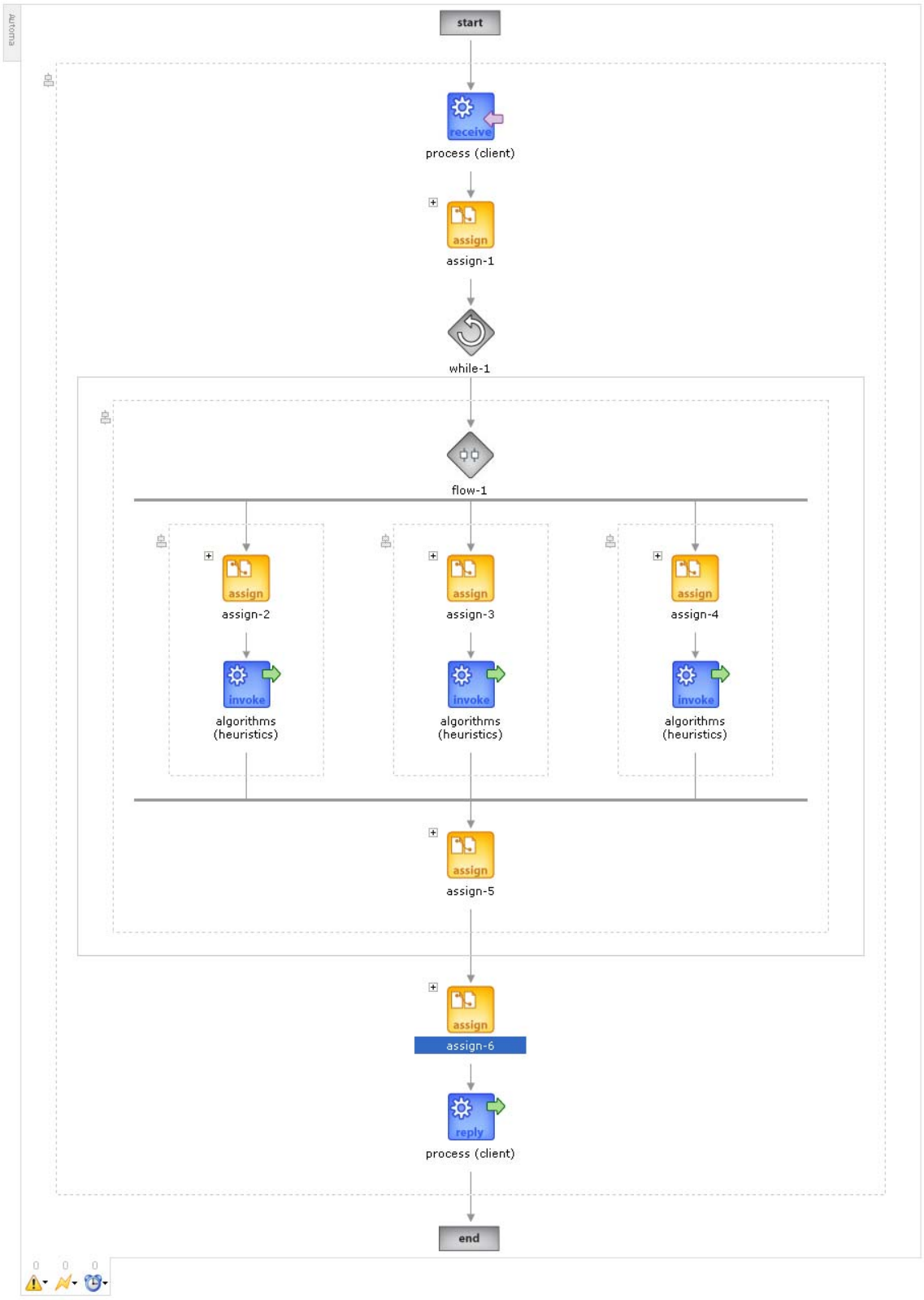


Fig. 1: Workflow implementato.

L'inizializzazione del workflow avviene mediate l'interfaccia grafica fornita dalla Sunflower Console (vedi Fig. 2), dalla quale è possibile:

- Eseguire l'upload del file contenente l'allocazione attuale del piazzale (Piazzale.csv).
- Eseguire l'upload del file contenente i lotti da allocare (Lotti.csv).
- Impostare il tempo di esecuzione del workflow.
- Specificare la QoS minima che i servizi devono rispettare, per garantire l'assenza di colli di bottiglia nel flow e massimizzare il numero di cicli di while nel tempo imposto.

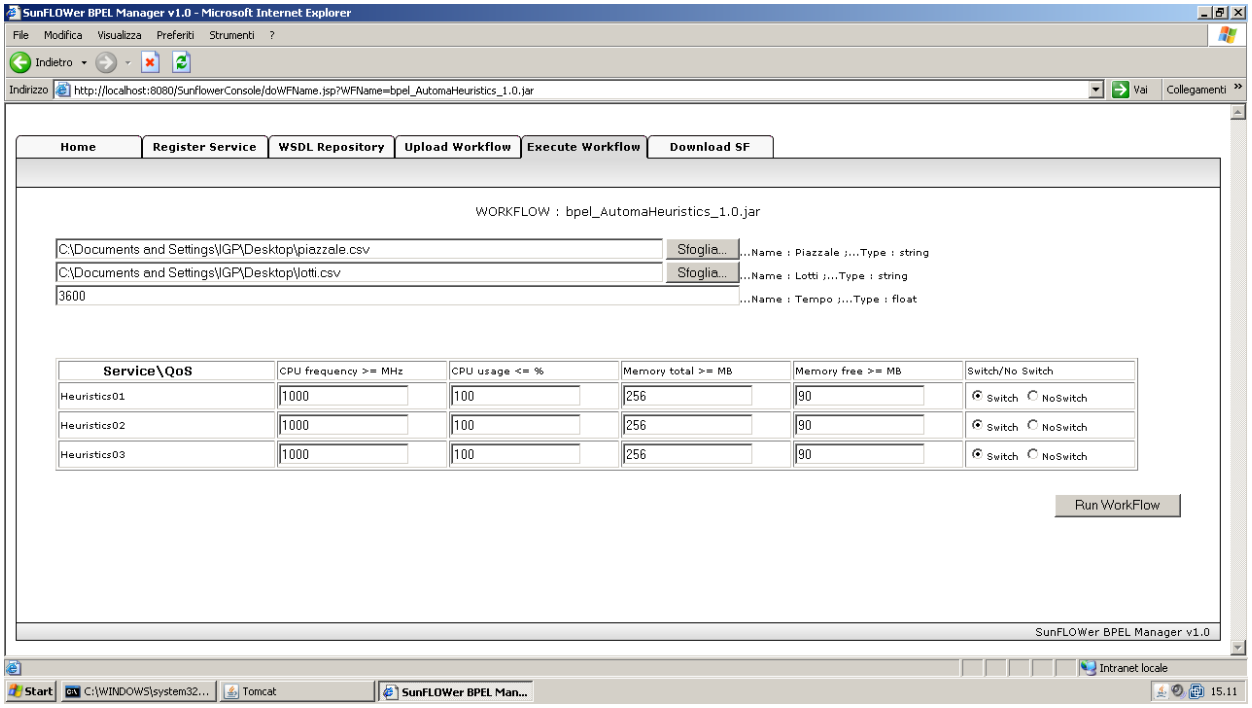


Fig. 2: Inizializzazione di un'istanza di workflow.

L'esecuzione dell'istanza di workflow può essere monitorata attraverso la finestra di monitoring (vedi Fig. 3) che appare al termine dell'inizializzazione dei parametri.

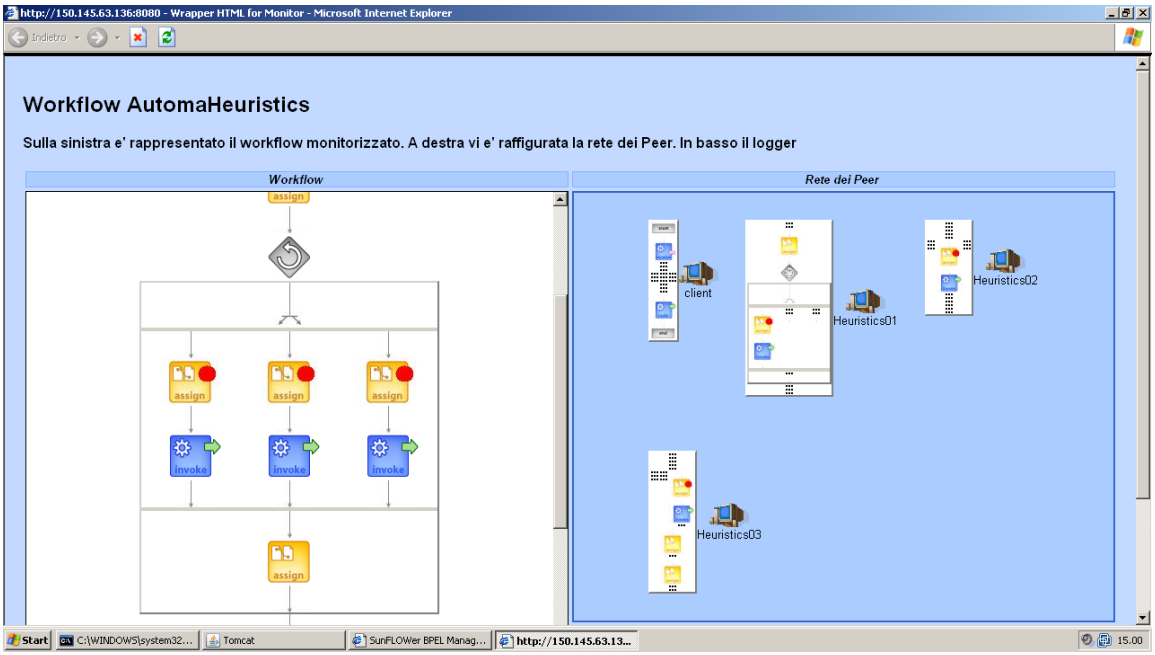


Fig. 3: Monitoring dell'esecuzione decentralizzata.

Al termine dell'esecuzione, le soluzioni individuate dalle diverse euristiche sono proposte mediante interfaccia grafica (vedi Fig. 4) all'operatore umano, il quale sceglierà quella che ritiene più opportuna in base alla sua esperienza e a vincoli non esprimibili matematicamente. Compiuta la scelta tra le soluzioni proposte, il sistema metterà a disposizione il nuovo file "Piazzale.csv" contenete il piano di lavorazione dei lotti da allocare. Inoltre tale file rappresenta la nuova allocazione attuale del piazzale che dovrà essere usato in caso di arrivo di altri lotti.

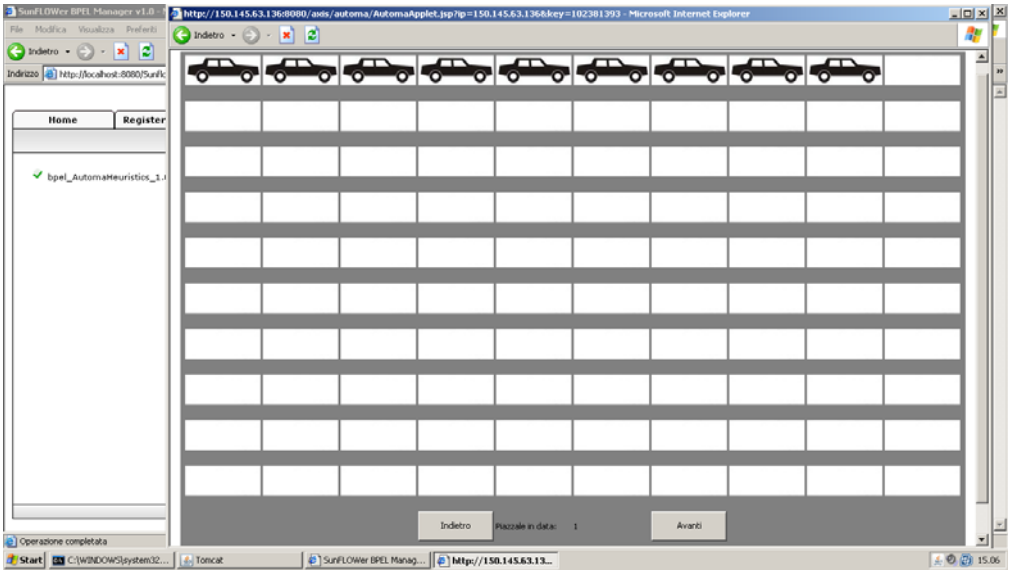


Fig. 4: Visualizzazione delle soluzioni individuate dalle diverse euristiche.

# 4 Implementazione e validazione

Lo scenario di validazione, del workflow implementato per il pro getto Automa, prevede l'uso degli algoritmi euristici per l'allocazione dei lotti di test su una ricostruzione virtuale del piazzale per lo stoccaggio delle auto.

Il piazzale (vedi fig. 4) è costituito da più buffer contigui organizzati in modo da formare tredici settori di dimensione variabile dove, per ogni buffer, sono memorizzati i metri che ogni auto deve percorrere per lo sbarco o l'imbarco rispetto a ogni banchina di ormeggio delle navi.

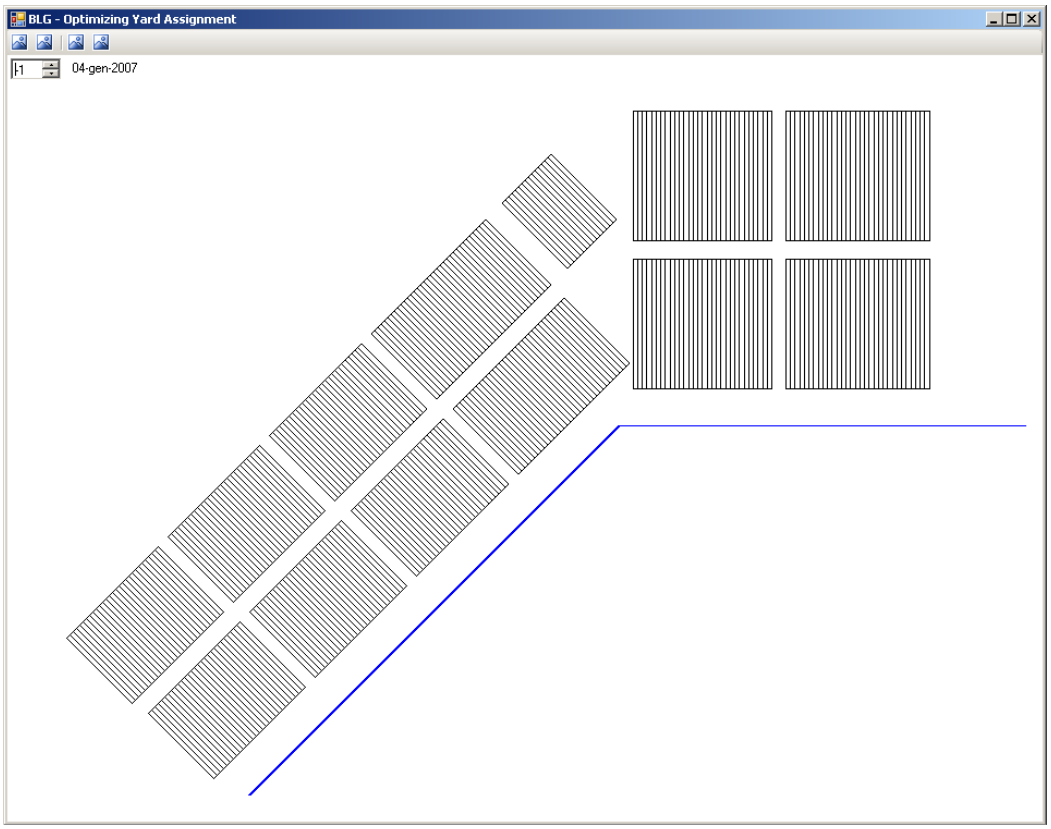


Fig. 4: Ricostruzione 2D del piazzale.

Per il test del sistema sono stati predisposti 12 diversi scenari, ognuno dei quali riproduce l'insieme dei lotti di auto che devono essere allocati in un periodo di 30 giorni. Dove, ogni lotto riporta i dati concernenti: la data e la banchina di arrivo, la data e la banchina di partenza, e ulteriori dati non rilevanti per la nostra analisi (modello ,marca , ecc. ).

La taratura degli algoritmi euristici avviene attraverso il settaggio di quattro parametri, che sono rispettivamente:

- Percentuale della soluzione attuale da utilizzare per calcolare la nuova soluzione.

- Tempo di esecuzione di una singola istanza di algoritmo euristico.
- Numero di strategie da utilizzare per la ricerca della soluzione.
- Probabilità di utilizzo della soluzione corrente

Grazie a tali parametri è possibile modellare il comportamento dell'algoritmo affinché si adatti ai diversi scenari; è possibile inoltre, imporre se cercare la nuova soluzione nell'intorno di quella corrente o andare ad analizzare un nuovo punto dello spazio di ricerca.

Predisposto il banco di prova per gli algoritmi euristici, si è provveduto ad eseguite il workflow su tutti i 12 scenari di test. Tali risultati, riportati nella tabella di figura 5, sono stati messi a confronto con le soluzioni ottenute utilizzando CPLEX (un software commerciale IBM) e con quelli ottenuti da un operatore umano che eseguia ripetutamente l'algoritmo variando i parametri in modo intuitivo. Le colonne della tabella riportano rispettivamente:

- Scenario: nome del test-set usato per la prova.
- CPLEX: risultato ottenuto dal software commerciale prodotto dall'IBM, che è stato usato come riferimento per la valutazione delle soluzioni euristiche.
- Operatore: variazione percentuale rispetto a CPLEX del risultato ottenuto da un operatore umano, il quale ha eseguito più volte l'algoritmo su ogni test-set, variando a ogni prova i parametri in base alle prove precedenti e al suo intuito.
- Workflow: variazione percentuale rispetto a CPLEX del risultato ottenuto eseguendo il workflow sui test-set.

| Scenario | CPLEX | Operatore | Workflow |
|---|---|---|---|
| test2_20_05A | 4112551,84 | -0,39% | -1,65% |
| test2_20_05B1 | 4703116,46 | 0,65% | 0,80% |
| test2_20_05B2 | 5164748,78 | -0,95% | 0,03% |
| test2_20_05B3 | 4971897,99 | -0,41% | -0,70% |
| test2_20_05B4 | 4589962,99 | -0,36% | 0,49% |
| test2_20_05B5 | 4975491,64 | 0,77% | 1,43% |
| test3_20_05A | 3987628,8 | 1,30% | 0,91% |
| test3_20_05B1 | 3796366,73 | -0,47% | -0,14% |
| test3_20_05B2 | 4710034,07 | 2,13% | 1,90% |
| test3_20_05B3 | 4082842,62 | 0,85% | 1,79% |
| test3_20_05B4 | 4439883,94 | 0,11% | 1,11% |
| test3_20_05B5 | 5064481,98 | 1,80% | 2,49% |

Fig. 5: Tabella riassuntiva dei risultati.

Come si può facilmente notare, sia l'operatore sia il workflow riescono a individuare una soluzione molto simile a quella proposta da CPLEX, ed addirittura migliorarla in alcuni casi. Quindi possiamo affermare che, a meno di piccoli scarti percentuali, mediamente gli algoritmi riescono a individuare soluzioni equivalenti.

A parità di soluzioni trovate, la strategia del Workflow presenta maggiori vantaggi rispetto alle altre due sia come tempi di esecuzione sia come semplicità di utilizzo. Infatti, CPLEX è un'applicazione stand-alone che richiede lunghi tempi di elaborazione, mentre l'Operatore deve ad ogni esecuzione dell'algoritmo verificare il risultato e individuare i nuovi parametri per rilanciare l'algoritmo. Al contrario, il Workflow è in grado di eseguire più euristiche in parallelo e ricalcolare i nuovi parametri in modo automatico senza ausilio di personale esperto, guadagnando notevolmente su tempi di esecuzione e semplicità di utilizzo.