ISTITUTO NAZIONALE DI FISICA NUCLEARE

Direzione Sistemi Informativi

INFN-23-17-DSI
4 maggio 2023

Un motore di Workflow per applicazioni web

Antonello Paoletti[1], Giancarlo Terilli[1], Claudio Ciamei[1], Francesco Serafini[1]

[1)] *INFN, Direzione Sistemi Informativi, I-00044 Frascati (Roma), Italy*

Abstract

Progettazione e sviluppo di un **motore di workflow** per processi gestionali informatizzati. Il termine *workflow* identifica il modello digitale di un processo di *business*[1] attraverso la sua rappresentazione in forma di *grafo orientato*[2]. Ogni *nodo* del grafo identifica un'azione ben definita, declinata in forma di passaggi decisionali, sottoprocessi e/o manipolazione dei dati di contesto, con l'obiettivo di supportare l'iter di un processo in maniera efficiente, misurabile e ripetibile. La definizione di un *workflow* implica sia aspetti statici che dinamici di un processo e ne modella il comportamento come un *automa a stati finiti*[3]. Ogni *collegamento* dell'automa stabilisce un collegamento fra due attività, definendo condizioni di *percorrenza* formalizzate come funzioni *booleane*. Tali funzioni sono attivate nel momento in cui un *attore* o un sotto-processo conclude le attività previste dalla fase attuale e vuole transire alla successiva. Emerge, a questo punto, la necessità di un *orchestratore* che governi l'andamento del *workflow*, garantisca il rispetto dei vincoli imposti dal processo sottostante e punti alla "fase attuale" del flusso, permettendo di individuare i percorsi attivabili e le condizioni di percorribilità in funzione del *contesto* e degli *attori*.

DOI n. 10.15161/oar.it/76987

Published by
Laboratori Nazionali di Frascati

[1] https://it.wikipedia.org/wiki/Processo_aziendale
[2] https://it.wikipedia.org/wiki/Digrafo_(matematica)
[3] https://it.wikipedia.org/wiki/Automa_a_stati_finiti

1 Introduzione

Le applicazioni web di natura gestionale supportano processi amministrativi complessi che si articolano in sequenze di operazioni su dati e documenti svolte dagli *attori* che vi partecipano. La corretta formalizzazione del flusso di processo è cruciale per la buona riuscita di un'analisi/revisione del processo stesso, sia nel merito del dominio applicativo ("cosa devo fare") che nelle modalità di attuazione del flusso ("chi, quando e come deve farlo"). Allo stesso modo, è cruciale stabilire una politica implementativa efficace, poiché al pari di una cattiva modellazione, un'implementazione mal ponderata, ha ricadute sia sull'efficacia della soluzione che sulla sua manutenibilità nel tempo.

Un approccio basato su *business rule engine*[4] prevede l'implementazione di flussi complessi all'interno degli stessi applicativi gestionali, adottando paradigmi *imperativi* e soluzioni *hardcoded* nel pacchetto software[5]. Tale approccio, per quanto veloce e semplice da attuare, dimostra la sua scarsa efficacia nel lungo periodo, soprattutto a fronte di flussi molto complessi e variabili nel tempo. Al contrario, adottando approcci *dichiarativi* e usando notazioni standard (ad es. BPMN[6]) se ne astrae la complessità operativa, permettendo di disaccoppiare l'implementazione della logica di *business* (tipica di ogni fase) dal *routing* che disciplina i passaggi fra una fase e l'altra. Quest'ultimo in particolare può essere demandato a sistemi terzi che assistono e guidano l'applicazione gestionale nell'iter di processo. In questo lavoro sarà illustrato il modello concettuale e implementativo di un *motore di workflow*, il cui scopo è proprio offrire servizi di *routing* ad applicazioni terze a fronte della formalizzazione di un *workflow* in formato JSON[7].

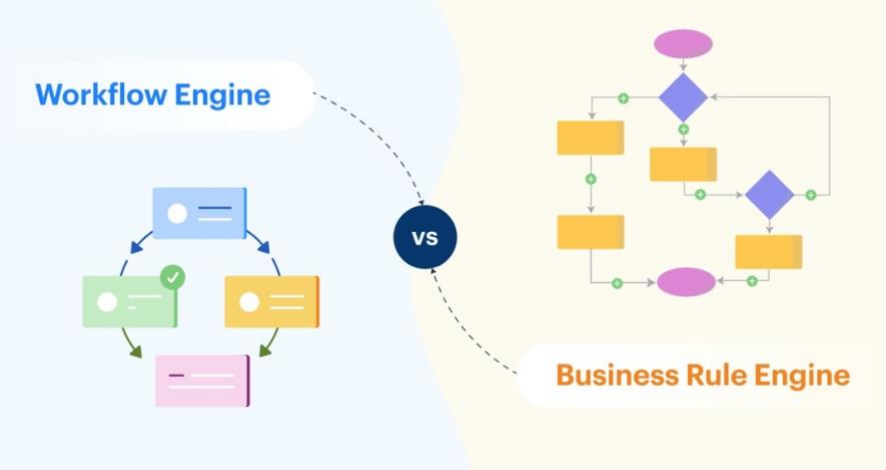

Figura 1 - Confronto fra motore di workflow e motore di business

2 Modello concettuale di un workflow

[4] https://en.wikipedia.org/wiki/Business_rules_engine
[5] https://kissflow.com/workflow/workflow-engine-business-rule-engine-difference/
[6] https://it.wikipedia.org/wiki/Business_Process_Model_and_Notation
[7] https://it.wikipedia.org/wiki/JavaScript_Object_Notation

Come accennato in precedenza, le basi teoriche di un *workflow* si trovano nella disciplina delle *Finite State Machines* (FSM o *automi a stati finiti*), nella specifica connotazione della *Macchina di Mealy*[8]. Un flusso di lavoro, in quanto sequenza complessa di fasi, può essere modellato attraverso un *grafo orientato* composto da *nodi* (fasi di processo) e *archi* (passaggi fra fasi) in funzione del processo e del dominio applicativo rappresentato.

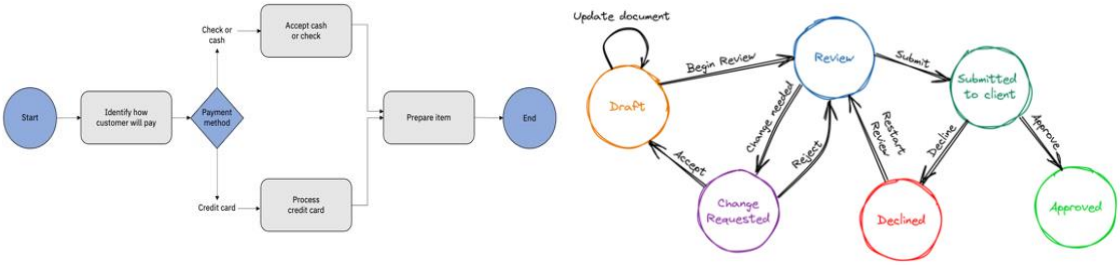

Figura 2 - Formalizzazione di un workflow come automa a stati finiti

2.1 Formalizzazione delle fasi di processo come "nodi"

Ogni *nodo* del grafo (equivalente a una fase del *workflow*) include un insieme di operazioni ben definite che un *attore* deve svolgere. Tali operazioni sono rappresentabili come approvazioni a sistema (es. "click" del direttore che approva una pratica), manipolazioni delle informazioni di *business* (es. aggiunta o modifica di un dato/documento) o attività automatizzate. A loro volta, le fasi possono articolarsi in sotto-processi (con relativi *sub-workflow*) anche paralleli, la cui conclusione è condizione necessaria per permettere il passaggio alla fase successiva.

2.2 Formalizzazione dei passaggi di fase come "archi" (o collegamenti)

I *collegamenti* fra i *nodi* sono responsabili dell'avanzamento del flusso e permettono di governarne la regolare esecuzione in relazione alle attività attese e al loro *output*. Ogni *nodo* può avere uno o più *collegamenti* di uscita, attivabili in funzione del ruolo di chi opera (es. è richiesto il ruolo "direttore"), del *contesto* (es. la variabile X deve essere non nulla) e/o della finalizzazione di sotto-processi, secondo la cosiddetta *funzione di uscita*, che formalizza le condizioni per passare da un *nodo* all'altro. A seconda della numerosità dei *collegamenti* attivabili si verificano diversi scenari:

- **nessun collegamento**: l'*attore* corrente non può operare e il flusso è fermo
- **un collegamento**: l'*attore* può sbloccare il flusso intervenendo a sistema
- **più collegamenti**: l'*attore* deve individuare esplicitamente la prossima fase

[8] https://it.wikipedia.org/wiki/Macchina_di_Mealy

Un *collegamento* può essere navigato in automatico qualora sia l'unico collegamento attivabile e in fase di progettazione sia definito come *fast-forward link*.

2.2.1 Rappresentazione di una condizione di percorribilità

Per rappresentare la *funzione di uscita* di un *collegamento* è possibile fare uso di un insieme di funzioni *booleane* implementate *built-in* e richiamabili attraverso formalismi del tipo *RolesConditions.isNotNull($varName)* ad indicare ad es. che il *metadato* **varName** del *contesto* deve essere presente e valorizzato. Attraverso l'uso di operatori AND e OR è possibile definire proposizioni complesse e aggregate che saranno valutate dal *motore di workflow* per approvare la richiesta di transizione.

2.2.2 Script e funzioni di "post esecuzione"

Nel caso in cui sia possibile far avanzare il flusso, il collegamento scelto verrà "navigato" e si attiverà uno *script*, definito formalmente *funzione di transizione,* che attiverà funzionalità secondarie (es. chiamate di altri servizi o ulteriori manipolazioni del *contesto*) garantendo un comportamento uniforme e deterministico nell'iter di processo. Lo *script* è definito utilizzando linguaggi e formalismi standard ed è eseguito da un motore di scripting basato su *JavaScript*. Questa funzionalità permette di innestare parte della *business logic* imperativa all'interno del *workflow* con il solo scopo di garantire comportamenti uniformi per ogni esecuzione della stessa transizione.

3 Istanza di un *Workflow*

Un *workflow* è la rappresentazione statica di un processo che, una volta avviato (in forma di "pratica") attiva una nuova *istanza*[9]. Un'*istanza di workflow* è un'entità che accompagnerà la pratica (e la *business logic* che la governa) in tutto il suo iter presentandosi come un "puntatore" alla fase attuale di un modello di *workflow*[10].

3.1 Contesto dell'istanza di workflow

Il passaggio da *grafo orientato* ad *automa a stati finiti* è determinato dalla presenza di un *contesto* (o *stato del processo*) all'interno del quale il flusso si articola ed evolve. Il *contesto* racconta la "storia" di un'*istanza di workflow* operando come un "raccoglitore" di dati e documenti a supporto delle diverse fasi. Il *contesto* è formalizzato come una lista strutturata di *metadati* e valori, i primi dipendenti dalla *business logic* modellata nel *workflow* e i secondi derivanti dalla specifica *istanza*. Ogni *metadato* può essere manipolato secondo condizioni legate alla fase attuale e ai ruoli dell'*attore* corrente. Per fare un esempio: se nel *contento* è presente il *metadato* X, sarà possibile definire una funzione che permetta la modifica di X solo al direttore e solo in una determinata fase del *workflow*. Su questa base, l'applicazione gestionale potrà mostrare il relativo campo per

---

[9] https://www.ibm.com/docs/en/opw/8.2.0?topic=fundamentals-workflow-definitions-workflow-instances

[10] https://help.sap.com/docs/workflow-capability/workflow-cloud-foundry/workflow-definition-versus-workflow-instance

acquisire X solo se le condizioni definite su X sono verificate. È possibile definire un *metadato* come "persistente" in modo da estenderne la visibilità oltre il nodo attuale.

4 Rappresentazione di un workflow

A fronte di quanto stabilito nei paragrafi precedenti, si propone la seguente struttura gerarchica per modellare e rappresentare un *workflow*:

- Identificativo univoco del *workflow*
- Lista dei *nodi* (e per ogni nodo…)
    - Tipologia del *nodo* (nodo semplice o nodo *host*)
    - Stato del *workflow* (in corso, terminato o annullato)
    - Lista di *metadati* con relative condizioni di accessibilità
    - Lista di *collegamenti* in uscita (e per ogni collegamento…)
        - *Nodo* di destinazione
        - Condizione di percorribilità
        - Tipo di *collegamento* (semplice o *fast-forward*)
        - *Script* da eseguire a transizione avvenuta

5 Rappresentazione di un'*istanza di workflow*

Analogamente, si propone una struttura per rappresentare l'*istanza di workflow*

- Identificativo univoco dell'*istanza*
- Identificativo del *workflow* da attivare
- *Contesto* dell'*istanza*
- Puntatore al nodo corrente del *workflow*
    - Lista dei collegamenti in uscita
    - Lista di *metadati* visibili

6 Implementazione

Il *motore di workflow* è implementato come un'applicazione web *backend* basata sul framework Java *Spring*. Espone API *RESTful* autenticate con funzionalità di creazione/modifica dei *workflow* e gestione delle *istanze di workflow*. Il motore di persistenza è *MongoDB*, un sistema *NoSQL* che ben si presta a gestire strutture arboree molto articolate. Lo *scaffolding* del codice è guidato dal framework e la nomenclatura delle API riflette la struttura a *risorse* come da standard[11]. I processi di autenticazione e autorizzazione sono mediati dal protocollo *oAuth2*[12] che permette di inoltrare *token* di riconoscimento fra diverse applicazioni con i relativi ruoli dell'utente autenticato. Al momento la *coverage* dei test è al 90% del codice, garantendo solidità e stabilità anche a fronte di modifiche estese, in una logica di *fail-fast*.

[11] https://restfulapi.net/resource-naming/
[12] https://it.wikipedia.org/wiki/OAuth

7 Interazione con applicazioni gestionali

Le applicazioni gestionali possono accedere ai servizi del *motore di workflow* generando automaticamente un *client* basato su file JSON con specifiche *Open API*[13] rilasciato insieme al *backend* del *motore di workflow*. Tale *client* renderà le chiamate HTTP verso il motore in forma di chiamate a funzione per il linguaggio prescelto, astraendo allo sviluppatore la complessità del livello di rete. L'applicazione potrà quindi interrogare il *motore di workflow* tramite il *client* usando i *Data Transfer Object* definiti dal motore stesso.

7.1 Creazione di una nuova *istanza di workflow*

La tipica interazione fra un'applicazione e il *motore di workflow* prevede la richiesta di istanziazione di un *workflow* tramite la chiamata API ***createWFInstance***, indicando l'identificativo univoco del *workflow* da attivare e il *contesto* iniziale inteso come chiavi *metadato/*valore. Il motore genera una nuova istanza del *workflow* richiesto, puntando al *nodo* di partenza del *workflow* in quanto nodo attuale dell'*istanza*.

Poiché la chiamata API è autenticata con il *token* dell'utente connesso, il *motore di workflow* userà le informazioni di autorizzazione presenti nel *token* per stabilire da subito i percorsi ammessi a partire dal nodo corrente, oscurando i *collegamenti* non percorribili. L'*istanza* così creata viene inoltrata all'applicazione gestionale che ne salverà il riferimento all'interno del proprio database, in modo da far avanzare in parallelo le attività di *business logic* sulla pratica con le evoluzioni dettate dal *motore di workflow.*

7.2 Aggiornamento di una *istanza di workflow*

Il riferimento al nodo corrente e la lista di collegamenti percorribili, insieme al contesto e alle regole di accesso ai *metadati* saranno usati dall'applicazione gestionale per comporre una maschera utente con ad es. pulsanti e campi, che riflettano i *collegamenti navigabili* e i *metadati* scrivibili. A partire da questa maschera, infatti, l'applicazione costruirà le basi per la successiva interazione con il *motore di workflow*, popolando campi del *contesto* e acquisendo la volontà dell'utente di proseguire (o meno) nel *workflow*.

Queste due attività si riflettono nell'aggiornamento del *contesto* dell'*istanza di workflow* e nel richiedere la "navigazione" del collegamento relativo. Queste informazioni sono inviate al *motore di workflow* tramite l'API ***updateWFI*** che prende in input l'identificativo dell'*istanza di workflow*, il *contesto* aggiornato dall'applicazione e l'identificativo del *collegamento* da attivare sul nodo corrente.

[13] https://swagger.io/specification/

Il *motore* recepisce la richiesta, ricarica dal proprio database *l'istanza*, la popola con il *contesto* modificato dall'applicazione e prova ad attivare il *collegamento* richiesto dopo averne verificato l'esistenza e la navigabilità. Se il controllo va a buon fine, il *motore* attiva lo *script* definito sul *collegamento* e aggiorna l'*istanza* modificando il puntatore del *nodo* corrente al *nodo* appena raggiunto. Il sistema a questo punto notifica l'applicazione che il passaggio è andato a buon fine, allegando il nuovo *contesto* e le informazioni sul nuovo *nodo* corrente.

7.3 Creazione di *sub-istanze di workflow*

Come anticipato, un nodo di un *workflow* può essere configurato per ospitare a sua volta uno o più *sub-workflow* da avviare in parallelo. Un caso d'esempio è l'approvazione da parte di due responsabili di fondi che possono intervenire parallelamente sul flusso senza che uno debba aspettare l'altro. Un nodo di questo tipo è definito *host* e viene attivato in sede di avvio del *workflow* inserendo nel contesto due *metadati* particolari, riferiti agli identificativi dei *sub-workflow* da attivare e ai *contesti* delle *sub-istanze*.

Al recepimento di questa richiesta da parte del *motore*, seguirà la creazione di due nuove sub-*istanze di workflow* all'interno del nodo *host* che saranno eseguite parallelamente e separatamente. L'iter di queste due istanze sarà oggetto di valutazione per l'uscita dal nodo *host*, definendo politiche di *join* sullo stato delle *sub-istanze*. Ad es. sarà possibile uscire dal *nodo host* se almeno una delle *sub-istanze* è terminata con/senza errori o se tutte le *sub-istanze* sono concluse.

8 Applicazioni attuali e spunti futuri

Dal 2018 è attivo in ambito INFN un *motore di workflow* sviluppato e presidiato dalla **Direzione Sistemi Informativi** dell'Amministrazione Centrale. La versione attualmente in uso è basata su framework *Spring MVC* e supporta l'applicazione "Ciclo Acquisti" con circa 40.000 pratiche/*istanze* gestite. Il caso d'uso è basato su un *workflow* principale di 16 *nodi*, un reticolo di decine di *collegamenti* e 4 *sub-workflow* orientati a specifici sotto-processi (approvazione fondi, gestione RUP, creazione determine e gestione post-gara). È in corso un processo di aggiornamento tecnologico con passaggio al framework *Spring Boot*, in linea con le ultime tecnologie disponibili in area DSI. Questa transizione permetterà di disaccoppiare ulteriormente le attività di gestione *workflow* dall'attività di progettazione degli stessi, utilizzando ad es. interfacce di progettazione basate su BPMN2.