Performance Measurement of Heterogeneous Workflow Engines

Master's Thesis submitted to the

Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics

Faculty of Ingegneria dell'Informazione of the *Politecnico di Milano*
in partial fulfillment of the requirements for the degree of
Laurea Magistrale in Ingegneria Informatica

presented by
Marco Argenti
ID number at Università della Svizzera Italiana : 13980115
ID number at Politecnico di Milano: 798165

under the supervision of
Prof. Cesare Pautasso
co-supervised by
Vincenzo Ferme
at Università della Svizzera Italiana
and
under the supervision of
Prof. Barbara Pernici
at Politecnico di Milano

September 2015
Accademic Year 2014/2015

To my beloved

If today were the last day of my life,
would I want to do what I am about to do today?
The only way to do great work is to love what you do.
Don't settle.
You have to trust that the dots will somehow connect in your future.

Steve Jobs, conference at Stanford University.

Abstract

Nowadays companies formalize their business processes to maximize their efficiency and effectiveness. Workflows are business processes with components that automate companies' processes and the workflow executions are managed by the Workflow Management System (WfMS). In the last years the number of WfMSs, and in particular the WfMSs supporting the latest version of Business Process Model and Notation (BPMN 2.0), has increased. Even if a benchmark of WfMS BPMN 2.0 is recognized as necessary and similar tools for benchmarking WfMSs exist, a benchmark of WfMS BPMN 2.0 has still not been developed. In this context, the BenchFlow project has the goal to be the first BPMN 2.0 WfMS benchmark. The BenchFlow framework is formed by a set of services and its components are: the driver, the collector, the monitor, the data cleaner and reconciler and the performance meter and data aggregator.

This thesis is a contribution to the BenchFlow project and it focuses on the design, implementation and evaluation of the data cleaner and reconciler and of the performance meter and data aggregator. Additionally, in this thesis, WfMS metrics are defined employing the Software Measure Definition Method; Processes Completion Time, Number of Completed Processes and Number of Uncompleted Processes metrics are implemented in the performance meter and data aggregator. Finally, to prove that the application of the two tools developed is feasible, as proof of concept, load tests are performed on two WfMSs and the implemented WfMS metrics are applied to compare their performance.

Thus, the main contributions of this thesis to the research community are the creation of two tools, one that standardizes data from all the source databases of different WfMSs, and one that aggregates stored data to obtain information about WfMS BPMN 2.0 metrics, belonging to the BenchFlow framework, and their application on one trial experiment run as a proof of concept.

Sintesi

Da qualche decennio, esistono società di notevoli dimensioni e con l'aumentare della dimensione la complessità dell'azienda è incrementata esponenzialmente. Questa complessità può rendere difficoltoso, o almeno non sempre efficiente, il raggiungimento degli obiettivi che l'azienda si è posta; perciò le imprese si sono rese conto della necessità di formalizzare in modo facile, comprensibile ed immediato i meccanismi di funzionamento della società stessa. Questi meccanismi di funzionamento sono chiamati processi aziendali.

I processi aziendali sono formati da attività collegate e correlate tra loro. Tali processi aziendali sono progettati per acquisire input e trasformarli in specifici output [96]. In particolare, il Business Progress Management (BPM) è una disciplina che si occupa della gestione dei processi di business in aziende. Lo scopo del BPM è fornire all'azienda uno strumento che definisca, esercisca e controlli i processi di business in modo tale che l'azienda riesca a raggiungere le proprie strategie di business [102].

Infatti, i sistemi BPM offrono risultati in termini di riduzione dei costi e aumento del fatturato quando supportano attività molto complesse, critiche per l'azienda e che vedono interagire un numero elevato di persone appartenenti a funzioni differenti. Ad ogni modo, affinché il BPM supporti realmente gli obiettivi aziendali, è necessario che tutti gli utenti, anche con diverse conoscenze, siano in grado di comprendere i modelli e le notazioni che descrivono tali processi [44].

In passato le notazioni utilizzate per la modellazione dei processi potevano essere molto diverse; invece ora viene utilizzata una notazione standard chiamata Business Process Model and Notation (BPMN) e l'ultima versione è il BPMN 2.0 [55].

Per garantire lo scopo di fornire un meccanismo semplice per la modellizzazione dei processi di business, gestire e soddisfare le relative complessità, l'aspetto grafico della notazione è organizzato in specifiche categorie. Nella notazione BPMN sono presenti cinque diverse categorie di elementi base: i flow objects, i connecting objects, gli swimlanes, i data e gli artifacts.

Inoltre, con il passare degli anni vi è stata la possibilità di velocizzare l'esecuzione dei processi aziendali con l'introduzione di nuove tecnologie. Una di queste tecnologie riguarda l'uso di Workflow Management Systems (WfMSs), i quali hanno permesso, parzialmente o totalmente, l'esecuzione automatica dei processi aziendali [49]. I Workflow Management Systems hanno principalmente tre aree funzionali:

- Build-time functions, riguardano la funzionalità di modellazione dei processi e delle relative attività, quindi questa funzionalità è utilizzata soprattutto dai designer di processi.

- Run-time functions, riguardano la gestione dell'esecuzione del modello del processo facendo rispettare il flusso definito e gestendo le attività ed i requisiti che tali attività richiedono, come ad esempio dati di input. Inoltre, devono gestire anche l'aspetto di ripristino dell'esecuzione e perciò mantengono anche dati riguardanti la storia dell'esecuzione.
- Run-time interactions, riguardano la gestione delle interazioni del WfMS con l'esterno. Le interazioni con l'esterno si dividono principalmente in interazioni con l'utente e interazioni con i Web Services. Questi due tipi di interazioni hanno bisogno dell'uso di interfacce: User interface e Application interface. Inoltre, le interazioni con gli utenti e con i Web Services hanno bisogno anche di gestori, ed in particolare nel caso degli utenti il gestore si chiama Worklist. Infine le interazioni vengono definite nel processo dai designer di processo tramite le funzionalità di Build-time.

Inoltre, l'utilizzo dei WfMS comporta una serie di vantaggi per l'azienda che li utilizza. Innanzitutto, l'automazione dei processi aziendali comporta una riduzione dei costi e dei tempi per eseguirli e un efficientamento dei processi stessi, poiché i passaggi inefficienti vengono eliminati. Oltre a ciò, grazie alla minore incertezza nell'esecuzione dei processi, i WfMSs incrementano la produttività e il customer service. In aggiunta, essi amplificano la flessibilità aziendale poiché la loro riprogettazione non comporta un'onerosa riconfigurazione. Infine, i WfMSs incrementano il controllo sui processi grazie alla standardizzazione degli strumenti di controllo e supportano le decisioni e le pianificazioni aziendali dato che aumentano l'accesso alle informazioni da parte del management aziendale.

Questi vantaggi hanno portato ad una rapida diffusione dei Workflow Management Systems [97], soprattutto di quelli che supportano il BPMN 2.0 [87]. Dato che l'utilizzo di un Workflow Management System rispetto ad un altro può impattare fortemente l'esecuzione dei processi di un'azienda, è importante avere una metodologia per comparare i diversi Workflow Management Systems e poter eseguire delle analisi che permettano di identificare il migliore WfMS per la tipologia di processi che le imprese eseguono. Nonostante ciò, ci sono poche informazioni riguardanti la performance dei WfMSs, [29, 41, 43] e i pochi studi empirici si focalizzano, ad esempio:

- sull'apprezzamento della tecnologia da parte degli utenti finali [62];
- sull'implementazione dei WfMSs[76];
- sulla riduzione del WfMS lead time [74];
- sull'efficienza delle individuali attività strutturali e sulla comparazione dei diversi engines [15];
- sull'impatto del database utilizzato nei server e nei sistemi throughput [41];
- sull'impatto del database utilizzato nel carico di lavoro (workflow) [17].

Lo studio che può essere considerato una valutazione della performance dei WfMSs è "engine performance evaluation by a black-box approach" scritto da Daniel, F., Pozzi, G., Zhang, Y. [29], dove, attraverso l'approccio chiamato black box, vengono comparati cinque Workflow Management Systems. Questo approccio, però, si focalizza sullâĂŹutilizzo di diverse macchine virtuali, ognuna configurata in base alle impostazioni di default del rispettivo WfMS.

Anche se sono presenti pochi studi empirici, è universalmente riconosciuto il bisogno di un

benchmark per la performance dei WfMSs [58, 64, 100]. Questa necessità di un performance benchmark è presente anche per i BPMN 2.0 WfMSs [100], soprattutto poiché nel 2014 quasi 20 Workflow Management Systems supportano il BPMN 2.0. [87].

In questo contesto, l'Università di Lugano e l'Università di Stuttgard hanno iniziato un progetto chiamato BenchFlow, il quale ha come scopo lo sviluppo del primo benchmark framework per i WfMSs BPMN 2.0 [13]. Il BenchFlow framework è formato da una serie di servizi che non devono interferire con la normale esecuzione dei componenti del sistema. I componenti del BenchFlow framework sono:

- Driver, è responsabile dell'infrastruttura. Esso gestisce il set-up delle configurazioni, il workload e la funzionalità dell'intero sistema durante l'esecuzione e la pulizia quando l'esecuzione è terminata.
- Monitor, controlla l'esecuzione del workload. Esso ha il compito di capire quando il sistema ha finito di eseguire il carico, evitando di interferire con l'esecuzione.
- Collector, raccoglie i dati generate dai componenti del sistema e, in particolare recupera i log files o database records. Esso agisce dopo che il monitor ha assicurato che il WfMS ha completato tutte le richieste inviate del driver.
- Data cleaner and reconciler, il quale pulisce e riconcilia i dati collettati dai database dei vari WfMSs. Quindi, il suo obiettivo è la standardizzazione dei dati provenienti da database di diversi Workflow Management Systems.
- Performance meter and data aggregator, il quale è l'aggregatore di dati e il misuratore della performance. Esso struttura e organizza i dati in modo chiaro e comprensibile, affinché possano essere utilizzati per le decisioni dagli utilizzatori.

Questa tesi è un contributo al progetto BenchFlow. Essa si focalizza sulla progettazione, implementazione e valutazione del data cleaner and reconciler e del performance meter and data aggregator. Inoltre, descrive i vari componenti del BenchFlow, identifica le metriche dei WfMSs e provvede a presentare una proof of concept con lo scopo di dimostrarne la fattibilità delle analisi di confronto dei WfMSs utilizzando i due componenti implementati.

Riguardo la definizione delle principali metriche dei WfMSs, esse vengono identificate applicando il Software Measure Definition Method (SMDM) [19] e prendendo spunto dal paper "A Framework for Benchmarking BPMN 2.0 Workflow Management Systems" [37] di Vincenzo Ferme, Ana Ivanchikj e Cesare Pautasso.

La metodologia SMDM è basata su quattro fasi:

- definizione della metrica, la quale consiste nella identificazione e creazione della metrica;
- validazione teorica attraverso il SMART framework [32], il quale valuta la specificità, la misurabilità, l' attendibilità, la rilevanza e la tempestività della metrica;
- validazione teorica, la quale consiste nell'uso di dati sperimentali per validare la metrica nella pratica;
- validazione psicologica, la quale descrive come la metrica influenza i soggetti coinvolti e usualmente è aggregata alla validazione teorica.

Le metriche identificate, riguardano le tre principali entità del WfMS [8, 38] che sono il Workflow Engine (WfE), il process e il construct. Il Workflow Engine esegue i processi che sono richiesti dall'utente. Il process è il processo che viene eseguito dal Workflow Engine quando l'utente lo richiede. Il construct è un componente del process. Il Business Process Model and Notation identifica come costrutti: i flow objects, i connecting objects, le swimlanes, i data e gli artifacts. Quindi, il termine costrutto può essere riferito a differenti elementi BPMN. Come è deducibile dalla loro definizione, queste tre entità sono in relazione tra di loro; infatti esse compongono una gerarchia, dove il Workflow Engine esegue i processi i quali sono composti da costrutti.

Le metriche identificate per il Workflow Engine sono:

- Response time;
- Throughput;
- Latency;
- DB usage;
- Network usage;
- Disk usage;
- CPU usage;
- Capability;
- NumCompletedProcesses.

Invece, quella per il process è il complition time.

Infine, quelle per il construct sono:

- Completion time;
- Delay;
- Latency;
- Construct capability;
- NumCompletedConstruct.

Dunque, secondo le attività di identificazione e creazione della metrica della metodologia SMDM per ogni metrica è stato definito il nome, il goal, la descrizione e la pseudo formula. Inoltre, alle metriche definite è stato applicato il SMART framework [32]. Tutte le metriche identificate sono state ritenute SMART poiché soddisfano tutte le caratteristiche SMART.

Infine, le metriche SMART implementate nel performance meter and data aggregator sono: processes completion time, number of completed processes and number of uncompleted processes; dove quest'ultima è la metrica complementare a number of completed processes. Le metriche individuate con la metodologia SMDM sono fondamentali per la loro implementazione nel performance meter and data aggregator e poiché esse evidenziano i requisiti minimi che devono essere considerati per realizzare lo schema del database del data cleaner and reconciler.

Il data cleaner and reconciler ha l'obiettivo di standardizzare i dati provenienti da database di differenti Workflow Management Systems. Questa standardizzazione è richiesta poiché i WfMSs immagazzinano i dati rilevanti per l'analisi con differenti strutture e notazioni; inoltre permette di avere una struttura che semplifica l'analisi dei dati rendendo palese le relazioni tra i dati stessi. Quindi il data cleaner and reconciler esegue il processo di estrazione, trasformazione e immagazzinamento (ETL), cioè esso è il componente del BenchFlow che recupera i dati dai database dei WfMSs e li trasforma in un altro formato in modo automatico e li inserisce in un database di destinazione chiamato CleanRawData. Infine, il data cleaner and reconciler gestisce anche le statistiche generate dall'API di Docker riguardanti l'utilizzo delle risorse dell'environment, e.g. il database che permette l'esecuzione del WfMS.

I dati che vengono immagazzinati nel database del WfMS ed il contesto dei WfMSs permettono di identificare due entità: l'entità dei costrutti e l'entità dei processi. La prima entità riguarda i costrutti utilizzati per costruire un modello di processo, mentre l'entità dei processi riguarda le istanze di processo; tali istanze di processo sono state eseguite secondo un trial, che è un'esecuzione di un esperimento composto da richieste di esecuzioni di modelli di processo. Quindi è utilizzata anche un'entità esperimento che riguarda i trial eseguiti degli esperimenti. In aggiunta, un'entità environment è necessaria per gestire i dati riguardanti le statistiche di utilizzo dell'environment durante l'esecuzione dei trial. Dalle entità evidenziate, si possono riconoscere le relazioni tra le entità environment ed esperimento e la relazione tra le entità esperimento, processo e costrutto.

Particolare attenzione va posta nelle relazioni tra le entità esperimento e processo e tra le entità processo e costrutto; infatti, queste due relazioni applicano le tre regole di normalizzazione. Inoltre in fase di analisi è rilevante sapere quali costrutti appartengo ad una particolare esecuzione di un esperimento, cioè a quale trial. Ad ogni modo, questa analisi richiede di usare entrambe le relazioni rendendo tale analisi onerosa. Per questo motivo è stata effettuata una denormalizzazione dell'entità costrutto che avrà anche una relazione diretta con l'entità esperimento. Un altro dettaglio sul design ed implementazione del data cleaner and recinciler database riguarda le chiavi primarie delle diverse entità che vengono determinate utilizzando MD5 hash function, la quale permette di utilizzare un singolo campo come chiave primaria dell'entità. Inoltre, è veloce da computare, è sufficientemente resistente alle collisioni e grazie alla lunghezza fissa dell'output permette di ridurre lo spazio di memoria utilizzato.

Il data cleaner and reconciler è stato realizzato con Java e la struttura principale si basa su una gerarchia che ha Cleaner come superclasse. Dunque, Cleaner è una superclasse astratta e provvede alle funzionalità di connessione con il database CleanRawData e al relativo caricamento dei dati. La superclasse Cleaner ha due sottoclassi: EnvCleaner e DBCleaner. La sottoclasse EnvCleaner è una classe concreta e gestisce i dati generati dell'API di Docker riguardanti l'environment, mentre DBCleaner gestisce i dati appartenenti ai database dei WfMSs e quando viene aggiunto un nuovo WfMS questa classe deve venir estesa specificando quali dati del WfMS devono essere trasformati. Inoltre, per gestire le interazioni con i database, sia quelli dei WfMSs che il database CleanRawData, viene sfruttato il framework Java Object Oriented Query Language (jOOQ) che è un Object-Oriented Query Language specifico per Java che semplifica l'interazione con i database e la gestione dei dati derivati. In aggiunta, vengono utilizzate delle classi di supporto per l'interazione con i databases: StoreCleanData e DB. In particolare la classe StoreCleanData gestisce l'inserimento dei dati nel CleanRawData attraverso immissioni di dati in batch per ridurre gli overhead di comunicazione, mentre la classe DB gestisce le

connessioni ai database attraverso i meccanismi di connection pool.

Infine, il data cleaner and reconciler è valutato; in particolare sono analizzate le performance, le quali mostrano che l'operazione più onerosa è l'interazione per l'inserimento dei dati, in quanto è necessario costruire la query per inserire i dati, trasmettere la richiesta di inserimento al database e inserire effettivamente i dati nel CleanRawData. Inoltre la memoria RAM è identificata come un bottleneck del componente. Per quanto riguarda la correttezza dei dati, essa viene garantita sfruttando i vincoli di integrità referenziale, con un controllo basato sul numero di record e applicando dei JUnit tests. Inoltre, viene valutata e descritta anche la possibilità di identificare i dati originali da cui sono stati generati i dati salvati sul database CleanRawData e la possibilità di aggiungere nuovi WfMSs da cui poter ottenere dati.

Invece, il performance meter and data aggregator ha l'obiettivo di aggregare i dati, presenti nel CleanRawData, per ottenere informazioni. Questa aggregazione è eseguita basandosi sull' applicazione delle metriche, sugli esperimenti e sui trial legati ad uno specifico esperimento.

Inoltre, i dati generati dal performance meter and data aggregator devono essere mantenuti per permettere di fare analisi future; perciò è necessario utilizzare un database che contenga tali dati. Dato che i dati mantengono i valori delle metriche e nuove metriche possono essere aggiunte, il dato deve poter essere modificabile aggiungendo ulteriori valori. Per soddisfare tale requisito, MongoDB è stato scelto come database in quanto è un database non-relazionale e permette di mantenere dati senza specificare a priori lo schema del dato stesso, come invece avviene nei database relazionali. In aggiunta, MongoDB è stato scelto perché mantiene una meta-struttura del dato, permette di accedere ai dati rapidamente ed ha un'alta curva d'apprendimento data dalla sua similarità con i database relazionali.

Per quanto riguarda l'implementazione del componente, Java è utilizzato per la sua realizzazione. Dato che nuove metriche devono poter essere aggiunte incrementalmente, una gerarchia riguardante le metriche è creata. La superclasse Metric è astratta ed oltre a gestire l'interazione con il database del performance meter and data aggregator contiene un metodo che deve essere implementato dalle sue sottoclassi concrete. Tale metodo riguarda la computazione della metrica sia a livello di trial, cioè di una singola esecuzione di un esperimento, sia a livello di esperimento, cioè aggregando dati di più esecuzioni dello stesso esperimento. La superclasse Metric ha due sottoclassi astratte: ProcessMetric e ConstructMetric che verranno estese per implementare concretamente le metriche che riguardano rispettivamente i processi ed i costrutti. Tali sottoclassi concrete vengono utilizzate nella classe MetricCalculator, la quale istanzia l'oggetto concreto di tale metrica e calcola la metrica sfruttando il metodo esposto dalla superclasse Metric. Inoltre, la classe MetricCalculator gestisce le interazioni con il database CleanRawData per ottenere i dati da utilizzare per calcolare le metriche e, se richiesto dall'analisi da effetturare, ha una opzione per identificare se il trial da analizzare ha dei processi che non sono stati completati prima di calcolare le metriche. Le metriche implementate nel performance meter and data aggregator sono: ProcessCompletionTime, NumberCompletedProcess e NumberUncompletedProcess. La classe ProcessCompletionTime calcola il tempo di completamento dei modelli di processo; la classe NumberCompletedProcess computa il numero di instanze di processo che hanno raggiunto il completamento; e la classe NumberUncompletedProcess quantifica il numero di instanze di processo che non hanno raggiunto il completamento.

Infine il performance meter and data aggregator è valutato; in particolare sono valutate le

performance, le quali mostrano che l'operazione più onerosa è l'ottenimento dei dati dal CleanRawData. Ad ogni modo bisogna considerare che questa operazione viene eseguita un'unica volta ed i dati ottenuti vengono forniti a tutte le metriche da calcolare. Inoltre, ogni volta che viene aggiunta una nuova metrica e che viene valutato un nuovo trial, la quantità di dati da immagazzinare all'interno del database cresce notevolmente; quindi viene valutato che applicando il meccanismo di sharding, si ha la possibilità di estendere la capacità di storage del database del performance meter and data aggregator. Infatti, lo sharding risponde alla necessità di un incremento della domanda di mantenimento dei dati dividendoli in base alla chiave di shard su diverse macchine.

Inoltre, è realizzata una proof of concept con lo scopo di dimostrarne la fattibilità delle analisi di confronto dei WfMSs utilizzando solamente i due componenti implementati. L'infrastruttura utilizzata è basata su un computer singolo, un MacBook Pro machine con la versione 10.9.5 OSX e 64 bit di architettura; quindi essa non rispetta gli standard del BenchFlow che prevedono una controllata infrastruttura Cloud.

L'analisi è compiuta per due WfMSs (Activiti e Camunda); attraverso l'esecuzione di load tests rappresentativi per imprese di: piccole, medie, medie-grandi e grandi dimensioni. In particolare i load tests sono basati sul numero di utenti che interagiscono con il WfMS e sono considerati: 50 utenti per le piccole aziende, 250 utenti per le medie aziende, 400 utenti per le medie-grandi aziende e 1000 utenti per le grandi aziende. Gli utenti che interagiscono con il WfMS aumentano secondo un periodo di rampa che è equivalente al numero di utenti che il test considera. Quindi, il test usa una stepwise continuous injection di utenti e ogni secondo un nuovo utente è iniettato nel sistema fino a che il numero di utenti del test non è raggiunto. Lo stesso avviene nella fase di declino, dove, però, ad ogni secondo un utente è rimosso dal test.

I modelli di processo utilizzati nei tests, derivano da una reale collezione di processi chiamata "Insurance Process and Service Models" fornita da IBM. La collezione presenta 400 modelli di processo che possono essere classificati, secondo la tesi di Ivanchikj [53], in sei cluster, basandosi sulle loro caratteristiche statiche [28, 53]. I workload dei tests sono effettuati su quattro modelli di processo basati su due cluster: cluster 3 e cluster 4, dove ogni cluster ha due modelli di processo.

In generale ogni test mantiene il workload per novanta minuti, solo il test con 50 utenti del cluster 4 è eseguito per trecento minuti. Ogni test per 50, 250, 400 e 1000 utenti è eseguito conseguentemente e anche per i due clusters i tests vengono eseguiti uno alla volta. Inoltre, ogni test è eseguito una volta sola, quindi l'analisi è basata su un singolo trial.

Le metriche utilizzate per determinare la performance dei WfMSs sono:

- completion time of processes, è il tempo richiesto dal WfMS per eseguire completamente le istanze di processo e nell'analisi viene considerata la media di questa metrica;
- number of completed processes, è il numero d'istanze di processo eseguite che sono state completate;
- request error percentage, è computato dal driver e considerato per capire quante richieste di esecuzione di processo sono state rigettate dal WfMS;
- responce time, la quale media è computata dal driver ed è considerata per interpretare i

risultati ottenuti.

Come la tabella 1 riassume, il miglior WfMSs, tra i due analizzati, non è sempre lo stesso, ma la performance varia a seconda della grandezza dell'azienda e delle caratteristiche dei clusters.

Table 1. Sintesi dei risultati dei tests

Test con		Media del tempo di completamento (ms)		Numero di processi completati (#processi)		Best performed WfMS
		Camunda	Activiti	Camunda	Activiti	
Cluster 3	50 users	∼	∼		✓	Activiti
	250 users	✓		∼	∼	Camunda
	400 users	∼	∼		✓	Activiti
	1000 users	∼	∼		✓	Activiti
Cluster 4	50 users	✓		∼	∼	Camunda
	250 users	✓		∼	∼	Camunda
	400 users	∼	∼		✓	Activiti
	1000 users	∼	∼		✓	Activiti

∼: value difference is not relevant; ✓: value difference is relevant.

Infatti, basandosi sui risultati e sull'analisi effettuata, per il cluster 3, Activiti risulta essere il WfMS migliore tra i due per piccole, medio-grandi e grandi aziende. Invece, Camunda dovrebbe essere scelto nel caso di medie imprese. Per il cluster 4, Camunda può essere considerato come migliore WfMS tra i due, per piccole e medie aziende, viceversa dovrebbe essere scelto Activiti se la dimensione dell'impresa è maggiore.

Per quanto riguarda la percentuale di errore essa è rilevante solo per i tests con 400 e con 1000 utenti. Gli errori principali di Activiti sono socket exceptions ed errori di internal servers, invece per Camunda gli errori maggiormente rilevanti sono exceptions, connection exception ed errori di internal server.

La proof of concept effettuata ha varie limitazioni e le principali riguardano: l'uso di un'unica macchina con ridotta potenza per gestire l'infrastruttura, la trasformazione dei modelli di processo per renderli eseguibili e l'utilizzo dei soli componenti implementati.

Invece, la limitazione principale dei due componenti riguarda la metodologia di interazione che richiede l'uso della command line. Inoltre, le limitazioni del data cleaner and recociler riguardano: la memoria RAM disponibile, l'utilizzo del framework jOOQ che supporta solo database relazionali e la necessità che i dati dell'environment siano derivati tramite le API di Docker. Mentre la principale limitazione del performace meter and data aggregator riguarda la velocità di lettura dei dati dal CleanRawData.

Per finire, il maggiore contributo di questa tesi alla comunità di ricerca è la creazione di due componenti appartenenti al BenchFlow framework [13]: il data cleaner and reconciler che standardizza i dati provenienti da database di differenti WfMSs e il performance meter and data aggregator che aggrega i dati immagazzinati e ottiene informazioni riguardo alle metriche dei WfMSs; oltre alla loro applicazione compiuta per due WfMSs, attraverso l'esecuzione di load tests e l'analisi dei rispettivi risultati.

I principali lavori futuri sono l'estensione del data cleaner and reconciler ad altri WfMSs, l'aggiunta di nuove metriche per il performance meter and data aggregator e l'utilizzo del BenchFlow framework per effettuare analisi riguardanti l'individuazione di parametri rilevanti per valutare le performance dei WfMSs in modo tale che le aziende possano avere tutte le informazioni necessarie per confrontare i WfMSs e scegliere consapevolmente quale adottare.

Acknowledgements

First of all, I would like to thank my family that has been with me during all the steps of this journey.

Moreover my thanks go to my advisors, Prof. Dr. Cesare Pautasso and Prof. Dr. Barbara Pernici, who gave me the opportunity to do my thesis as a contribution to the BenchFlow project and so allowing me to better understand BPMN 2.0, Workflow Management Systems and their benchmark state of art. I really appreciate their knowledge sharing and their guidance helped me in all the time of research and writing of this thesis.

Additionally, I would like to express my gratitude to Vincenzo Ferme, a PhD student working on the BenchFlow project. He supported me during all my work and he was always available.

Finally, I would like to thank my relatives and friends for supporting me.

Contents

Figures

Tables

Terminology

To better benefit the content of the thesis the follow terminology description can be useful. The key terminologies are:

- BenchFlow Framework, it is a set of services and its components are: driver, monitor, collector, data cleaner and reconciler, performance meter and data aggregator [78].
- Business process, it is a process that focused upon the production of particular products. These may be either physical products, such as an aircraft or bridge or less tangible ones such as a design, a consultation paper, or an assessment. So the product could also be a service [96].
- Business Process Management (BPM), is a management discipline that deals and manages business processes into a company. The purpose of BPM is to provide to companies a tool that defines, executes and controls business process so that companies might reach their business strategies [102].
- Business Process Model and Notation (BPMN), is the most commonly used processes modelling standard [55]. It is a language that standardises the gap between the business process design and the process implementation decreasing the existing fragmentation between the enormous amount of notations and the process modelling tools.
- Construct, it is a general element of the BPMN notation [75].
- Construct entity, it contains information regarding the constructs used to build the process.
- Data cleaner and reconciler, it cleans and reconciles the data collected by the BenchFlow framework [78] and it is a BenchFlow component.
- Environments, they are all systems that allow the execution of trials.
- Experiment entity, it contains information about experiment trials, which are executions of the experiment. Experiment, it consists of procedures to test WfMS. Trial, it is an execution of an experiment.
- Extraction, Transformation and Loading (ETL) tool it is a tool that retrieve data from source database, transforms the date in another format in an automatic way and refreshes or updates or insert the data in the destination database [99].
- Hash functions, they are functions that operate and map an input string of different length to an output of bit string with a fixed length, called hash [69].

- Load test, it is a type of test that sends to the system a volume of requests that is expected to match real request volume [67].
- MD5, it is a type of hash functions and it is a strengthened version of MD4 [84].
- Measure, it is defined as a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute [36].
- Metric, it is a measurement function [7].
- MongoDB, it is a NoSQL database and it is a type of Document Store [27].
- Non-relational databases, they are databases which have schema-less data structures, very simple replication, high availability. They scale horizontally and they use alternative querying methods. [77, 98]
- NoSQL, it is a class of database. In a NoSQL database, SQL is not the only data query language used and the data store does not follow the relational model [94].
- Object-Oriented Query Language (OQL), it is designed to provide an object-oriented query interface for traditional relational database systems [65].
- Object relational mapper (ORM), it is a tool which enables fetching objects from relational databases [40].
- Performance meter and data aggregator, it aggregates stored data to obtain information [78] and it is a BenchFlow component.
- Process entity, it contains information about the process instance of a specific experiment trial.
- Process instance, it is a process model under execution [34].
- Process model, it is a visualization of a process which specifies how tasks have to be performed and in which order to successfully complete the process [96]. It can also be referred to as process definition.
- Relation databases, they are databases where relationships between data items are based on item values [24].
- Workflow, it is a partial or total automation of a business process [49].
- Workflow Engine (WfE), it is a service that enable and enact the run time functions and its main functionalities are to execute process models, manage process instances and interact with external sources [49].
- Workflow Management System (WfMS), it is a system for the processes execution management, it especially runs workflows.
- Workload, it is defined as all the inputs that are sent to the system under test [39].
- Throughput, it is the amount of work that the Workflow Engine is able to handle in a unit of time [63].

Chapter 1

Introduction

With the increasing of a company size, the firm's complexity boosts exponentially. This complexity could hinder the company's goals achievement or could make the firm lose efficiency. Thus the companies understood the necessity of formalizing in an easy, comprehensible and immediate way the firm's operation mechanisms that allow the company to operate and serve particular goals.

The operating mechanisms interest almost everything; for instance, software product development, customer acquisition, and so on. Those operating mechanisms, which do not affect only the big companies but all firms, are called business processes. Indeed, each company has more than one process to carry on and some processes are common to all firms, such as the management of taxes.

Other business processes are specific of the company sector where the company operates and only a very limited number of processes are specific to a firm. However also common processes among firms might be slightly different. Anyway, inside a company, the business processes' individuation and formalization allow the optimization, the simplification or the faster and easier application of those processes. The individuation of a firm's business processes could be hard if the processes are embedded in the firm's operation, whereas during the business process formalization there is the necessity of considering the point of view of the various process actors.

Finally, to be able to improve an existing process, a considerable intellectual activity is always needed. Over the years the business processes' execution has become quicker due to the introduction of new technologies. One of those new technologies is the Workflow Management System (WfMS), which grants partially or totally the automatic execution of the processes and it allows the simplification of other relevant execution aspects such as the communication, the data retrieval, the handover of the execution process responsibilities, the consistent execution state maintenance, and so on. Ultimately, the use of the Workflow Management System in the processes execution of the company's business processes can significantly affect the firm performance and the achievement of the company's goals. Workflow Management Systems start their diffusion since the nineties, whereas their concept is older [35], and they are "one of the most successful genres of systems supporting cooperative working" [33].

There are few information about the performance of WfMSs [43] and the few empirical studies about WfMSs do not focus on their performance, whereas they focus on appreciation of the technology by final users, [62], or their implementation [76].

Due to the lack of a performance benchmark of WfMSs it is important to design and develop a benchmark for assessing and comparing the performance of Workflow Management Systems. The first benchmark with this goal is the BenchFlow [13], that plans to use a model-driven, self-/recursive testing approach to eliminate the impact of the external services by having them implemented as processes and it focuses mainly on BPMN 2.0 WfMS.

This thesis is born in the context of the BenchFlow project. As a matter of fact, in this study to allow the analysis and management of the data from different Workflow Management Systems a data cleaning and reconciling tool is designed and implemented. Moreover, in this thesis, a performance meter and data aggregator is designed and implemented to examine the data, so that it is possible to obtain information about the used Workflow Management System, such as average completion time of executed processes and number of executed processes in a given amount of time. Thus, the thesis scope is Workflow Management System performance.

Finally, to demonstrate that those types of analyses are feasible and the two tools developed work properly, a proof of concept is provided. Thus, given the data cleaner and reconciler and the performance meter and data aggregator these kind of analyses are enabled in the BenchFlow framework and a first proof of concept is provided. On a single low-power machine load tests are performed and the different data from two WfMSs databases are cleaned and reconciled through the data cleaning and reconciling and the clean data are structured and organized to obtain the WfMSs metrics and to compare the WfMSs performance through the performance meter and data aggregator.

1.1 Context

The business processes are a composition of activities connected and correlated to each other. They are modelled to acquire inputs and to transform them into specific outputs [96]. To define business processes using notations that allow to execute and to control process instances, a software system called Business Process Management (BPM) is used.

In the past, notations used to model processes could be very different, whereas nowadays a common standard notation is used: the Business Process Model and Notation [102], whose last version is the BPMN 2.0 [55].

One of the main goals of BPMN is to allow the modelling of executable processes. Indeed the BPMN execution semantic specifies how every element has to be translated to perform in a BPM system [102].

Moreover, the companies have to manage an enormous amount of data so their business processes need to be automatic. Workflows are business processes with components that automatize completely or partially the process and the workflow executions are managed by the Workflow Management System [49]. In particular the WfMSs have mainly three functional areas:

- Build time functions, which concern modelling functionality of the processes and their activities;
- Run time functions, which concern the process execution management;
- Run Time interactions, which concern the management of the WfMS external interaction.

In the last years the number of Workflow Management System, and in particular the WfMSs supporting BPMN 2.0 [87], has been increasing [97]. Besides the evident necessity for benchmark to compare the different WfMS BPMN 2.0 [100], not many attempts have been made to define one up until now. In this context the University of Lugano and the University of Stuttgart initiated a project named BenchFlow with the ultimate goal of developing such a benchmark framework [13].

The BenchFlow framework enables the automation of the performance tests execution on WfMSs. It does so by providing a set of services and libraries that simplify and automate the definition and the execution of performance tests, as well as the data gathering and analysis. The BenchFlow framework components are:

- Driver, which is responsible of the infrastructure. In particular it manages set-up configurations, the workload, the functioning of the whole system during execution, the clean up when the execution is finished and so on.
- Monitor, which watches the workload execution. It tries to understand when the system has finished executing the load without interfering with the execution itself.
- Collector, which collects the data generated by the system components, in particular by retrieving log files or database records. It is executed after that the monitor has ensured that the WfMS has completed all requests sent by the driver.
- Data cleaner and reconciler, which cleans and reconciles data collected from WfMS database. Thus its objective is to allow standardizing data from databases of different Workflow Management Systems.
- Analyzer, which is the data aggregator and performance meter. It has to structure and organize the data in a way that they can be understandable and clear for the users' decisions.

1.2 Goals

The BenchFlow framework enables the automation of the performance tests execution on WfMSs, providing the execution of performance tests, as well as the data gathering and analyses. WfMSs collect the data in different structured forms in their databases, so a tool is needed to standardize those data; moreover the clean data should be structured and organized to have understandable and clear information about the WfMSs performance.

Thus, the goals of this thesis are:

- to build a data cleaning and reconciling tool that allows the analysis and management of the data from different WfMSs;

- to build a performance meter and data aggregator tool to examine the data, so that it is possible to obtain information from the Workflow Management System used;
- to demonstrate the feasibility of the application of the two tools developed testing them and applying the implemented WfMS metrics on two real world open source Workflow Management Systems.

1.3 Challenges

During the implementation of the data cleaner and reconciler and the performance meter and aggregator, there have been some challenges. In particular the data cleaner and reconciler has to perform the Extraction, Transformation, Loading process because it has to standardize data from databases of different WfMSs to enable analyses on data. Instead the performance meter and data aggregator has to aggregate stored data into the data cleaner and reconciler database to gain information and the data aggregation is performed according to metrics; for this reason it needs to have a schema that can easily have additional fields, which will contain new metrics values. Thus, the faced challenges are:

- for the data cleaner and reconciler:
  - allowing reverse mapping from clean data to source data;
  - managing the integrity of data;
  - managing the possibility to interact with different types of database.
- for the performance meter and data aggregator:
  - having a schema that can easily have new additional fields.
- for the proof of concept demonstration:
  - defining workload intensity.

1.3.1 Data cleaner and reconciler

1.3.1.1 Reverse mapping

The reverse mapping regards the possibility of mapping from clean data to source data. So that, in case it is required, it is possible to observe the source data. Moreover since only some data are cleaned, it is required to allow a more in-depth analysis.

1.3.1.2 Data integrity

Data integrity regards consistency and accuracy of data. In particular during data cleaning the meaning of the data must be preserved.

1.3.1.3 Database interaction

The data cleaner and reconciler has to interact with many different source databases. Source databases might be based on different Database Management Systems (DBMS). This means that the interaction with source databases might require using different SQL dialects and the cleaner has to manage different database schemas.

1.3.2 Performance meter and data aggregator

1.3.2.1 Schemaless database

The performance meter and data aggregator database needs to have a schemaless structure, where new additional fields, which will contain new metrics values, can be added easily.

1.3.3 Proof of concept demonstration

1.3.3.1 Workload intensity

Workload intensity must be determined representing as much as possible the interactions that a WfMS is subject to.

1.4 Thesis structure

The rest of the thesis is organized as follows: Chapter 2 illustrates the State of the Art of Business Process Modelling, the Business Process Model and Notation, the Workflow Management Systems architecture and its effects on the various components and the benchmark framework. Chapter 3 describes the BenchFlow framework and its components. Chapter 4 identifies the Workflow Management Systems metrics applying the Software Measure Definition Method and it defines the implemented metrics. Chapter 5 is about the extraction, transformation, and loading of performance data. It illustrates the design implementation detail of the data cleaner and reconciler and it evaluates the amount of managed data, correctness, reverse mapping of the data cleaner and reconciler and it analyses the scalability to different Workflow Management Systems. Chapter 6 describes the design and implementation details of the performance meter and data aggregator and it evaluates their performance and scalability. Chapter 7 applies the defined metrics on two different WfMSs and performs one trial of load tests. The workload of the tests are based on two different clusters, each one containing two process models. Thus, it demonstrates the feasibility of the analysis through the application of the two developed tools. Chapter 8 concludes the thesis and states future work possibilities.

Chapter 2

State of the Art

2.1 Business Process Management

2.1.1 BPMN introduction

"Each company has business processes, which are focused upon the production of particular products. These may be either physical products, such as an aircraft or bridge or less tangible ones such as a design, a consultation paper, or an assessment. In other words, the product can also be a service" [96].

Indeed according to Rummler and Brache business process is "the series of steps that a business executes to produce a product or service" [86]. So each company has business processes, which are a composition of activities connected and correlated to each other. These business processes are modelled to acquire inputs and transform them into specific output.

In particular the Business Process Management (BPM), is a management discipline that deals and manages business processes into a company. The purpose of BPM is to provide to companies a tool that defines, executes and controls business processes so that companies might reach their business strategies.

Then the BPM supports company performances aligning strategies and management and productive processes, such as processes that manage production, innovation, management of financial resources, human resources and clients. Indeed BPM systems offer advantages in terms of cost reduction and earnings increment; this advantages are increased when BPM systems support very complex and critical processes for companies that interact with a very high number of person that belongs to different functions.

Business processes, usually, are not expressed formally and they are implicit into companies. BPM system is a software system that defines business processes using notations that allow to execute and to control process instances. Companies have to collect all information regarding business process to allow that BPM supports the real business objectives. Moreover different users of companies must be able to understand the notation used to model processes. This

should occur also in case that user belongs to different function and area of company and even if users have different knowledge [44].

In the past, notations used to processes modelling could be very different. In the last years those notions have been reduced, reaching a limited numbers of standards [44]. The beginning of the standardization of the modelling notations started with the book "*Improving performance*" written by G. Rummler and A. Brache in the 1990 [86]. In this book the authors represent some BPMN notations concepts. Before 1990 the models made by the business users were technically autonomous from the ways of representations used to implementing them.

2.1.2 BPMN standard

The use of standards allows different type of users to understand models realized by other people; the different understanding of the same model, by different type of users, highlights that a model can be read from different points of view. Indeed, the processes modelling are based on how user, who realises the model, take part at it and on how user knows and understands it. Anyway, in general the process model should be able to represent all the different points of view because whoever has a part in the process should be able to see its activities and other activities that do not belong to him/her but are made during the process.

The main standards of a process can be labeled as:

- Graphics standard: it responds to the requirement of simplicity; the process should be easily understood with a graphic representation;
- Interchange standard: it responds to the requirement of changing the executions platform without adapt the process to the new platform. The main characteristic of the interchange standard is the common format used to represent the processes;
- Execution standard: it responds to the requirement of process execution feasibility on the BPM system. It is characterized by a semantic and syntax that allow the execution of the represented process.

In 2004, after two years of work, the Business Process Management Initiative (BPMI), has published a standard: the Business Process Modelling Notation (BPMN) 1.0. The main objective of BPMN is to deliver a notation readily understandable by all type of users. The most relevant category of users are:

- process model analyst,
- developer,
- process responsible,
- process executor.

The work of process model analysts is to identify and to represent process models. Developers have to implement the technologies that will perform those process models defined by analysts. Finally process executors execute the process model activities, while process responsible manage the execution of the processes and monitor their progress [101].

Thus, the BPMN tries to create a language that standardises the gap between the business process design and the process implementation decreasing the existing fragmentation between the enormous amount of notations and the process modelling tools. All this is possible due to its wide use in the sector.

For the following years the BPMN is run by Object Management Group (OMG), after the merge between OMG with BPMI. Today the last version is the BPMN 2.0 released by OMG in 2011, where BPMN stands for Business Process Model and Notation. Nowadays, even if there are a lot of processes modelling standards, one of the most common and most used by the BPM tools producers are the BPMN standard [23].

In the BPMN the Business Process Diagram (BPD) is defined as a flowcharting technique customized for realising graphical models of business process operations. The BPMN is developed with the goal to supply an easy method to model the business processes and be able to handle the complexity of possible processes [101].

Another important objective of BPMN is to allow the modelling processes execution. To reach this goal the BPMN has a correspondence between the graphics elements and the execution semantics, namely that a semantic XML execution can formalize all the BPMN elements. The BPMN execution semantic specifies how each element should be translated to perform in a BPM system. The use of XML standard supplies also the possibility of interoperability between different BPM systems. Indeed the process models can be interchanged among different BPM systems.

2.2 Workflow Management System

Workflow Management System (WfMS) is a system for the processes execution management, it especially runs workflows. Workflows are business processes with components that automatize completely or partially the process. In the Workflow Reference Model [49], the Workflow Management Coalition (WfMC) defines both the workflow and the Workflow Management System:

- "Workflow: The computerised facilitation or automation of a business process, in whole or part."
- "Workflow Management System: A system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic."

The Workflow Management Coalition is a consortium founded with the aim to supply standards for the processes and for the Workflow Management Systems to allow their interoperability. In 1993 the consortium was founded by a group of various subjects, among them there were companies of the sector such as IBM, HP, Oracle and Sun Microsystems.

Nowadays, due to the changing market condition and enterprises requirements, a system for the processes execution management needs to support more complex business processes. According to K. Hayes and K. Lavery [47] the need of controlling, monitoring and managing the business process creates the term workflow. Indeed the first Workflow Management Systems are introduced in the market to help the enterprise to automatize completely or partially their

processes in the eighties; before there were not generic tool to support WfMS. Currently a lot of vendors are offering WfMSs; see Figure 2.1 for the WfMS history.

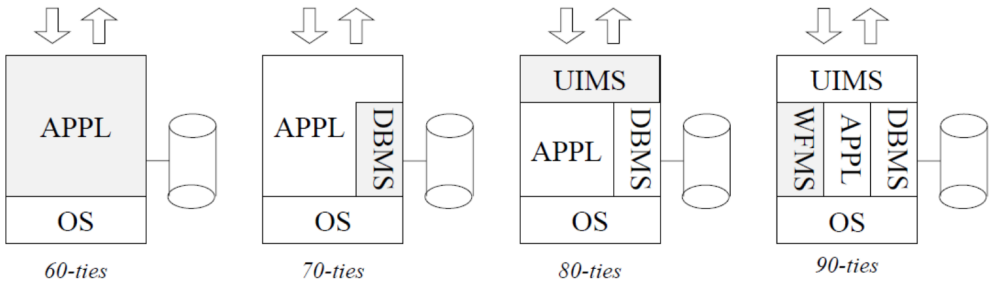


Figure 2.1. System architecture history [95, pg. 162]

The WfMSs coordinate the activities conduction so that they execute the flow described by process models. Thus the WfMS allows the process purpose achievement. Anyway the process or workflow representation has to specify some additional characteristics to allow its control and coordination.

In the majority of the processes, the process has to be executed by the various participants. Thus it is necessary that the WfMS coordinates the various participants' jobs. Indeed each process's participant has to carry out activities to allow the process execution's continuation. Moreover, the WfMS has to consider that the participants could be people or machine and therefore they require different management methodologies.

2.2.1 WfMS functional division

The Workflow Management Systems have mainly three functional areas [10, 49, 82] as it is shown in Figure 2.2:

- Build time functions;
- Run time functions;
- Run Time interactions.

2.2.1.1 Build time functions

They concern the modelling functionality of the processes and their activities. They are used mainly by processes designers. In this phase predetermined WfMS tools' are used to create the workflow.

2.2.1.2 Run time functions

They concern the process execution management of the model. They ensure the defined flow and they manage activities and requirements that the activities request such as input data.

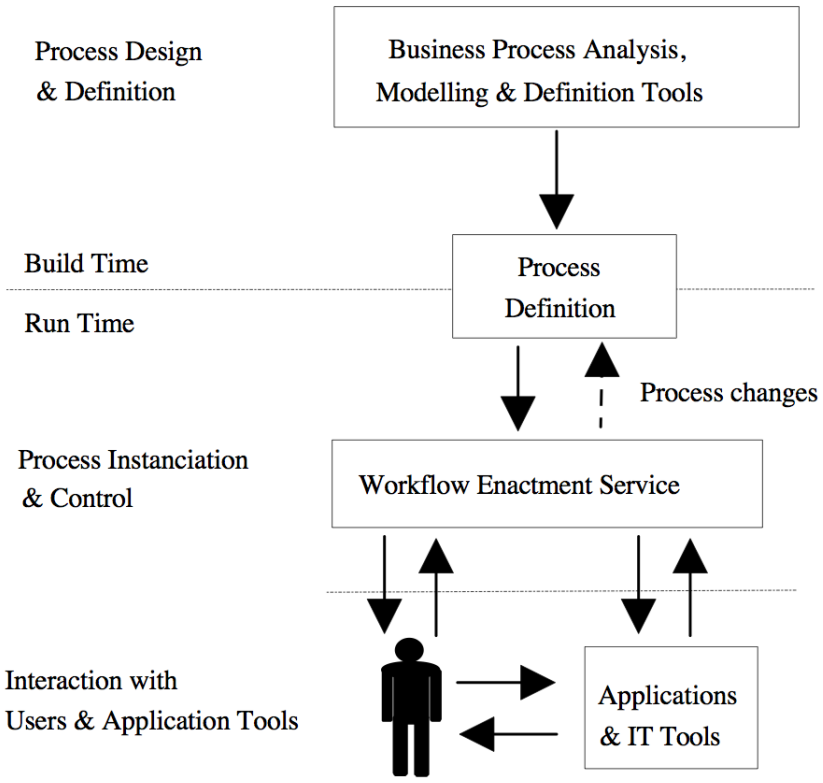


Figure 2.2. Workflow Management System functional division [49, pg. 7]

Moreover they manage the execution restore and they store history execution data.

2.2.1.3 Run time interactions

They concern the management of the WfMS external interactions. The external interaction can be:

- Interaction with the user;
- Interaction with web services.

Those two interactions need interfaces:

- User interface;
- Application interface.

Moreover interactions, with users and with web services, need managers. The user manager is called Worklist Handler. Finally the interactions are defined by processes designers in the process through build time functions.

According to the build time and run time functions the WfMS architecture has two key components:

- Definitions tool
- Workflow enactment service

The workflow enactment service's main component is the Workflow Engine (WfE) with a collection of other components that allow the correct WfMS operation.

2.2.2 WfMS architecture

The Workflow Management System architecture could be very different according to the used WfMS. Anyway every WfMS implementation has to be built with components that manage common issues.

According to the Workflow Reference Model of Workflow Management Coalition, as it is shown in Figure 2.3, the WfMS architecture is composed by:

- Definition Tools or Workflow Modeller,
- Workflow Client Applications or Worklist Handler,
- Invoked Applications,
- Workflow Engine,
- External Workflow Engines,
- Administration & Monitoring Tools.

Moreover there is the DBMS component.

2.2.2.1 Definition Tool or Workflow Modeller

It is used to model the workflows that will be executed. It uses a notation that is understood by the WfMS. Usually, if the WfMS language is BPMN the definition tool allows graphics representations of the process model in addition to the XML language.

Moreover the definition tool does not have to be specific for a WfMS if the language used is an interchangeable standard and the WfMS allows that language. This is the BPMN case, even if it depends from how the specific WfMSs are implemented compared to the standards.

The workflow realized by the definition tool must be executed, so it has to be written in a language which is supported by the WfMS and it has to contain all the information needed for its execution.

2.2.2.2 Workflow Engine

It mainly provides the environment to execute workflows and it must be able to maintain multiple workflows. In practice it is a service that enables and enacts the run time functions and

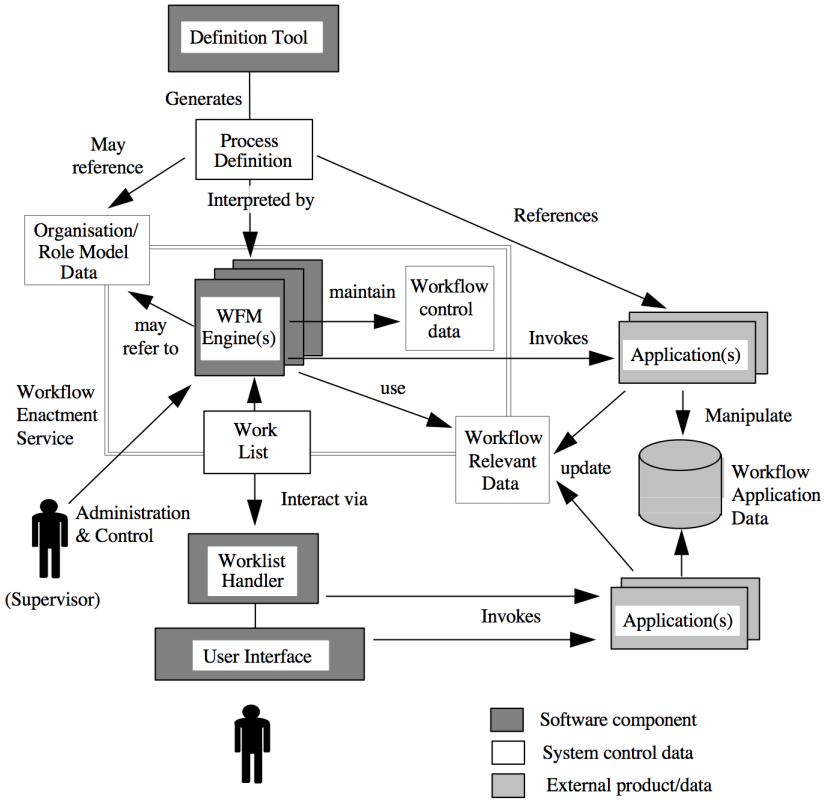


Figure 2.3. Workflow Management System components architecture [49, pg. 13]

its main functionalities are to:

- execute process models by interpreting them;
- manage process instances;
- interact with external resources;
- allow process monitoring.

Therefore it has to manage interactions that the process instances require and to ensure the correct execution of process instances according to process models by managing resources, communication and information.

Moreover it is possible to have different Workflow Engines to handle runtime functions. Workflow Engines can work as centralized systems or as distributed systems; moreover it can interact with external Workflow Engines. Indeed it is possible to have inter-business workflow instances. In particular having sub-process instances among different Workflow Engines is a consequence of interoperability among them.

2.2.2.3 Invoked application

Invoked applications are web services that are external to the WfMS and that are called according to process model requirements. To call Invoked application is necessary to know:

- application location,
- parameter required by the invoked application to be executed.

Moreover it is necessary that there is a suitable interface that allows the interaction between WfMS and invoked application. Usually standard interfaces are used, indeed interoperability is a key characteristic of WfMSs.

2.2.2.4 Administration & monitoring tool

It is used by administrators to manage the Workflow Engine and to audit functions. In particular it allows to monitor workflow instances by accessing WfMS logs and database data. Moreover it might be able to perform statistical analysis and to manage workflow instance termination and restart in case of failure.

2.2.2.5 Worklist Handler

The WfMS might have users that execute some tasks of processes. The WfMC has decided to separate the user interaction model, therefore Workflow Engine delegates to Worklist Handler the management of the tasks which require interaction with the users.

Thus, the Worklist Handler collects all the tasks which have to be executed by users and it manages the tasks dispatch and allocation to users. Moreover the Worklist Handler keeps additional data for each task, such as the task deadline, the necessary data for the task execution and the data that the task has to produce.

The dispatchment of the task that should be executed could be based on priorities decided by the Worklist Handler or by the user's discretion. The Worklist Handler complexity depends on how the WfMS is developed.

2.2.2.6 DBMS

During the processes execution the WfMS has to maintain progress state of processes execution, therefore it has to store data. The WfMS data storage has to happen for different reasons such as process execution development, the conservation of data which will be used for monitoring, execution reinstatement of all the process in case of system failure. It might maintain, also, other information which are about the WfMS execution configuration, data history of previous executions and so on.

Moreover the stored data could be used to understand the process execution performance on the specific WfMS and if problems happened inside the WfMS. Finally, beyond the process data storage, also the executed processes models have to be saved to allow their instantiation in future.

The DBMS allows separating WfMS job from the data storage management and usually the DBMS used is a relational database [66]. Moreover there is a remarkable literature about DBMS and they have reached a notable state of art. For this reason, the DBMS usage allows a better efficiency given by the considerable experience and literature in this sector. Moreover, some WfMS problems are solved by the DBMS, such as the concurrent accesses to data.

Anyway the DBMS has to be configured according to the used WfMS. The configuration should happen by both ways: DBMS and WfMS. Certainly the WfMS must have the components to interact with the DBMS, such as drivers and connectors. Whereas the DBMS must be configured to contain the tables required by the WfMS and respect some limitations such as data consistency or data access.

2.2.3 Advantages of WfMS

WfMS embraces a set of advantages. Firstly its business processes automation entails cost and time reduction to execute these business processes and efficiency increase by decreasing the unnecessary or inefficient steps and it also causes the reduction in the time required to execute a process instance. Moreover, WfMS increases productivity and customer services due to the less uncertainty in the processes execution. Furthermore, it improves flexibility because in general process redesign does not require expensive reconfigurations. Finally, WfMS increases processes control due to processes standardization and control tools implementation and it supports decisions planning due to the fact that it improves the access of information to the management.

2.3 Benchmark framework

2.3.1 Performance tests design

To design a test is very important to know and to understand the context of the test, so that the test can be focused on relevant aspects [67]. Indeed state of the art regarding BPMN and Workflow Management System were discussed so far.

There are different types of performance testing for system under study [67]:

- Basic performance test or baseline test [85]: it sends to the system only one request when the system is not executing any request. Since baseline test check system characteristic during underload system period, this test usually provides best case characteristics. Best case performance characteristics are never reached in practice, but they can be used as base to compare results of other test's types. For example a metric such as process execution delay, which is the difference between process execution time and minimum or ideal execution time, can be computed as difference between process execution time provided by load test and process execution time provided by baseline test. Moreover it might be applied to set requirements on system characteristics providing as base the best case system characteristics.

- Load test, it sends to the system a volume of requests that is expected to match real requests volume. Which means that a system has to handle normal and peak loads. Load test is used to determine performance characteristics of the system under real workload. A particular type of load test is the endurance test, which is similar to load test, but it is executed for a longer time interval. The purpose of the endurance test is to understand possible problems that might arise using the system as a long term solution. Therefore it might show also performance degradation. Load test might identify if the system is not able to manage normal application loads. Indeed if workload is too high for the system under study, load test might expose system performance problems that are usually showed in stress test.

- Stress test, sends to the system a volume of requests that exceed the working condition of the systems. The request volume is above both normal and peak load, where peak load is the maximum number of requests that the system receives. This type of test is used to identify possible system bugs on extreme load situation; it shows conditions that might cause system failure, so that indicators of system failure are monitored, and it shows which are components that will fail. Finally it expresses how much a system can go beyond its actual utilization. A particular type of stress test is spike test, which overloads the system in a short period of time with a high volume of requests.

2.3.2 Workload performance benchmark

Workload is used to study system behaviour. Workload should represent an interaction with the system under study by means of workload mix and volume of requests. In addition workload could include also users and data. In Workflow Management Systems, workload describes which and how processes are executed.

The workload mix is represented as a set of process models that are executed on the WfMS under study. Each process models used has different characteristics with respects to the others. Workloads of software performance benchmark can be classified in four categories, as it is shown in Figure 2.4 [68]:

- Basic operations performance benchmark, it is a *synthetic performance benchmark workload*, so they do not represent any possible real request. Therefore they are especially used to test specific isolated characteristics as micro-benchmark. Anyway their usage is very limited due to their limited possibilities to provide meaningful insights on real systems.

- Toy performance benchmark workload, it represents a possible request to system, therefore it might provide insights on the system under study. Anyway they are not very useful to predict real application performance, because it is difficult to understand possible performance of real problem workload given proof of concept workload. Indeed toy performance benchmark workload do not analyse any real problem requests.

- Kernels performance benchmark, it represents requests extracted from real requests to running systems. Since it focuses only on most important part of system and on the majority of real requests, performance information relevant for end users might not have a high degree of precision and accuracy with respect to actual performance.

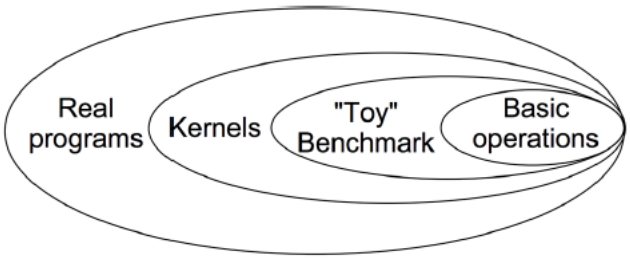


Figure 2.4. Types of performance benchmark [68, pg. 266]

- Real programs performance benchmark, they provide real requests to system used in real world applications. This type of performance benchmark workload is the most accurate one, even if it is specific for the real world application studied.

In Workflow Management System, basic operation performance benchmark workload is given by requests defined by process models that study single BPMN elements. For example a basic operation process might be a process that contains message or timer event, or even a process that contains a gateway connected to some activities. Therefore this synthetic performance benchmark workload does not constitute a real performance test, but it can be used to understand Workflow Engine capability to execute BPMN elements.

Toy performance benchmark workload uses process models which correlates different BPMN elements without a real application. Process models might be also really complex. Potentially a toy performance benchmark workload might use a process model that contains all possible type of BPMN elements, but this kind of process is very unlikely to be executed on real applications.

Kernel performance benchmark workload is composed of processes that are derived by analysing real processes and by applying their main characteristics. In practice it uses process model patterns.

In WfMS, real program performance benchmark workload uses process models that might be obtained by businesses. Anyway it is very difficult to find businesses, which are willing to provide their business processes to test. Indeed business processes might expose insights of company competitive advantage to competitors. However when business processes are not available, there are also public domain processes that might be used. For example *BPM Academic Initiative* provides a set of publicly available processes.

2.3.3 Workload intensity

When workload is determined, it should be determined also the workload intensity at which workload is loaded to the system under study. Usually workload generator is located on a different system, so that there are no interference between the workload generator and the studied system. For example the studied system might drain resources from workload generator and it might alter how many request the system has received.

So there are different ways to load the workload [60, 85]:

- with arrival rate: arrival rate is the mean number of requests that arrives to system per time unit. So it is a rate at which system receive requests. By definition arrival rate $r(t)$ at time t is expressed as:

$$r(t) = R'(t) \tag{2.1}$$

  where $R(t)$ is the number of requests u_{t_0} with arrival time t_0 and that are arrived until time t

$$R(t) = \text{ count}(\{u_{t_0} | t_0 \leq t\}) \tag{2.2}$$

  The arrival rate can be:

  - variable: variable arrival rate might be used to perform spike test. It could be also used to perform stress test by increasing arrival rate until the system does not respond any longer or a certain performance threshold is achieved. Finally variable arrival rate is used when system trace of requests is available, so that it allows to simulate real request behaviour in system under study.
  - constant: constant arrival rate can be used for both load and stress tests. In stress tests, it is used a constant arrival rate extremely high to check if the system is able to manage such generated workload. In load tests, constant arrival rate should almost match real request arrival rate to estimate system performance.

- with continuous injection [70, pg. 55]: requests arrive to the system continuously until a plateau is reached and the number of requests remain constant. Continuous injection is most suitable for concurrent user consideration, indeed it considers that the requests to systems are performed by users that progressively decide to perform action on system until the normal number of user is reached. Therefore the number of users that interact with the system and their requests increases gradually. However there are different possibilities to reach the plateau; the stage before the plateau is usually called ramp-up and it could be characterized by how it allows to reach the plateau stage according: stepwise approach or big bang approach. Big bang approach considers a little ramp-up stage in which all user and their respective requests are injected all at once. While stepwise continuous injection gradually increases the number of users by steps.

2.3.4 Database benchmark

The idea to evaluate and to compare relational database goes back to a paper written by J.Gray et al. in 1985 [16] and Gray's following paper written in the nineties [42].

The studies about the performance database benchmark are flourishing. Indeed as the database technology evolved and new type of databases were created, also their benchmark is produced according to the new databases. A few examples of benchmark proposals for different type of databases are:

- Object oriented database benchmark [21];
- XML database benchmark [73, 103];
- Stream data management system benchmark [11];
- NoSQL benchmark [20, 93];

- database management system benchmark [17, 88, 92].

2.3.5 WfMS benchmark

There are few information about the performance of WfMSs [29, 41, 43] and the few empirical researches about WfMSs do not focus on their performance, whereas they are, for example, about:

- the metamorphosis of the project objectives [48];
- appreciation of the technology by final users, [62];
- their implementation [76];
- the reduction of WfMS lead time [74];
- the efficiency of individual structural activities and the different service oriented middlewares performance comparison [15];
- the impact of database work on the server and systems throughput [41];
- the impact of the database on the workflow [17].

The study that could be considered as a performance evaluation of the WFMSs is the "engine performance evaluation by a black-box approach" by Daniel, F., Pozzi, G., Zhang, Y. [29]. Their approach, called black box, compares the performance of five WfMSs, comparing their constituent elements. Indeed in the study the WfMSs are installed considering default configuration in their respective five different virtual machines.

Even if there are few empirical researches there is a common understanding about the need of having a WfMSs benchmark. Indeed many authors recognize the necessity of a WfMS benchmark [58, 64, 100]. The lack of a performance benchmark is present also among BPMN 2.0 WfMSs; even if there is a recognized necessity for a BPMN 2.0 WfMSs benchmark [100]. The BPMN 2.0 WfMSs benchmark is particularly important due to the rapid spread of the BPMN 2.0 standard. Indeed at the end of 2014, almost 20 Workflow Engine systems were supporting BPMN 2.0. [87].

In this context of lack of a performance benchmark among BPMN 2.0 Workflow Management Systems, the BenchFlow [13], project is born; due to the importance to design and develop a benchmark for assessing and comparing the performance of Workflow Management Systems. Indeed, the first benchmark with this goal is the BenchFlow, that plan to use a model-driven, self-/recursive testing approach to eliminate the impact of the external services by having them implemented as processes. Indeed the BenchFlow considers:

- the number and heterogeneity of the WfEs under test,
- the growing complexity of the workload mix, and
- the type of performance test that will observe a broader spectrum of raw performance metrics and aggregate them into meaningful KPIs. [87]

Chapter 3

BenchFlow framework

Given the growth in number of Workflow Management Systems [37, 97], particularly the ones supporting BPMN 2.0 [87], a benchmark of WfMS BPMN 2.0 is recognized as necessary [100]. BenchFlow [13] is the first BPMN 2.0 Workflow Management Systems benchmark. As a matter of fact, even if similar tools exist for benchmarking WfMSs [14], they do not focus on BPMN 2.0 Workflow Management Systems performance evaluation and benchmark [46], as discussed in Section 2.3.

In this context the Università della Svizzera Italiana (USI) and the University of Stuttgart have started to develop the BenchFlow project with the goal to provide a benchmark framework for WfMS BPMN 2.0. Undeniably having a benchmark framework would fix the boundary condition around the WfE and allow the measure and the quantitatively comparison of WfMSs. To do it one of the most important requirement of the BenchFlow framework is the elimination of the impact of external service and components. Thus, BenchFlow is a model-driven, self recursive testing approach that eliminates the impact of the external services by having them implemented as processes [78]. In point of fact, the BenchFlow framework has an efficient and flexible architecture to assure the quality of the benchmark. For instance the flexibility is offered using Docker [3], which is a tool that exploits different hardware resources in a flexible way through lightweight virtualization and assure a good level of isolation, compatibility, configurability and quick start up [87].

In particular the BenchFlow requirements [87] are:

- a model driven approach;
- a flexible deployment mechanism;
- a flexible way to configure different hardware resources and switch between different configurations;
- the best level of scalability to new WfMSs, so that the effort of adding a new WfMS is reduced;
- frozen initial condition, which means that the initial state of the different WfMS components is the same for every benchmark execution;

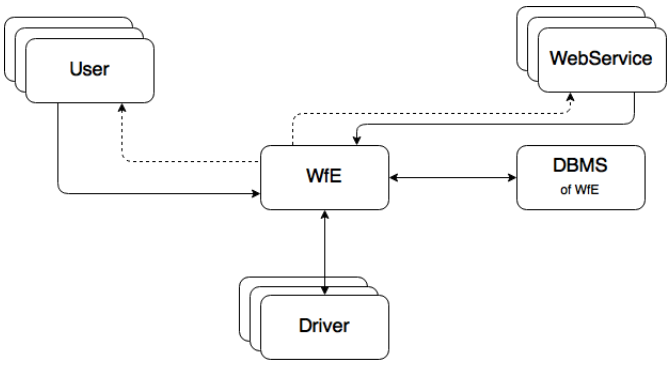


Figure 3.1. System under study

- reliable and non-intrusive data collections;
- flexible and extensible metrics computation.

The BenchFlow framework is a set of services (see Figure 3.1); the choice to use services has been made because these services must not interfere with the normal execution of components of the system. Undoubtedly, it is important for companies to have a framework that allows the data retrieval in a way that the framework has the least impact on the processes execution. Moreover if this is not the case the framework can influence the performance of the targeted system, and therefore be less useful in taking an informed decision.

As Figure 3.2 shows, the framework components are:

- Driver;
- Monitor;
- Collector;
- Data cleaner and reconciler;
- Performance meter and data aggregator.

The driver manages set-up configurations, the workload, the functioning of the whole system during execution. It is the component responsible of the infrastructure and it allows guaranteeing the BenchFlow framework flexibility, model driven approach and switch with past configuration.

The monitor observes the workload execution and, without interfering, tries to understand when the system has finished executing the workload. The monitor is a service formed by a distributed system of many monitors, that in this thesis is going to be called with the generic term monitor.

The collector, as the name implies, collects the data generated by the system components, specifically it retrieves log files or database records. It is utilized after that the monitor has ensured that the WfMS has completed all requests sent by the Driver. The collector is a service formed by a system of many collectors, that in this thesis is going to be called with the generic term collector.

The data cleaner and reconciler standardizes data extracted from databases of different Workflow Management Systems and it ensures the best level of scalability with different WfMSs. Moreover it stores also data about the environment.

Finally the performance meter and data aggregator structures and organizes the WfMS data and perform the WfMS metrics computation.

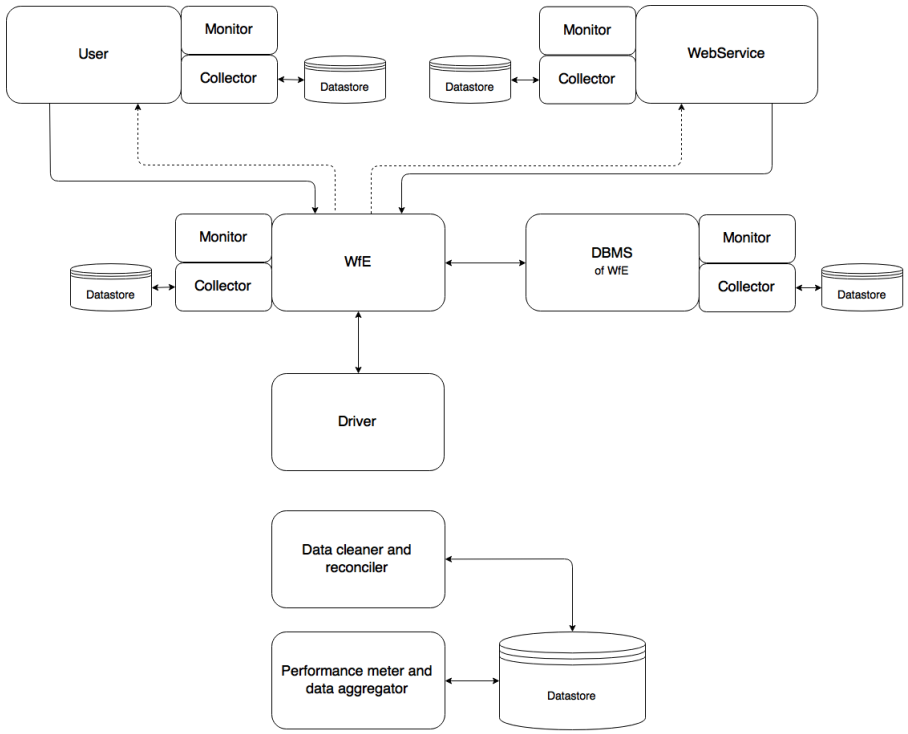


Figure 3.2. Framework components and the system

3.1 Driver

The driver ensures that the starting states of the different components are the same for every workload execution by using Docker containers [3]. The driver is part of the benchmark environment and it is responsible of the infrastructure. In particular it manages set-up configurations, the workload, the functioning of the whole system during execution, the clean up when the execution is finished and so on. The driver is the backbone of the infrastructure and it triggers all the system components; moreover it can manage both synchronous and asynchronous workloads. In addition, the driver writes logs, which report information such as: request errors and response time of API calls, that might be useful to understand the functioning of the system.

3.2 Monitor

The monitor has to understand when the WfMS has finished to execute the requests sent by the driver and it must fulfil its duty trying not to interfere with the requests execution on the WfMS. In particular each architectural component of the WfMS has attached a monitor, which controls the component execution status. The monitor uses an indirect and low demanding resource request to control the current execution status of the requests execution. When this request shows the possibility that the execution is finished, the monitor has the possibility to use a direct and more demanding resource request to have a more accurate reply.

The monitor should start to perform its duty after that the driver has finished to send requests to the WfMS. To reduce interferences, the monitor interacts with the WfMS according to a two phases interaction:

1. During the first phase the monitor interacts with the environment to understand the resource utilization of the component to which it is attached. This means that it checks the CPU utilization and network utilization to avoid a direct interaction with the WfMS component, which might still be executing process instances. So in this phase, the monitor objective is to interfere as little as possible with the WfMS execution and to understand with high probability when the WfMS has finished to execute requests. When the monitor sees low CPU utilization and low network utilization for a long enough time, the second phase of interaction can take place, because the Workflow Management System might have finished executing process instances, but this condition cannot be ensured in this first phase.

2. During the second phase the monitor directly interacts with the Workflow Management System components to understand if there are some process instances that are still under execution. This interaction does not necessary require to interact directly with the Workflow Engine; indeed it is possible to check the execution status using WfMS database, due to the fact that the database usually contains all information about executed process instances and process instances under execution. In this second phase, the monitor objective is to ensure with higher accuracy the completion of the execution requests sent by the driver and this phase is more resource expensive than the first phase, because the monitor interacts directly with the WfMS components and in particular the WfMS database.

In addition the monitor has to check that WfE can run without interferences caused by problems of other WfMS components, such as DBMS, Web Services and User interfaces. In particular problems might arise in case that these components have a high number of job requests in their queues, because this leads to a high response time degradation and causes a negative influence to the whole performance of the Workflow Management System [63]. Moreover, in the worst case scenario, WfMS components might manage long queue rejecting new job requests or, even worst, losing job requests already in queue. This could cause the WfE inability to complete process instances and the monitor should be able to recognize this situation.

3.3 Collector

The collector is used to collect the data generated by the system components, in particular by retrieving log files or database records. It is performed after that the monitor has ensured that the WfMS has completed all requests sent by the driver. In particular, when the driver has received a positive reply about the completed requests execution from the monitor, it asks the collector to retrieve the useful data.

The collector knows where WfMS components store their data on the file system or database. Indeed, whenever it is possible the collector has to avoid to interact directly with the WfMS components, because the API interface, exposed by the WfMS, might limit the collector. For example the WfMS can expose an API method about retrieving completed process instances, but it might happen that if there are too many completed process instances, so the API method might not be able to retrieve them all. However, in some cases the collector has to interact with the system components, for example it has to interact with the Database Management System to collect database data. Moreover, it must be ensured that the collector is not able to modify data, therefore it has only to read them and it must have only the read privilege when it is possible to specify it.

Finally the collector has to write data read to a new location, which contains data from all WfMS components related to the same experiment trial. Moreover data must be categorized and organized according to their origin component, in other word their provenance, so that it is possible to determine which data are related to a specific system component.

Therefore the hierarchical organization of the new location is:

1. experiment trial;

2. system component.

The most important data collections performed by the collector are those that allow to determine the WfMS metrics. One of the most important task of the collector is the gathering of data from the WfMS database, because it contains data about Workflow Management System executions. Other important data collections occur when the collector collects data from the driver and from the environment, because driver data allows understanding the interaction with the WfMS and the workload it were subjected to during the experiment.

3.4 Data cleaner and reconciler

The data cleaner and reconciler has the duty to clean and reconcile data collected, thus its objective is to allow standardizing data from databases of different Workflow Management Systems. This standardization is required because different Workflow Engine stores relevant data for analysis with different data structures and different notations. In addition the cleaner and reconciler loads statistics from environments. For more information refer to Chapter 5.

3.5 Performance meter and data aggregator

The performance meter and data aggregator has to aggregate stored data into the cleaner database to obtain information. Data aggregation is performed according to metrics application based on the experiments and on the trials related to a specific experiment. Thus, it has to structure and to organize the data in a way that they can be understandable. For more information refer to Chapter 6.

Chapter 4

Metrics

According to Fenton and Pfleeger [36], a measure is defined: "as a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute." and a metric is a measurement function [7].

Metric is a quantity that can be used as a measurement standard. Metric, in particular, focuses on attributes of the metric context and, to apply a metric, it is necessary that the metric is clearly defined. Metrics are used to identify possible improvements and to monitor progress. They provide information to support quantitative managerial decision-making during the software lifecycle [89] and they will provide objective evidences to take decisions upon.

The thesis scope is performance metrics of Workflow Management System and about the definition of the main WfMSs metrics, they are identified applying Software Measure Definition Method (SMDM) [19] and they arise from the paper "A Framework for Benchmarking BPMN 2.0 Workflow Management Systems" [37] written by Vincenzo Ferme, Ana Ivanchikj and Cesare Pautasso.

To identify metrics there are two questions that can be asked:

- what to measure?
- how to measure?

At first to identify metrics [80], the question should be: "what to measure", instead of "how to measure". Indeed the how question might remove some valuable metrics from being considered. Moreover the how question focuses on data collection, which affects metric costs. Therefore after that relevant metrics have been identified, the how question should be answered to identify which metrics are cost effective. In this study, for the "what to measure" question is used the Software Measure Definition Method [19] methodology and the "how to measure" question will be treated at one point of the methodology.

The Software Measure Definition Method is used to determine metrics in the Workflow Management System context. This methodology is based on four steps [19]:

1. metric definition: it is about determining metrics considering characteristics of the context and experience with the context of users, developers and modellers;
2. theoretical validation: tries to determine if the metric is valid from a measurement point of view. So theoretical validation checks that metrics measure properly characteristics of the context;
3. empirical validation: it uses experimental data to validate metrics and to understand metrics validity in practice;
4. psychological explanation: it wants to describe how metric influences subjects that deal with the context. Moreover psychological explanation might be combined into theoretical validation.

Metric definition is realised by two main activities [83]:

1. Identification: is the most important activity, because the following activities and following steps of metric definition are influenced by this one. Therefore it is required to apply an accurate process to perform this activity. Identification activity has to establish measurement goals, questions, abstractions and hypothesis. At this stage it is important to focus also on available literature;
2. Creation: formally define metrics with clear specification.

The SMDM suggests using the GQM framework for the metric definition step by following a structured process. Indeed GQM framework focuses on [12]

- Goal;
- Question;
- Measure.

Goal and question concern the identification activity, while measure consists of the creation activity. However SMDM provides an extended process to reach the final metrics, which consists of [83]:

1. determine entities in the context of study;
2. determine on which external properties of the entities to focus on; external properties can also be called quality attributes. Indeed attribute is a detail of the property. For example some of quality attributes are: accuracy, availability, compatibility, extensibility, maintainability, responsiveness, scalability;
3. determine the goals according to the GQM framework; in particular by determining the objective to measure quality attributes of entities. In other words, GQM goals detail what objects (in this thesis the objects are the entities: WfE, process and construct) are measured for what purposes. Moreover, it must be defined also a point of view for the whole analysis;
4. determine which internal attributes of the entities should be measured;
5. determine abstractions to measure the attributes;
6. determine the questions from the goals;

7. state hypotheses which identifies relationship between external properties and internal attributes of entities.

Continuing from the identification activity to the creation activity, the following step is to define metrics by providing proper and clear definition. In particular it should be stated:

1. metric's name;
2. metric's goal: it must be coherent with the identification step and GQM framework;
3. metric description of what metric indicates;
4. how compute metric values. In particular formulas might be provided.

The theoretical validation can be performed by applying the Doran's [32] SMART framework, where metrics

- specificity: metrics should be goal oriented and metric's goals should be specific to the metric context. Moreover metrics should be understandable by workers of the metric context. Indeed metrics should be well written by specifying assumptions and definition, so that they can be well interpreted;
- measurability: metrics should be quantified in order to compare metric results. Workers of metric context should be able to monitor the context to perform, e.g., corrective actions;
- attainability: metrics should be credible and realistic. It should be possible to compute the metrics and metric data must be collectable. This characteristic responses to the "how to measure" question;
- relevance: metrics should be important in the metric context. Importance can be estimated in relation to the metric context objectives;
- timeliness: metric should be ready when you need them. So a predetermined time bound should be applicable considering execution time requirements.

The empirical validation is obtained by testing the derived metrics applying the benchmarking system according to different experiments trial and by studying the results of these trials. Thus, empirical validation is achieved by obtaining empirical evidence of the metrics in practice thought experiments.

Finally metrics are usually distinguished between performance metrics and quality of service metrics. Quality of service metrics are not going to be analysed in this thesis and they should be considered in future works; for example the thesis does not analyse the quality of services regarding requirement such as the security of the WfMS or WfMS portability or the level of documentation available.

4.1 Metric definition process

4.1.1 Metric identification activity

Metric identification requires to determine a point of view for the analysis. What is applied in this thesis is the point of view of the user. The user is intended as client of the WfMS service

who instantiates processes; so the user is the final user that receives the process outputs.

In Workflow Management Systems context, it is possible to identify mainly three entities [8, 37].

- Workflow Engine, which executes process instances that are requested by the user. It is defined by a specific architecture that conditions its process instance execution performance. Moreover the Workflow Engine is executed on specific machine and interacts with external environments, which constitute Workflow Engine context. Furthermore the Workflow Engine is a software component.
- Process, it is the process model that is executed by the Workflow Engine when user instantiates it. Usually process models are incident to a specific sector, so it is possible to classify processes according to their belonging sectors. This characteristic of process model can be identified as process scope and it is implied in Process entity. The Process entity requires a Workflow Engine entity that executes its executable part.
- Construct, it is a component of process model. Referring to Business Process Model and Notation, construct identifies BPMN elements: flow objects, connecting objects, swimlanes, data and artifacts. Therefore Construct entity particularity is that it can refer to different BPMN elements.

Finally these three entities are in relationship to each other and in particular process and construct entities has a strong relationship. Indeed the Workflow Engine can instantiate Processes and Process is composed by Constructs.

4.1.1.1 Quality attributes of the entities: Workflow Engine, process and construct

The thesis focus is about performance of WfMS. Performance of a system is the characteristics respect to its quality attributes such as responsiveness, scalability, and resource usage during entity utilization. The ISO/IEC 9126 and later the ISO/IEC 25010:2011 [51] address the need to define an international standard regarding system and software quality. The ISO/IEC 25010:2011 identifies that quality model can be categorized into eight categories:

- Functional suitability;
- Performance efficiency;
- Compatibility;
- Usability;
- Reliability;
- Security;
- Maintainability;
- Portability.

In particular, in this thesis the performance efficiency is composed by three subcategories:

- time behaviour: regards throughput rates, response and processing time when the system is responding to requests.

- resource utilization: regards how much a resource is used and which resource is used when the system is responding to requests.
- capacity: ability of the system to fulfil a request and amount of requests that the system is able to handle.

The following set of quality attributes related to performance are considered for each entity:

- The selected Workflow Engine entity quality attributes are:
  - time behaviour;
  - resource utilization;
  - capacity.
- The selected Process entity quality attributes are:
  - time behaviour.
- The selected Construct entity quality attributes are:
  - time behaviour;
  - capacity.

Only these quality attributes have been selected for each entity due to the relationships between the entities that make quality attribute redundant; especially capacity attribute of the process entity is contained in the capacity attribute of the Workflow Engine entity. However, it could be interesting to add also to Process and Construct entities the resource utilization attribute, but usually this type of attributes is applied to the system under study and to other systems which interact with the system under study [70]. In particular, the reason why it is not selected is that evaluating resource utilization attribute for entities that are not system component might not be accurate.

4.1.1.2 Determine the goals and the internal entities attributes

The goals are determined from the viewpoint of the user and they are:

- Workflow Engine entity: analyses the Workflow Engine with the purpose of evaluate its performance from the user viewpoint to understand if they meet the user performance requirements. So Workflow Engine is analysed to understand its time behaviour and resource utilization for the user requests and its capacity to satisfy increasing number of user requests.
- Process entity: analyses the time behaviour required to execute a process and to understand if the WfMS is able to meet time expectations from the user viewpoint.
- Construct entity: analyses the ability to execute or not a construct and the ability to understand if the WfMS can handle a construct with proper performance from the user viewpoint.

So, this study's goal is to evaluate the entities by judging their value according to the performance efficiency attributes, which are time behaviour, resource utilization and capacity, from the point of view of the user, who is the client of the WfMS service and the one who instantiates processes.

Furthermore, internal attributes of Workflow Management System are:

1. start time of the test;
2. number of requests submitted by the test;
3. start time of each request of the test;
4. connection time required for each request of the test;
5. response time required for each request of the test, it is also identified as sample time;
6. utilized WfE RAM memory;
7. utilized WfE processor time;
8. amount of utilized Bytes in network communication in and out;
9. start time of a process instance;
10. end time of a process instance;
11. start time of a construct;
12. end time of a construct;

The internal attributes till the 8 attribute are used for the Workflow Engine entity. Attributes from the 9 to 10 are used for the Process entity and for the Workflow Engine entity. The final two internal attributes are used for the Construct entity and the Process entity.

4.1.1.3 Abstractions to measure the attributes

It is possible to identify three categories of internal attributes according to the abstraction required for their measurement. Attributes from 1 to 5 refer to a test, therefore data about these internal attributes can be measured by tools of load testing, such as the driver (refer to Section 3.1) or JMeter [4] , which might maintain record of the test launched. Attributes from 6 to 8 refer to how the system exploits resources and it is required a resource monitor to retrieve these information. The final internal attributes from 9 to 12 are all related to the execution of process models, therefore are attributes that depend on the Workflow Management System and its configuration.

4.1.1.4 Determine the questions from the goals

In this study, to determine the set of possible questions related to WfMS, it is used the advanced brainstorming technique to brainstorm by yourself [57] (thus the participant of the brainstorm by yourself is only the author of the thesis). As conventional brainstorming, in this phase all questions are considered and any raised questions are considered. Indeed during brainstorming the objective regards quantity of ideas and not quality of ideas. To avoid wasted time and come

to a conclusion of this step, it is applied a time frame mechanism. At first, the question from which the brainstorm starts must be clearly stated. In particular, questions for each entity are stated for their quality attributes.

Questions related to Workflow Engine entity:

- time behaviour property: Which time behaviour characteristics of the Workflow Engine are relevant for the user?
  - how much time is required to create the process instance when a process execution request is queried to the WfMS?
  - at which rate are processes completed?
  - how many processes can be completed over a specific time interval defined as the complete test duration?
- resource utilization property:
  - how much external storage is used to store process data and execution information?
  - which are the minimum resources requested to complete a process? Resources in term of: space resources (e.g. disk space, RAM memory usage), processing resources (e.g. CPU usage) and communication resources (e.g. network usage regarding packets received and sent)
- capacity property:
  - which is the maximum number of processes that can be executed successfully?
  - how many process instances has the engine handled without any error?

Questions related to Process entity:

- time behaviour property:
  - how much time is the time range required for processes to be completed?
  - which is the most time consuming process of the workload mix?

Questions related to Construct entity:

- time behaviour property:
  - how much time is required to start a new activity when the previous ones have been completed? So which is the latency of connecting flow?
  - how much time is required to communicate to a service task? How much time is required to complete construct execution?
  - which is the most time consuming activity of a process in term of time needed by the WfMS to handle the BPMN elements?
  - when the user has work to do, how much time is required for a user to get assigned an activity by the Worklist Handler?
- capacity property:
  - which is the maximum number of activities that can wait in the Worklist Handler?

- how many constructs can a process contain at maximum without performance degradation?

4.1.1.5 Identify relationship between external properties and internal attributes

Quality attributes or external properties depend on more than one internal attribute as it is shown in the Table 4.1. In particular it is possible to state that time behaviour attribute depends on all time related internal attributes. Also capacity quality attributes depend on these time related internal attributes, because the increase in requests produces an increase in time behaviour that is beneficial to be studied [63]. Moreover, capacity property is also related to the number of requests submitted by the test driver. Finally resource utilization has relationships with internal attributes that regards how the resources are used.

Table 4.1. Entity external properties and internal attributes relationship

Time Behaviour depends on:	**Capacity depends on:**	**Resource utilization depends on:**
start time of a process instance	start time of a process instance	utilized WfE RAM memory
end time of a process instance	end time of a process instance	utilized WfE processor time
start time of a construct	start time of a construct	amount of received bytes
end time of a construct	end time of a construct	amount of transmitted bytes
start time of the test	start time of the test	
start time of each test request	start time of each test request	
connection time of each test request	connection time of each test request	
response time of each test request	response time of each test request	
	number of submitted test requests	

4.1.2 Metric creation activity

The metric creation is composed of mainly four components: name of the metric, metric goal, metric description and pseudo-formulas for computing the metric. Since metrics are derived from the previous identified questions that were divided according to the quality attributes of the entity under study, the same approach is used also for metrics creation [37]. In particular the metrics, which can be defined for the Workflow Engine entity, can be divided according to:

- time behaviour in:
  - response time

Name	Response time
Goal	Determine the time required by the Workflow Engine to respond to a request.
Description	Response time is the time between the time of the request and the time of the response [63]. It is the time that the user experience when he/she performs a request to the WfE and response time considers when the user sends the request and when the user receives a response. So response time considers the time from the sending of the process execution request to the process completion communication reception, which means that it considers process execution and the time to transmit the request and communicate a response. In case that the request is asynchronous, the response time time considers the time from the process execution request sending to the process completion.
Pseudo-formula	Response time $= \begin{cases} \text{end request time - start request time,} & \text{synchronous request} \\ \text{end process instance time - start request time,} & \text{asynchronous request} \end{cases}$

– throughput

Name	Throughput
Goal	Determine the process completion rate
Description	The throughput is the rate at which processes are completed [63]. It is the number of request completed over an interval of time. Throughput is also related with the number of requests that are submitted. Indeed the Workflow Engine cannot complete more requests that those which it has received. Throughput considers all requests that are completed in the time interval; therefore it is computed on requests that are started and completed in the time interval and requests that are started before the time interval and that are completed in the time interval.
Pseudo-formula	Throughput $= \frac{\text{number of completed requests}}{\text{time interval}}$

– process execution latency

Name	Latency
Goal	Determine overhead introduced by the Workflow Engine
Description	The latency is the time required to instantiate a process after the request to execute the process models. So it corresponds to the time between the request and the actual instantiation of the process instance, where the actual instantiation of the process instance indicates the creation of the process instance. It is influenced by communication latency and instantiation requirement.
Pseudo-formula	Latency = start request time - start process instance creation

- resource utilization in:
  - database bandwidth usage

Name	DB bandwidth usage
Goal	Determine how much the Workflow Engine exploits database
Description	DB bandwidth usage is the amount of bytes used by the Workflow Engine to transmit or to receive data from the DB.
Pseudo-formula	DB bandwidth usage = sum of the amount of bytes utilized in network communication in and out to the DB

  - RAM memory usage

Name	RAM memory usage
Goal	Determine how much the Workflow Engine relies on RAM memory
Description	RAM memory usage is the amount of bytes used by the Workflow Engine to store temporary data on RAM memory and to be operational.
Pseudo-formula	RAM memory usage = sum of the RAM memory utilized by the WfE

  - CPU usage

Name	CPU usage
Goal	Determine processing requirement of the Workflow Engine
Description	CPU usage is the percentage of processing time used by the Workflow Engine to operate.
Pseudo-formula	CPU usage = $\frac{\text{CPU utilization time}}{\text{time interval}}\%$

  - network usage

Name	Network usage
Goal	Determine Workflow Engine communication requirements
Description	Network usage is the amount of communication received and transmitted by the Workflow Engine to be functional, so it is the amount of communication among WfMS components, such as WfMS database, Web Applications, and WfE.
Pseudo-formula	Network usage = sum of the amount of bytes utilized in network communication in and out to the WfE

- capacity in:
  - capability

Name	Capability
Goal	Determine the ability of the Workflow Engine to manage incremental number of process instantiation requests
Description	Capability is the number of processes that the Workflow Engine is able to handle at maximum. To determine the capability a time interval should be defined and all processes instances that are in execution in that time interval should be considered.
Pseudo-formula	Capability = max(# process instance in time interval)

  - number of completed processes

Name	Number of completed processes
Goal	Determine the ability of the Workflow Engine to manage incremental number of process instantiation requests
Description	Number of completed processes is the number of processes instances that the Workflow Engine has handled. To determine the number of completed processes a time interval defined as test duration should be used and all processes instances that are in execution in that time interval should be considered.
Pseudo-formula	NumCompletedProcesses = # completed process instance in time interval

The metrics that can be defined for the Process entity for the time behaviour quality attribute are:

- completion time

Name	Completion time
Goal	Determine the time required by the Workflow Engine to execute a process instance
Description	Completion time is the time between start of the process instance and the time of its completion. It is the running time of the process instance on the Workflow Engine.
Pseudo-formula	Completion time = end process instance time - start process instance time

Finally the metrics that can be defined for the Construct entity can be divided according to:

- time behaviour in (see Figure 4.1):

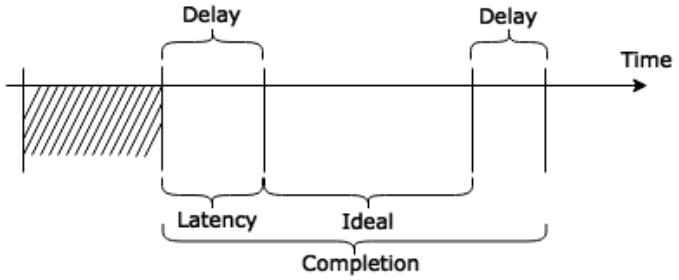


Figure 4.1. Time behaviour schema

  - completion time

Name	Completion time
Goal	Determine the time required by the Workflow Engine to execute a construct of a process instance.
Description	Completion time is the time between start of the construct and the time of its completion. It is the running time of the construct of process instance on the Workflow Engine.
Pseudo-formula	Completion time = end construct time - start construct time

  - delay

Name	Delay
Goal	Determine the delay introduced by the Workflow Engine to execute the construct.
Description	Delay is the time required by the Workflow Engine to execute the construct respect the ideal time required to execute it. So it is the time difference between actual execution time and expected or ideal execution time [37].
Pseudo-formula	Delay = actual construct completion time - ideal construct completion time

- latency

Name	Latency
Goal	Determine the time required by the Workflow Engine to start the execution of the construct.
Description	Latency is the time between end of a previous construct and the start of next one. It is determined by the time required by the WfE to follow the execution path, to determine the next construct to instantiate and to instantiate it.
Pseudo-formula	Latency = start next construct time - end previous construct time

- capacity in:
  - construct capability

Name	Construct capability
Goal	Determine the amount of construct that the Workflow Engine can process simultaneously
Description	Construct capability indicates how many constructs the WfE can execute and handle in waiting state at maximum. To determine the construct capability a time interval should be defined and all constructs that are in execution in that time interval should be considered.
Pseudo-formula	Construct capability = max (# construct in time interval)

  - number of completed constructs

Name	Number of completed constructs
Goal	Determine the ability of the Workflow Engine to manage incremental unit of work
Description	Number of completed constructs is the number of constructs that the Workflow Engine has handled. To determine the number of completed constructs a time interval should be defined and all constructs that are in execution in that time interval should be considered.
Formula	NumCompletedConstruct = # completed constructs in time interval

4.2 Theoretical validation

The theoretical validation is conducted applying the SMART framework [32]. The table 4.2 shows which metrics should be considered as SMART and which are the metrics implemented in this thesis. All the metrics are SMART, because they respect all the SMART characteristics. For further details on the SMART framework applied to each metric see Appendix A.

Due to time limitations not all the metrics are implemented, besides the set of implemented metrics allow achieving a first analysis of WfMS performance, as it is shown in Chapter 7. In particular, the construct metrics are not implemented, because they can be considered as an in-depth analysis of the process metric; indeed constructs execution is tightly correlated to process instances execution.

Table 4.2. Metrics Summary Table

Entity	Metric name	SMART	Implemented
Workflow Engine	Response time	yes	no
	Throughput	yes	no
	Latency	yes	no
	Capability	yes	no
	NumCompletedProcesses	yes	yes
	DB bandwidth usage	yes	no
	Network usage	yes	no
	RAM memory usage	yes	no
	CPU usage	yes	no
Process	Completion time	yes	yes
Construct	Completion time	yes	no
	Delay	yes	no
	Latency	yes	no
	Construct capability	yes	no
	NumCompletedConstruct	yes	no

4.3 Empirical validation

The empirical validation is presented in Chapter 7. Due to the fact that the Chapter 7 is an empirical validation made only using the implemented tools: data cleaner and reconciler, performance meter and data aggregator, a more complete empirical validation is proposed as a future work (see Section 8.2). The metrics considered are the metric implemented in the performance meter and data aggregator and, in particular, these metrics are called:

- ProcessCompletionTime,
- NumberCompletedProcess,
- NumberUncompletedProcess.

Where the NumberUncompletedProcess metric is a complementary metric of the NumberCompletedProcess. For more information on the implemented metrics see Section 6.4.1.

Chapter 5

Data cleaner and reconciler

The data cleaner and reconciler has the objective to standardize data from databases of different Workflow Management Systems. This standardization is required because different WfMSs store relevant data for analysis with different data structures and different notations. In practice the data cleaner and reconciler performs the Extraction, Transformation, Loading (ETL) process.

5.1 Background of ETL

Extraction, Transformation and Loading is a consolidation process that involves retrieving data from a source database, transforming it to meet business needs, and ultimately loading into a destination database, called data warehouse.

Although ETL processes are very important, ETL has little research. This is because of its difficulty and lack of formal model for representing ETL activities that map the incoming data from different source database to be in a suitable format for loading to the destination database [30, 50, 59].

In this thesis the conceptual model of ETL process proposed by Vassiliadis et al. [99] is explored, and the conceptual model is applied to the data cleaner and reconciler component of the framework.

Indeed, in this thesis the data cleaner and reconciler is the ETL tool and its steps are:

- data extraction, it extracts data from a source database that corresponds to the Workflow Engine database or a dump of it;
- data transformation, it transforms the data cleaning them, so that without other modifications the data can be used for query and analysis about the WfMS metrics;
- data loading, finally it loads the clean data in a destination database, which corresponds to data cleaner and reconciler database or CleanRawData database.

Summing up, the implemented data cleaner and reconciler brings heterogeneous and asynchronous source extracted data to a homogeneous environment. Moreover ETL is a complex process due to the fact that it has to reliably manage big amount of data.

5.1.1 ETL conceptual model

The ETL conceptual modeling goal is to map the attributes of the data, extracted from the data source, to the attributes of the data destination schema. In particular, the conceptual part of the definition of the ETL deals with the first stages of the destination database design. During those stages the destination database designer have to manage:

- the collection of requirements needed in the destination data;
- the analysis of the structure and contents of the data, present in the data sources, and their intentional mapping to the destination database.

In their paper [99] Vassiliadis et al. focus on the interrelationships of attributes and concepts, that are caught through provider relationships that map data attributes from the source to the destination data; and the needed transformations that the source data has to take before loading. More specifically, the ETL conceptual model phases are: Extraction, Transformation and Loading, and the advantages of ETL process are:

- adjusting and conforming data from multiple sources to be used together;
- structuring data;
- documenting measures of confidence in data;
- capturing the flow of transactional data;
- enabling subsequent analytical data processing.

5.1.1.1 Extraction

During this phase the relevant data are extracted from a source database. There are three types of extraction:

- static extraction;
- incremental extraction;
- full extraction.

The static extraction is done when the destination database has to be populated for the first time. The incremental extraction is used for a periodic update of the destination database and it captures only the changes happened in the data of the source database from the last extraction. The full extraction is used when the destination database has to be populated with all the source databases; the data cleaner and reconciler applies this type of extraction.

5.1.1.2 Transformation

During the transformation phase the data, extracted from the source database, are converted from the source database format to the destination database format, so that they can be used for query and analysis conformed to the business needs. During the transformation the data must be converted, a common standard has to be created to transform the source data in destination data that can be analyzed and evaluated. Thus the data are translated into the desired design and required form of the destination database. This phase is the most critical and important of the ETL process [99].

5.1.1.3 Loading

The last phase of the ETL process is the loading, which consists of the insertion of the cleaned data to the destination database.

5.2 Design of the data cleaner and reconciler database

The data cleaner and reconciler has to maintain the clean data; therefore a common database, which will contain data from all the source databases, must be realized. At first it is required to design such common database or data warehouse. The design is based on [99]:

- collection of basic requirements,
- context knowledge.

5.2.1 Requirements

The requirements are mainly derived from the metrics that are going to be considered. The considered metrics are fundamental; they highlight the minimum requested entities and the information required in each table, because entities define the tables that must be designed, indeed each entity requires a specific table.

In particular the metric considered can be derived from data that can be stored by the WfMS into its database and in general these data are related to the process and construct executed by WfMS. Therefore accordingly to stored data in WfMS database, the possible entities that can be identified in the Workflow Management System context are:

- Process;
- Construct.

It is evident that these levels compose a hierarchy. As a matter of fact the WfMS executes process instances, where process instances are composed of constructs that are executed during the execution flow of the process instance. Therefore two entities can be determined: process and construct and the process entity must have a relationship with the construct entity, because a process is a composition of constructs.

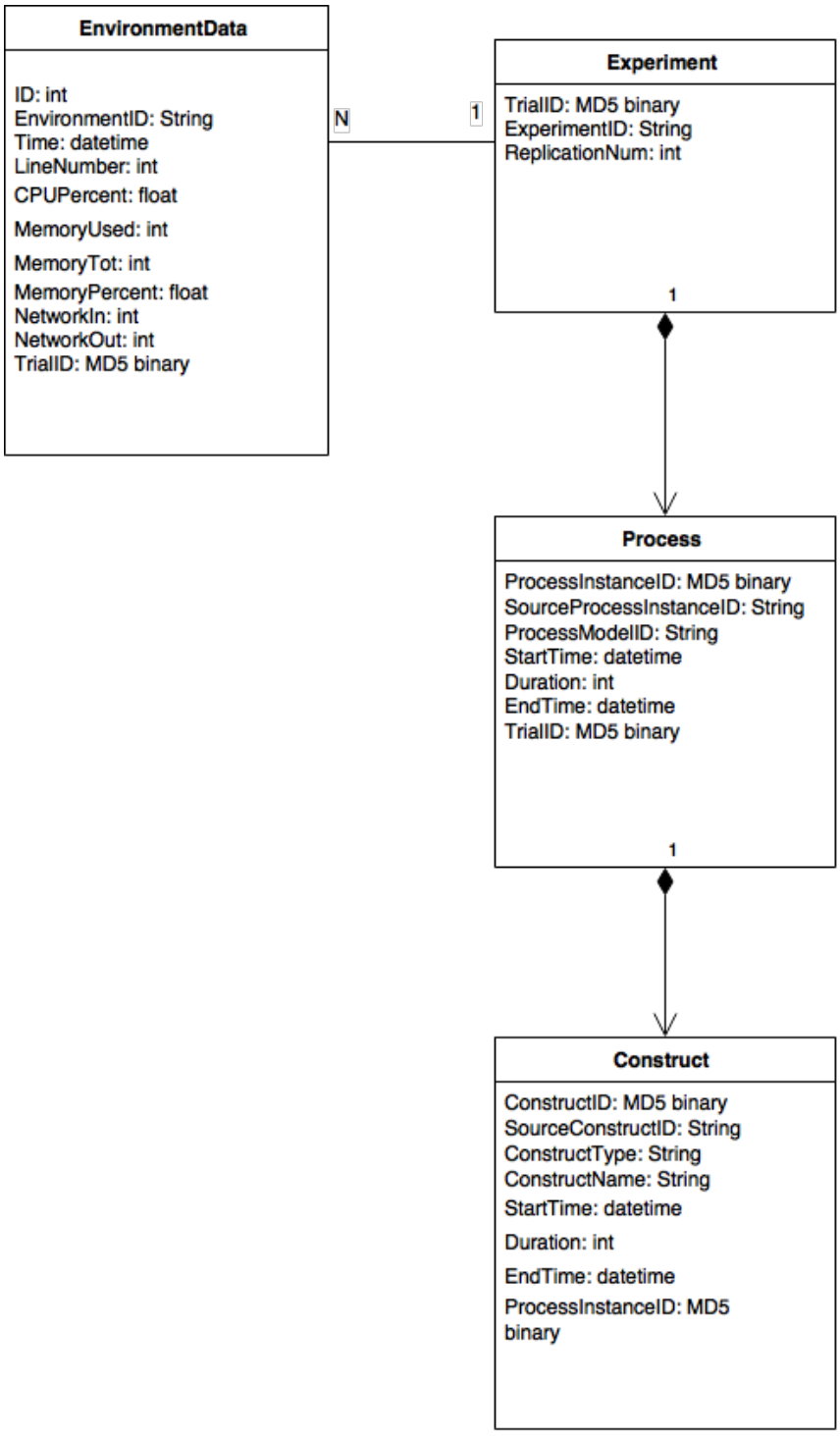


Figure 5.1. Target database schema

Moreover each process executed is related to a test, which is refereed as experiment to generalize the term. The experiment entity is related to the type of testing that is running to obtain the data. In particular in WfMS context, the experiment consists in requesting the execution of process models; so the experiment entity must have a relationship with the process entity, because the experiment is composed of process instances that are executed on the WfMS.

In addition the data cleaner and reconciler loads statistics from environments. Thus experiment entity also requires additional information about the environment and an environment entity might be used for this purpose.

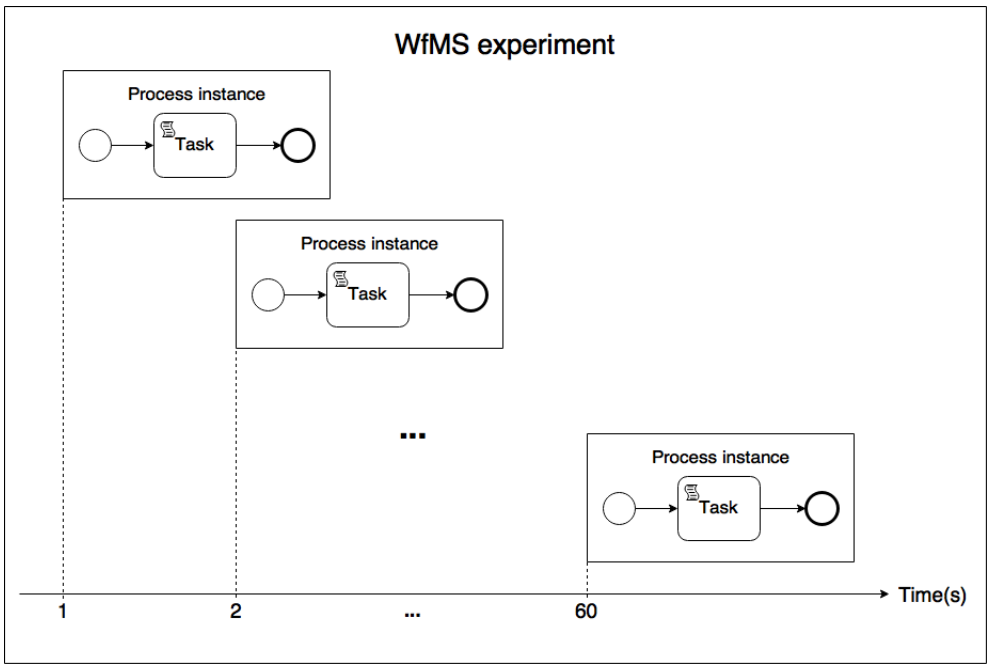


Figure 5.2. Example entities of data cleaner and reconciler requirements

The example shown in the Figure 5.2 is provided to better explain the role of the mentioned entities. It is possible to understand that there is an experiment entity that is characterized by performing one request every second for one minute and in particular the request regards the execution of a process instance. So the process entity is related to the experiment entity. Moreover the process instance execution requires to perform three constructs, which are a start event, a task and an end event; therefore it is shown also the construct entity and its relation with the process entity. Instead the environment entity is not shown in the Figure 5.2, but the WfMS to execute process instances must be deployed on a machine and it has to interact with its components, such as the DBMS (see section 2.2.2), which are the environment.

Finally no information should be removed. Hence for more detailed analysis it is preferable to store also the original data from which the cleaned data are derived from, because the data cleaner and reconciler database does not contain all files and data produced. For example it does not contain original data and it might not contain some logs. Therefore for each experiment execution, called trial, it should be created an external folder that contains all files that might be useful to further investigation on the trial. This duty is of the collector component, however

this principle introduces requirements according to which it is necessary to backtrack original data from the cleaned data.

5.2.2 Entity schema details

As discussed in Section 5.2, entities are represented by the following tables in the database: experiment, process, construct and environment data. See Figure 5.1.

5.2.2.1 Experiment entity

The experiment entity contains information about experiment trials, which are executions of the experiment. It might be related to an entity modeled in an external database which specifies the experiment characteristics, in particular this relationship is allowed by ExperimentID. ExperimentID specifies which experiment has been performed as a trial. Since an experiment can be repeated multiple times, the ReplicationNum specifies the trial number, therefore it is a incremental number increased by one for each trial of the experiment.

This means that for each ExperimentID should not be possible to have the same ReplicationNum, because every time that an experiment is performed the ReplicationNum should increase by one for that specific experiment. Finally the Experiment table is identified by the TrialID field.

5.2.2.2 Process entity

The process entity contains information about the process instances of a specific experiment trial, which means that each record of the process entity refers to one trial. Indeed, as identifier process entity has ProcessInstanceID, which distinguishes the process instances executed and it has a reference through TrialID to the Experiment entity to obtain information about the experiment itself. The ProcessModelID field identifies which is the process model that was instantiated and it might refer to an external database that holds information about process models.

Every time a process model is instantiated there is a related start time, which indicates when the process instance has started its execution. Of course if the process instance terminates it also has an end time, which determines when the process instance has been completed. Both these data time are stored respectively into StartTime and EndTime fields. Moreover, if process instance duration is not already computed, it is possible to determine the duration of a process instance, which is defined as difference between the time at which the process instance began and the time at which the process instance is completed. This information is stored into the Duration field, because it allows fastening future analyses that exploit this value.

Finally the process entity contains SourceProcessInstanceID field, which is used to allow users to backtrack process instances in source database. Therefore SourceProcessInstanceID field has the same value of the identifier used in the source database for the process instance, with the exception that it might be required a data type conversion.

5.2.2.3 Construct entity

The construct entity contains information regarding the constructs used to build the process and in particular it includes information about flow objects [55, pg. 27]. Since constructs belong to a specific process, each instance of the construct entity holds a reference to process entity by using the ProcessInstanceID field. In addition the ConstructID field univocally identifies an instance of the construct entity.

Since the construct entity contains different types of objects, such as gateways and tasks, ConstructType field is used to understand which type of construct it is. Similarly to a process instance a construct has a StartTime, Duration and EndTime fields. In addition there is the ConstructName field that is used to identify the same construct of different process instances.

Finally SourceConstructID field is used to allow users to backtrack constructs in the data source. Therefore SourceConstructID field has the same value of the identifier used in the data source, with exception that it might be required a data type conversion.

5.2.2.4 EnvironmentData entity

The environments are all systems that allow the execution of trials. For example environments in the Workflow Management System are the database used to store information of the WfMS and the Workflow Engine. An external database might contain data regarding environments on which experiment trials are running. Instead, EnvironmentData entity holds data regarding the run time statistics of the environments.

The EnvironmentData entity uses ID field as identifier of the instances and this entity also holds the EnvironmentID field, which specifies the environment. Moreover each instance of the EnvironmentData entity also refers to a specific experiment trial, which is specified by TrialID field. Therefore each EnvironmentData entity's instance has a relationship with both a specific environment and a specific experiment trial.

In particular the environment statistics could be derived by means of commands to obtain statistics, for instance, when Docker is used, Docker stats API data regards the CPU usage,RAM memory usage and network usage. CPU usage is a percentage of CPU used over time intervals and it is computed considering both user and system time. The user time is the time during which the environment process is directly controlling the CPU, whereas system time is the time during which the system is operating to complete requests performed by the environment process. RAM memory usage is composed by different values: RAM memory occupied by the environment, maximum RAM memory available and a percentage that represents how much of the maximum RAM memory is occupied. The network usage is composed by two values, one indicates the bytes received and the other the bytes sent.

Finally CPUPercent field is used to store CPU usage; MemoryUsed, MemoryTot and MemoryPercent fields are used to store RAM memory usage respectively to: RAM memory used, total RAM memory available and RAM memory percentage used; NetworkIn and NetworkOut fields are used to keep track of bytes received over network and bytes sent.

5.3 Implementation of the data cleaner and reconciler database

The key fields of the different tables, identified by the entities (refer to Section 5.2), are determined by the MD5 hash function of one or more of the following possibilities: other fields of the entity, fields of the source database and a seed determined by the data cleaner and reconciler, which is a value. Hash function is used because it can map source key fields to one single key field, indeed hash functions are functions that operate and map an input string of different length to an output of bit string with a fixed length, called hash. In general hash function is a one-way function that maps a domain value to a range value. Therefore it is an optimal solution to determine the identification key for two reasons: fixed length and the large range of output values. In particular the fixed length is useful to limit the storage size required to store the identification key itself and to increase the performance, because the key comparison will be quicker. In this study the MD5 is chosen, being considered an enough good collision resistant hash function. Indeed it is able to obtain 2^{128} output combinations, since its hash result is 128 bit long.

In the last years the hash functions, including the MD5, were objects of criticism about their security [90, 91]. Some of those criticisms are valid, but in this thesis the MD5 hash function is used because of:

- fast computation;
- resistance to the collisions;
- low space requirement.

Moreover the framework is not used for security reasons, thus possible MD5 security breaches are not considered as a real threats in this application due to their limited impact. In particular the key fields determined by the MD5 are: TrialID, ProcessInstanceID and ConstructID. Instead the ID field of the EnvironmentData entity is obtained by the auto increment function implemented by databases.

The current design of the cleaner database (see Section 5.2) applies the three normalization rules [24, 25]. The three normalization rules are usually applied to eliminate redundant data and to ensure correct data relationships. In particular, the first normal form states that each table contains atomic value attributes, which means that each attribute of a tuple, i.e. each attribute of a row of a table, does not have multiple values belonging to the attribute domain values. The second normal form requires that the first normal form is satisfied and that the non primary key attributes are fully dependent on the primary key attribute. The third normal form requires that the second normal form is satisfied and that the non primary key attributes does not depend on non primary key attributes. However during data analysis the access to process records and construct records is mainly based on the trial interested. This requirement does not provide any difficulties for process records, which have their foreign key at trial records. Whereas construct records have to be joined to their process record to determine the trial at which they belong.

This can be translated by the following SQL queries:

```
select `CleanRawData`.`Construct`.`ConstructID`, `CleanRawData`.`
    Construct`.`SourceConstructID`, `CleanRawData`.`Construct`.`
    ConstructType`, `CleanRawData`.`Construct`.`ConstructName`, `
```

```
    CleanRawData`.`Construct`.`StartTime`, `CleanRawData`.`Construct
    `.`EndTime`, `CleanRawData`.`Construct`.`Duration`, `
    CleanRawData`.`Construct`.`ProcessInstanceID`

from `CleanRawData`.`Construct`

where `CleanRawData`.`Construct`.`ProcessInstanceID` in

    (select `CleanRawData`.`Process`.`ProcessInstanceID`
    from `CleanRawData`.`Process`
    where `CleanRawData`.`Process`.`TrialID` = 'TrialIDvalue');
```

or

```
select `CleanRawData`.`Construct`.`ConstructID`, `CleanRawData`.`
    Construct`.`SourceConstructID`, `CleanRawData`.`Construct`.`
    ConstructType`, `CleanRawData`.`Construct`.`ConstructName`, `
    CleanRawData`.`Construct`.`StartTime`, `CleanRawData`.`Construct
    `.`EndTime`, `CleanRawData`.`Construct`.`Duration`, `
    CleanRawData`.`Construct`.`ProcessInstanceID`, `CleanRawData`.`
    Process`.`TrialID`

from `CleanRawData`.`Construct` join `CleanRawData`.`Process`
on `CleanRawData`.`Construct`.`ProcessInstanceID` = `CleanRawData`.`
    Process`.`ProcessInstanceID`

where `CleanRawData`.`Process`.`TrialID` = 'TrialIDvalue'
```

By analysing the queries, it is possible to see that even the optimal query, which uses "in" operator, and considering indexing on primary and foreign key, which is the best case scenario for these select queries, requires a $\mathcal{O}(c_n + c_m \times n)$ where n is the number of process instances belonging to the trial and c is the access time through indexes of process instances (c_n) and of constructs (c_m). In particular the access time through index is logarithmic, because indexes are stored in B-tree [6].

To avoid expensive time access on construct record related to trial, the third normalization rule [25] is ignored. Therefore, trial field is added to construct table and this causes an increase of disk space, but it is compensated by a faster access of records related to trials. Indeed access time is $\mathcal{O}(c_m)$.

5.4 Design of the data cleaner and reconciler

The data cleaner and reconciler has a hierarchy structure since it can be used to clean data from different WfMSs and also environments data. See Figures 5.3 and 5.4. Moreover the data cleaner and reconciler hierarchy has additional support for database management by two external classes to the hierarchy. These two classes are:

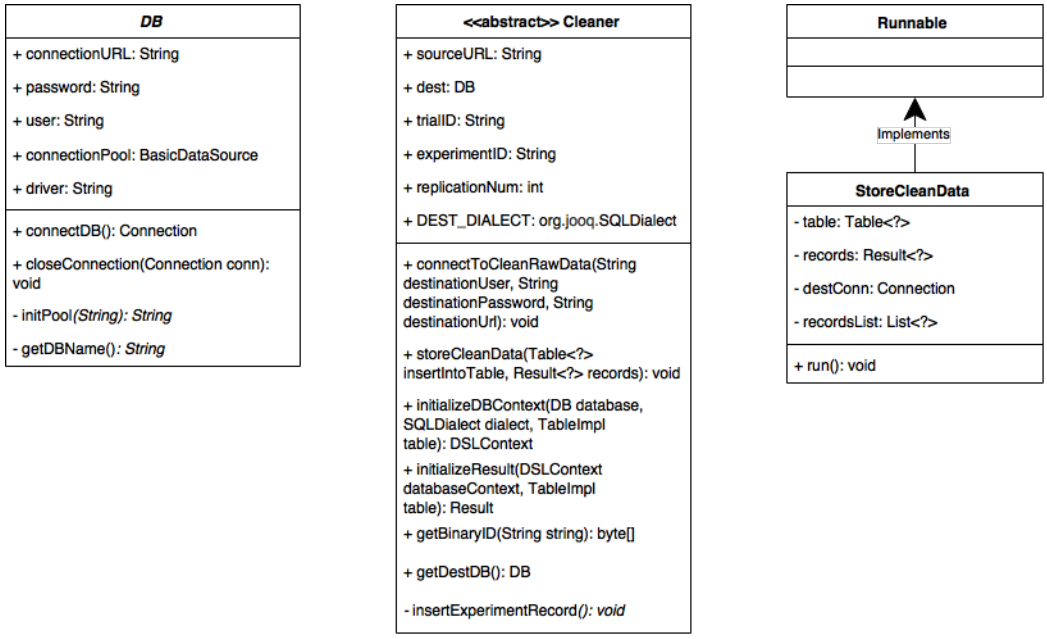


Figure 5.3. Data cleaner and reconciler Java classes - part 1

- DB;
- StoreCleanData.

DB class completely manages database parameters initialization and a connection pooling to database. While the StoreCleanData class is designed specifically to store clean data.

The hierarchy of the data cleaner and reconciler is based on an abstract class called Cleaner class. Cleaner class is composed by a set of fields and methods and Cleaner fields are mainly used to identify the location of the source data and of the destination database. Source data location could be a file or a database according to the required cleaning type. In particular the support class is used for the destination database and it manages database connection and requires the following parameters:

- Uniform Resource Locator (URL), which identifies location of the CleanRawData database;
- username to access the destination database and it must have write privileges;
- password to connect to the destination database and it is used for authentication.

Cleaner class has additional fields which are used to identify the experiment trial. In point of fact each time that the data cleaner and reconciler is used a specific trial must be specified, because clean data must refers to trials. According to the CleanRawData database schema 5.1, the Cleaner class requires to know the identifier of the experiment and the replication number, which is the value of how many times the experiment has been replicated respect to the specific trial; so that the trial identifier can be determine by applying the MD5 hashing function of the experiment identifier and replication number. Hence trialID is automatically computed given experimentID and replicationNum fields.

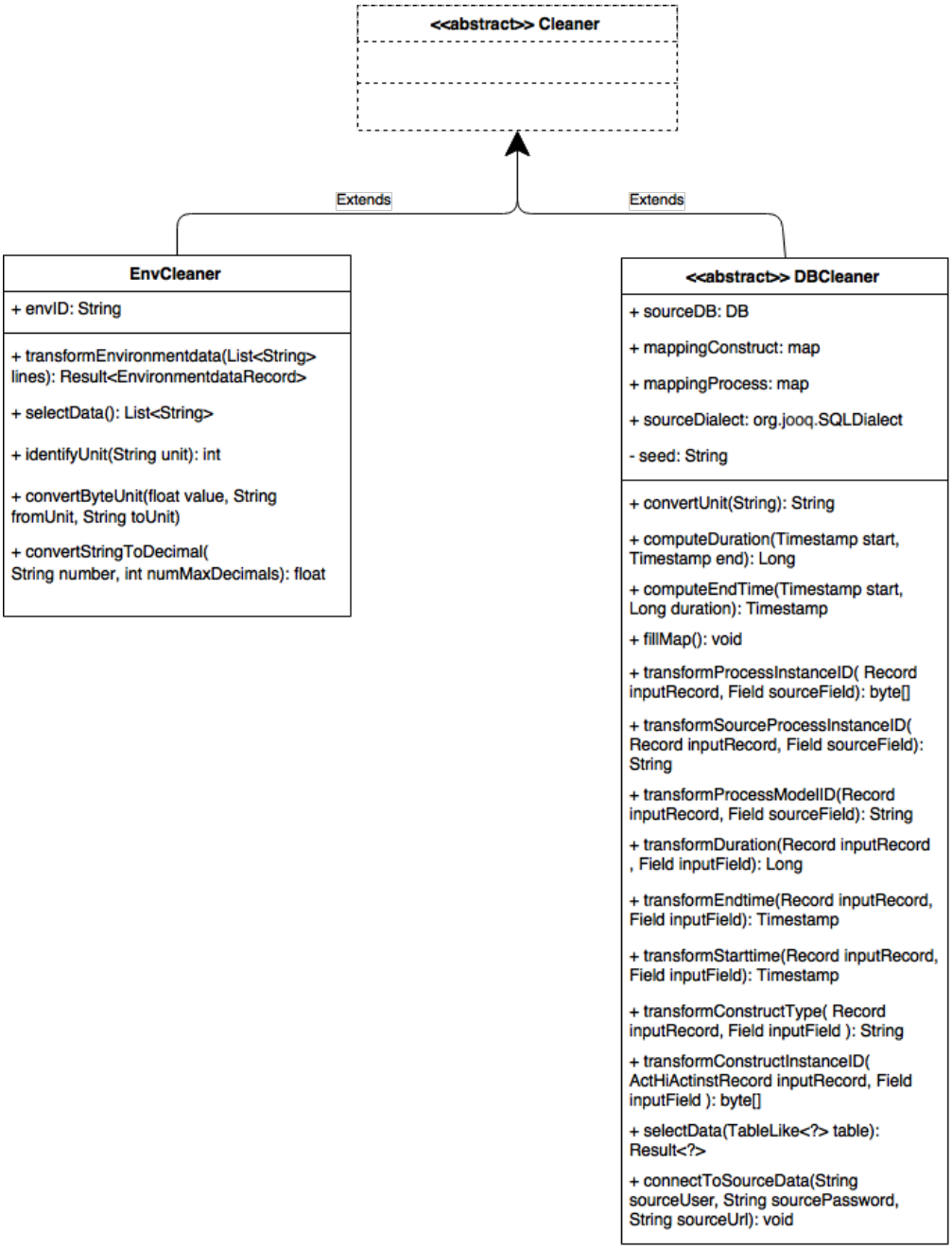


Figure 5.4. Data cleaner and reconciler Java classes - part 2

The Cleaner class is also composed of methods, which might be divided between abstract and concrete methods. The Cleaner class concrete functionality regards loading clean data in the clean database and provides some additional support functions. So Cleaner class methods can be also identified according to their use, which can be summarized to:

- access data resource by connecting the Cleaner class to source database or file and to destination database. For this purpose there are two methods:
  - connectDestDB(), it is a concrete method and it is used to connect objects with CleanRawData database.
  - connectSource(), it is used to open a connection with data source and it is an abstract method. Indeed data source type might be a file or a database depending on data involved refers to an environment or to process execution.
- store data into CleanRawData. Given cleaned data, the method uses the support class for load data. This method in the Cleaner class is concrete, but it requires that the input data respects clean database schema, because given a well formatted input data, it is possible to store data directly without requiring to reimplement store method.
- determine ID values according to MD5 hashing function and inputs (for more details on ID values see Section 5.3). For this purpose there are two methods; these two methods are almost identical except for their parameters, which can be a single valued variable or an array.

The Cleaner class is the root of inheritance and it has two direct subclasses:

- EnvCleaner, it has to clean data relative to environments;
- DBCleaner, it has to clean data relative to the execution of experiment trials, so it treats data that are stored from WfMSs in databases.

EnvCleaner class is a concrete subclass, because all the environment statistics data are collected by applying docker stats API. Therefore it is possible to assume that an output data has similar format for all type of environments and it does not depend on the environment itself. So EnvCleaner class reads statistics file, translates data in destination database schema format and loads translated data into the CleanRawData database. EnvCleaner has another field in addition to Cleaner class fields, which is the identifier of the environment and it is provided as input in the constructor of the class. Moreover the field sourceURL in EnvCleaner class will be valued with a path to a file containing environment statistics.

Instead DBCleaner is an abstract subclass, because it depends on the source database and its data structure. Furthermore DBCleaner assumes that data source is a Database Management System (DBMS). Indeed most of Workflow Management Systems uses DBMS to store information about execution. Hence, in addition to Cleaner class fields, DBCleaner fields are the username and password to access the source database with read privileges, the dialect to interact with the source database, transformation map field that allows to map source fields into destination fields of the CleanRawData database and a seed that can be added to create unique ID values.

DBCleaner has also additional methods that will be useful, since DB cleaning is more complex than environment cleaning. These methods functionalities regard:

- unit of measurement conversion: these methods convert values from a unit of measurement into values in the standard DataRawClean unit of measurement by exploiting the metadata that are contained in the data extracted from the source database;
- transformation, these methods allow to map input data into a well formated output data, that can simplify cleaning;
- data selection and database connection, they consist of methods that interact with source database. In particular data selection method extracts data from specified tables in the source databases.

The DBCleaner does not provide a full cleaning range of source database, because it must be as less specific as possible, but it provides a structured hierarchy to extend WfMS database cleaning jointly with implemented functions. So when a new WfMS database must be added, the required functions to implement are limited to the cleaning phase. In particular when it is required to clean data from a new WfMS, the DBCleaner class should be extended and cleaning methods require to be implemented according to the new data source. Instead if data are from a new version of an already implemented WfMS, it might be sufficient to extend the already implemented WfMS and override only the cleaning methods that treat data modified in the new version.

During the cleaning phase, the least amount of information should be removed. Only redundant and superfluous data might be removed, for example some application stores data that help the execution of the application itself and these data might not be interesting for other purpose and too much domain specific.

5.5 Implementation of the data cleaner and reconciler

The data cleaner and reconciler is developed in Java. Since the data cleaner and reconciler has to interact with databases, Object-relational mapping technique could be applied. Object relational mapper (ORM) is a tool which enables fetching objects from relational databases, thus the ORM is a technique able to translate objects to relational data. Indeed the most commonly used programming languages are object oriented, whereas some of the generally used databases are relational database.

Moreover according to Juneau [56] a ORM might allow to use the following features:

- definition of abstract entities, the development process makes sure that the used terminologies are the same and it eliminates languages that are non conformed;
- automatic synchronization, ORM uses database agnostics, called adapters, to synchronize specification with database schema;
- relationship modelization between defined entities;
- easiness of queries writing, using ORM is easy to write query because it does not depend on the language of the storage database;
- possibility to develop behaviour triggers that simulate business process;

- possibility to add validation objects attributes to have better data integrity and consistency.

So the main advantages of the ORM are:

- time saving with the automatic synchronization;
- developer performance thanks to the development process;
- the automatically generation of SQL queries, making data access more abstract and portable;
- integrity and consistency with the possible customizer configuration options to specify an object or a collection fetching;
- easiness of the language.

However the development of an effective ORM mechanism is difficult and not all these possible features are available [72] and for the following disadvantages it has been decided to use Java Object Oriented Query Language (jOOQ). Specifically one of the limit of the ORM is that the user is not able to go beyond simple SQL operation such as joins, nested, selects and aggregation. In addition the disadvantage of the ORM are classified in six categories by Ireland, C., Bowers, D., Newton, M., & Waugh, K. [52] and they are called ORM Mismatch Problems. The mismatch problems are:

- Granularity, relational databases do not have the same granularity of object language thus the representation of an object language class could not find direct correspondence with the relational database and the synchronization could be difficult.
- Encapsulation, ORM could have consistency problem due to the structural differences and functional units organization in object language programmes and relational database.
- Association, there are some limitations in the ORM when the state of the object is represented in a relational database because its state cannot be fully preserved.
- Identity, the problem of having two objects with different identity and same state due to difference concept of identity between object language and relational database.
- Navigation, there could be synchronized problems due to differences in terms of environmentals, manipulations and transactions between object language programmes and relational databases.
- Ownership, if the schema database and the class model are made by two different users there could be correlation problems.

Indeed a powerful option to the Object Relational access layer is the Object-Oriented Query Language (OQL). According to Zhongling Li [65], Object-Oriented Query Language (OQL) is designed to provide an object-oriented query interface for traditional relational database systems. The goal is to bridge the gap between object-oriented programming language (specifically Java) and set-oriented Standard Query Language (SQL), and make the persistence layer fit better in an Object-Oriented (OO) system design.

In particular jOOQ is a query Object-Oriented language originated from SQL and it maps SQL queries to objects. jOOQ assumes that the user wants a low level control over the queries execution, so it provides an easy interface for queries execution, rather than mapping objects from relational databases like ORM. The main advantages of the jOOQ are:

- that all operations are characterized at class level such as Create, Read, Update, Delete (CRUD) operations;
- the simple language due to similar simple syntax and grammar as Java;
- the easy integration with other frameworks;
- the portability, Java runs almost in every operating system.

Especially StoreCleanData class exploits the jOOQ framework and, instead of a single insert of single data, it applies a batch insert of the data to reduce communication overhead and thus improve loading performances. Moreover it implements Runnable interface to allow concurrent data load.

In addition to jOOQ, the DB class uses a connection pool mechanism, which is applied to improve the extraction and the loading performance of the data cleaner and reconciler. Indeed these two phases require to connect to databases to read or to store data. Specifically, the connection pool creates a pool of readily available connections anticipating the need to create a connection with the database, which is an expensive activity in term of resource and time. Thus, when a connection is required, the connection pool provides it immediately and later, when connection is not used any longer, it can be returned to the connection pool. Hence the connection is not closed and destroyed, but it is again available for a new use.

Furthermore sometimes duration of processes and of constructs are already computed and stored into data source. When this happens, it is a good practice to use this value instead of computing the duration by the difference of EndTime and StartTime fields. As a matter of fact a more accurate and precise computation might have been applied to compute this field in the data source. Moreover it must be noticed that EndTime and Duration fields might be null, if the process instance did not complete its execution. Maintaining these process instances with null EndTime and Duration fields is a good practice to avoid to lose possible valuable information.

Moreover the data cleaner and reconciler does not manage continuous data streaming, which means that it extracts all the data from source databases and all extracted data are maintained in RAM memory for fastening their access. Finally extraction from source database cannot occur during trials, which means that the source database is completely populated before an extraction occurs, because if the extraction is incremental, it might interfere with the experiment trial conducted on the Workflow Management System and so the data cleaner and reconciler could interact with WfMS database during its execution and remove resources to the WfMS execution. So the data cleaner and reconciler should be used when all process executions are finished and the state of the WfMS database is stable and at rest.

5.6 Evaluation of the data cleaner and reconciler

The objective of the data cleaner and the reconciler is to allow standardizing data from databases of different Workflow Management Systems. To evaluate if the data cleaner and reconciler is able to reach its objective , the most important characteristics to evaluate are:

- Amount of managed data;

- Correctness;
- Reverse Mapping;
- Scalability to the adding of new WfMSs.

In point of fact the data cleaner and reconciler should be able to manage hundreds megabytes of data[1] in a short period of time. The most important information is the time required for the operations and the ones evaluated are the extract, transformation and load time considered both for the processes and for the constructs data. Secondly, the data cleaner and reconciler should be able to clean the data without altering them, thus the correctness is an important characteristic that this tool should have. Thirdly, the user could need to go back to see the original data, so the data cleaner and reconciler should be able to enable the reverse mapping. Fourthly the data cleaner and reconciler should be able to work with different WfMSs, so its scalability to the adding of new WfMSs is evaluated.

The evaluation of the data cleaner and reconciler is performed on MacBook Pro machine with the 10.9.5 OSX version, 64 bit architecture, 8GB of DDR3 RAM and 2.4 GHz quad cores processor. The source and destination databases are two MySQL databases running on localhost through Docker containers and Docker is run on Virtualbox in a Ubuntu 14.04 LTS Virtual Machine; while the data cleaner and reconciler is run by Eclipse version 4.4.1 with default settings and with Java version 1.8.0_05. For more details about the used infrastructure read Section 7.2.

5.6.1 Source data generation for evaluation

The data cleaner and reconciler has been implemented for two WfMS: Camunda and Activiti. The implementation of these WfMSs was straightforward also due to the similarity of its database structure (see the migration guide at [2]).

For the evaluation the data cleaner and reconciler is applied only on Camunda, which has been chosen at random between the two WfMSs available due to the data similarity among them. The process model is designed by using the Camunda Process Definition tool and it consists in three constructs: a start event, a script task and an end event.

5.6.2 Amount of managed data

In this analysis, to evaluate the amount of managed data by the data cleaner and reconciler, some process instances and, as a consequence constructs, are submitted to the data cleaner and reconciler, in particular eight executions are made. The data cleaner and reconciler executions have a number of process instances and construct instances according to Table 5.1.

To evaluate the data cleaner and reconciler the considered time data are:

- process extraction time;
- process transformation time;

[1] The data of the execution number 8 occupy 616 MB of source database disk space

Table 5.1. Cleaner and reconciler executions

Execution number	Number of processes	Number of constructs
1	10,000	30,000
2	20,000	60,000
3	30,000	90,000
4	50,000	150,000
5	75,000	225,000
6	100,000	300,000
7	150,000	450,000
8	200,000	600,000

- process load time;
- construct extraction time;
- construct transformation time;
- construct load time.

The units of measure of all the time data are millisecond. Moreover, another important information for the data cleaner and reconciler evaluation is the throughput, which is the amount of work that the system component is able to handle in a unit of time [63]. In this thesis, in Workflow Management System context, the throughput is be defined as the number of process instances executed over the time required for the test to be performed. Thus, throughputs are considered for extraction, transformation, load operation for both processes and constructs data.

The results are shown on Tables 5.2, 5.4 and 5.3; in addition Figure 5.6 shows how the time related to the operations of extraction, transformation and loading varies when each operation is applied to an increasing number of constructs and similarly the Figure 5.5 shows the relation between the operations time and the increasing number of processes.

Table 5.2. Performance evaluation of data cleaner and reconciler

Execution number	Total time (s)	Maximum RAM used (MB)	Total number of threads (#)
1	4	235	10
2	7	321	18
3	10	410	26
4	19	658	42
5	64	891	62
6	85	1014	82
7	181	1208	122
8	331	1547	162

During all the executions, the number of cores used is 4.

Of course those data are linked, because the throughput is defined as the number of pro-

Figure 5.5. Evaluation of the execution time of the cleaner and reconciler according to process

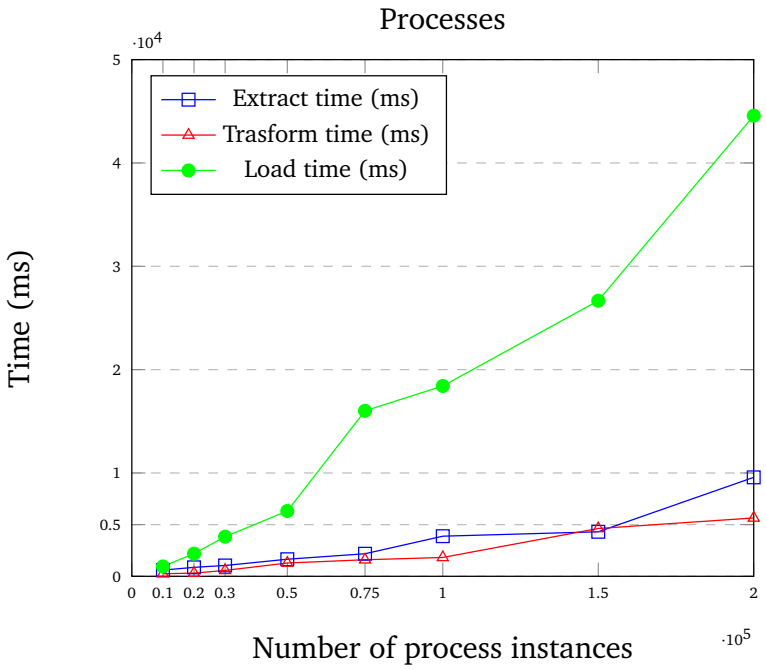


Figure 5.6. Evaluation of the execution time of the cleaner and reconciler according to construct

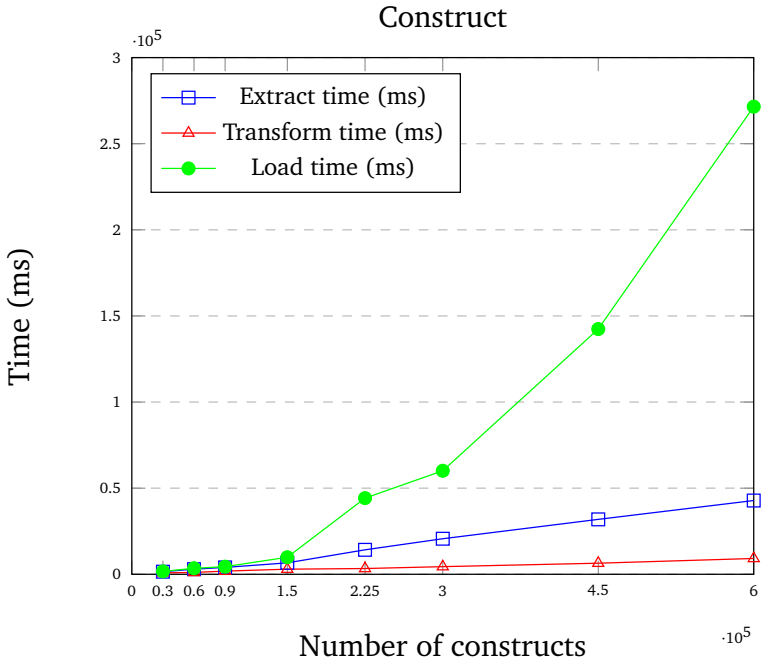


Table 5.3. Performance evaluation of data cleaner and reconciler according to process

Execution number	Number of processes	Time (ms)			Throughput ($\frac{\#process}{ms}$)		
		Extract	Transform	Load	Extraction	Transformation	Load
1	10000	599	249	952	16.69	40.16	10.50
2	20000	867	307	2193	23.07	65.15	9.12
3	30000	1047	571	3836	28.65	52.54	7.82
4	50000	1652	1286	6317	30.27	38.88	7.92
5	75000	2183	1601	16018	34.36	46.85	4.68
6	100000	3882	1823	18417	25.76	54.85	5.43
7	150000	4308	4626	26671	34.82	32.43	5.62
8	200000	9576	5647	44576	20.89	35.42	4.49

Table 5.4. Performance evaluation of data cleaner and reconciler according to construct

Execution number	Number of constructs	Time (ms)			Throughput ($\frac{\#construct}{ms}$)		
		Extract	Transform	Load	Extraction	Transformation	Load
1	30000	1497	838	1675	20.04	35.80	17.91
2	60000	2966	1068	3490	20.23	56.18	17.19
3	90000	3987	1838	4423	22.57	48.97	20.35
4	150000	6634	2966	9884	22.61	50.57	15.18
5	225000	14208	3339	44226	15.84	67.39	5.09
6	300000	20644	4433	60104	14.53	67.67	4.99
7	450000	31885	6435	142389	14.11	69.93	3.16
8	600000	42820	9180	271506	14.01	65.36	2.21

cess instances or construct instances executed over the time required for the test to be performed.

It is possible to see that the load phase is the most time expensive among the three phases and in particular the total time is mainly related to load time. Therefore, the majority of the optimization effort has been focused on the load phase. Additional performance incrementation can be achieved by removing some constraints, such as referential integrity, so that load times can be reduced causing a reduction of the total time. However this option should not take place, because it increases the possibility of having inconsistent data.

The load phase time depends mainly on two aspects:

- time required to build the insert query to execute;
- time required by the DBMS to actually insert records.

Moreover if the DBMS and the data cleaner and reconciler are not on the same machine, there is to consider also the time required to transmit the insert query to the DBMS. Indeed this time cannot be neglected, because each insert query consists of several thousands of records.

Additionally it is possible to see that the Extraction, Transformation and Load phases of processes are strongly related with Extraction and Transformation phases of construct. Their correlation is especially shown during the execution number 4, 5 and 6. Indeed they occur in parallel

and the resources are asked concurrently between process transformation and construct transformation; whereas the Load phase of construct does not influence the ETL phases of processes, because it occurs only after that the process load phase is concluded.

Finally the RAM memory used for the application should be considered, in point of fact all the data required by the data cleaner and reconciler component are temporarily maintained in the RAM memory. For the test performed, it has not been necessary to modify the default run configuration of stack memory and heap memory of Eclipse. Specifically the heap memory contains all the objects created by the application and it is by default to 2 GigaBytes in Eclipse. Therefore it is possible to conclude that for all the performed executions, this limit is never reached.

Anyway to have a more precise upper limit, VisualVM is used to analyze the RAM memory as shown in Table 5.2. This operation is performed on the execution number 8, which has the highest number of process instances and constructs compared to other executions, and it shows that the maximum amount of RAM memory used is 1547MB, which is approximately 25% less than the maximum RAM memory available for the execution.

In conclusion it is possible to state that the data cleaner and reconciler is able to manage WfMS's data considering that the RAM memory is a bottleneck of the tool, because all the data must be kept in RAM memory for being processed and that the load phase is the most time consuming phase.

5.6.3 Correctness

The correctness of data is very important when cleaning occurs. In particular it must be ensure that the transformed data maintain the original information and meaning [79].

Correctness is ensured by performing controls, which regard data aggregation on both source and destination databases. Mainly, the performed controls are counting the number of records in the Workflow Management System database and the number of new records in the CleanRawData database and these values must be equal to ensure that no data record is lost.

Additional control can be done by performing data aggregation on field values. Specifically Duration field in Process and Construct tables of the CleanRawData database is optimal to perform this check. Whereas on the WfMS database performing this computation might require more complex computation.

Another important characteristic is that the relationship between process instances and constructs of these process instances is maintained. In particular this characteristic is ensured by the referential integrity constraint enforced on the database. Indeed each process instance in the Construct table is a foreign key with a reference to the primary key of the Process table.

This constraint has dictated the requirement to load at first the process records into their specific table and only when the process load is completed to start construct storing. By removing this constraint it is possible to reduce cleaning duration, but it will be required to apply an expensive control at the end of the loading phase that might nullify the time reduction.

In addition, during data cleaner and reconciler testing by applying JUnit tests, it has been possible to verify the correctness of data transformation to clean record. The unit tests have covered

both Process, Construct and Environment tables fields and records transformation.

Finally correctness can be ensured by checking that the value of a record in the data cleaner and reconciler database is equivalent to the record on the source database from which it is derived. This control is performed by applying the reverse mapping (See Section 5.6.4).

5.6.4 Reverse mapping

The reverse mapping is made easier since the data reference cannot change. Indeed the references do not change, since the trial is concluded when the data is cleaned. However, reverse mapping requires that the user must be able to retrieve the location where he/she has stored the original data. This is helped by providing to the user the possibility to know the trial at which the data refers to by accessing the Experiment table. In any case the users have to organize the source data to access them according to the trial.

For environment records the reverse mapping is enabled by specifying the line of the log file at which the record refers to. For process records, the reverse mapping is possible by SoucesProcessID field. As a matter of fact this field contains the original identification values used by the Workflow Management System database. Hence it enables to identify all records related to the process instances on the WfMS database. For construct records, the reverse mapping is possible by SouceConstructID field. Indeed this field contains the original identification values used by the Workflow Management System database. Therefore it enables to identify all records related to the constructs on the WfMS database.

In addition, thanks to the referential constraint between the Process table and the Construct table, it is possible:

- given a process instance in the CleanRawData database, to determine the constructs of a process instance and to get the original data of these constructs.
- given a construct in the CleanRawData database, to determine the process instance of which the construct belong to and to get the original data of this process instance.

5.6.5 Scalability to the adding of new WfMSs

WfMSs use DBMS to store persistently their data and the data cleaner and reconciler is designed to integrate different source databases of different Workflow Management Systems. The stages required by the data cleaner and reconciler are extraction, transformation and loading.

The extraction phase is simplified and standardized by applying OQL technology; indeed when a new WfMS is added, it is not required to implement this stage. It is possible to decide which table should be extracted from the database and the type of SQL dialect to apply to interact with the database. However according to the jOOQ framework applied, it is required to generate the Java classes that model the source database schema of WfMS DBMS. Specifically the framework creates Java classes related to the tables and their fields.

Also the loading phase is standardized and it does not require any type of modification when a new Workflow Management System is added. In point of fact loading occurs only for records

that are already transformed according to the destination schema. Therefore, until the schema of the database remains similar to the actual one, it is not required to change it.

Finally the transformation phase is the only phase that requires to be actually modified when a new Workflow Management System is added. Indeed in general the database schema between different WfMS is different. However Java classes that represent the database schema with its tables and fields are very useful during the transformation phase; in effect they simplify the mapping between source and destination fields.

Chapter 6

Performance meter and data aggregator

The objective of the performance meter and data aggregator is to aggregate stored data into the data cleaner and reconciler database to obtain information. Data aggregation is performed according to metrics application based on the experiments and on the trials related to a specific experiment.

6.1 Design performance meter and data aggregator database

To realize the performance meter and data aggregator, a common database, which will contain aggregated data, must be used. The main requirement for this database is the possibility to have a schema that can be easily extended by adding additional fields, which will contain new metrics values. As a matter of fact it is fundamental that the performance meter and data aggregator can increase the applicable metrics range avoiding additional costs as much as possible.

In addition it is important to have the possibility to add additional metrics to a record after the record's creation. Since non-relational databases have native schemaless structure, it is used MongoDB (for more details see Section 6.2). The metric level of aggregation can occur at two levels:

- process
- construct

The process aggregation level combines data related to the process instances that belong to process models instantiated by trial experiment. Instead the construct aggregation level groups together data related to the constructs that belong to the process instances executed by trial experiment. Therefore two collections are used: one for each aggregation level. In particular records at process aggregation level have as basic fields: experiment, repetition number, process model and metrics fields. Whereas records at construct aggregation level have as basic fields: experiment, repetition number, construct name and metrics fields.

Listing 6.1. Process record

```
{
 "_id": ObjectId(value),
 "experiment": value,
 "repetition": value,
 "processDefinition": value,
 "metricName": metricValue,
}
```

Listing 6.2. Construct record

```
{
 "_id": ObjectId(value),
 "experiment": value,
 "repetition": value,
 "construct": value,
 "metricName": metricValue,
}
```

6.2 Implementation of performance meter and data aggregator database

To satisfy the requirement of the performance meter and data aggregator database (see Section 6.1), it is possible to apply an Entity Attribute Value (EAV) data model [54] or use a non-relational database.

The Entity Attribute Value (EAV) pattern [54] is used in databases where a potentially large number of parameters could describe something and relatively few apply to a given instance. Thus the EAV pattern is applied to relational database to compensate the problems of having a fixed schema in the database [54]. In particular the EAV table, consists of three fields:

- an Entity field, it describes the data item in the EAV database;
- an Attribute field, it describes attributes of entities, the attribute is a foreign key that has general information of every thing, called object, in the EAV database;
- a Value field contains the attribute value, that depends on the type of data.

All the type of facts are in the same "value" column so EAV data have the necessity to be transformed into one column per parameter structure, this operation is called pivoting. EAV pattern is widely used for clinical data repositories [31], but in other fields is not so popular due to its disadvantages. The main disadvantages of EAV are:

- Size of data: according to Chen et al. "the EAV representation consumed approximately four times the storage of our conventional schema." [22]. Indeed the application of EAV pattern implies a redundancy of the information.
- Data retrieval: relational databases are not designed to effectively organize and retrieve data in the EVA format. As a matter of fact relational database supports attribute centric queries, whereas EAV uses entry centric operation. Thus the query could be slower using EAV pattern, particularly if multiple criteria exist. [22, 31, 71]
- Columnar attribute representation: EAV does not work well for a class with a potentially high number of attributes due to its storage characteristic of information redundancy. In effect EAV is applied in clinical database because only a small number of parameters are recorded for each patient. [31]

- Relational databases advantage lost: applying EAV pattern some advantages of the relational databases are lost, such as the data integrity.

So applying EAV pattern to a relational database limits relational database advantages and it has significant drawback; in fact it is considered an antipattern. For these disadvantages, an alternative to relational databases are non-relational databases. The non-relational databases are generally called NoSQL databases, [81], and one DBMS belonging to this family is chosen for the implementation of the performance meter and data aggregator database.

In the last years the increase number of distributed databases has pushed researchers to develop alternative options of handling data at scale. Developers have started to use not only relational databases, which organization is found on relational model data, [24]. For more information about relational database used in WfMS, read Section 2.2.2. They have also developed and used non-relational databases, which have schemaless data structures, very simple replication, high availability, scale horizontally and alternative querying methods.

According to Vaish [94], the NoSQL databases are able to solve the horizontal scale which is a challenge to obtain in a relational database, due to the consistency among the units of relational database data. Thus the advantages of a NoSQL database are:

- Representation of schemaless data; so the database allows the storage of any type of data without any definition given before. The user can modify the database structure over time due to the schemaless data.
- Uninterrupted data availability; it prevents the failure of the system. Indeed even if a server fails, the system is able to do not lose data and continue to operate.
- Consistency, Availability, Partition tolerance (CAP) theorem[1] [18]; anyway the NoSQL database does not have the four key characteristics that belong to the relational database: atomicity, consistency, isolation and durability.

The NoSQL database can be categorized by the characteristic of their data store. The four database types are:

- Key-Value Store, which stores data as pair of key and values;
- Column Oriented Store, which stores data as section of columns;
- Document Store, which stores data as documents;
- Graph Store, which stores data as nodes, edges and properties according to graphs design.

In this thesis the database management system chosen is MongoDB. Even if MongoDB's characteristics are less performant compared to other NoSQL database, especially in updating documents [9]. However it is chosen, because stored document has an internal structure, the reading of an entire document is fast and implementing programs, using MongoDB, is easy enough due to its similarities with relational database.

MongoDB is a NoSQL database and it is a type of Document Store database type, which means that it stores data as binary-encoded serialization document, which are called BSON. It organizes together BSONs with similar characteristics in collection like tables in relational database,

[1] CAP theorem is not required to reach the goal of this thesis; however future works might have to consider the CAP theorem.

but each document in the same collection can have a different structure. In particular each database can organize documents according to collections, which are a set of documents inside a database and, in principle, documents inside the same collection should have some similarities. For these reasons the data performance meter and data aggregator uses a process collection and a construct collection.

MongoDB provides availability, reliability and high performance without separating application to handle any kind of operations. Indeed it can use a primary replica[2], which is a server that manages the totals of the write operations and secondary replicas, which read and apply the write operations. MongoDB has a strong reliability because if the primary replica is disconnected, one of the secondary replicas can replace it through an election process and the system does not fail. Moreover, it has a high availability due to the duplication of the data, it is very flexible due to its schemaless characteristic and at the same time it has a full query language and consistency. Finally, MongoDB can apply horizontal scalability. De facto MongoDB can spread data among various servers, called shards[3], overcoming the hardware limitation of the use of a single server.

Considering the schemaless structure of data in MongoDB databases, the set of metrics fields that each document has, depends on metrics computed over the experiment or the trial at which the document refers to, on the type of collection which the document belongs and to the user decision. In addition each document has a unique identification that is automatically generated and managed by MongoDB.

6.3 Design of the performance meter and data aggregator

The performance meter and data aggregator must allow to add new metrics incrementally, so a hierarchy relative to metrics is created, where the superclass is called Metric. Metric class is an abstract class with an abstract method called compute, which must be implemented to aggregate data according to the metric implemented. In addition Metric class contains all information related to the experiment and the trial that has to be processed. In effect, the user must specify which trial or experiment the performance meter and data aggregator has to deal with when it is used.

The Metric class has two abstract subclasses: ProcessMetric and ConstructMetric subclasses. The ProcessMetric class is designed to be extended for metrics that compute process level metrics, whereas construct level metric extends the ConstructMetric subclass. Moreover Metric, ProcessMetric, ConstructMetric classes manage the connection and basic interactions with the performance meter and data aggregator database. For example these basic interactions regard the extraction of documents related to an experiment or a trial, the insertion of a new document and the update of an already inserted one.

The concrete subclasses of ProcessMetric and ConstructMetric classes have to compute the metric according to their specification and they have insertion methods exposed by their superclass to store computed metric values. This is obtained by overriding and implementing the compute

[2] The described mechanism has not been applied and it is described to show the possibilities enabled by the use of MongoDB.

[3] See Footnote 2

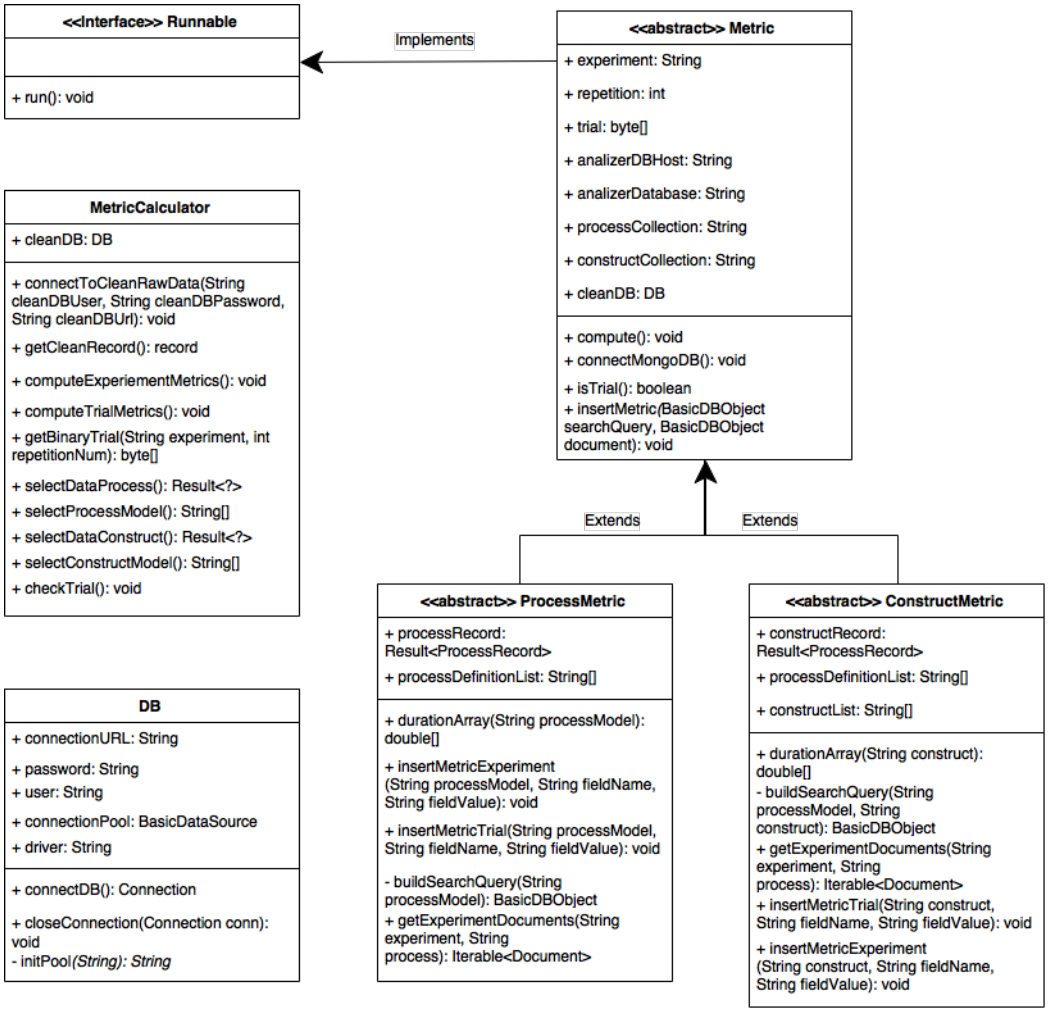


Figure 6.1. Performance meter and data aggregator Java classes - part 1

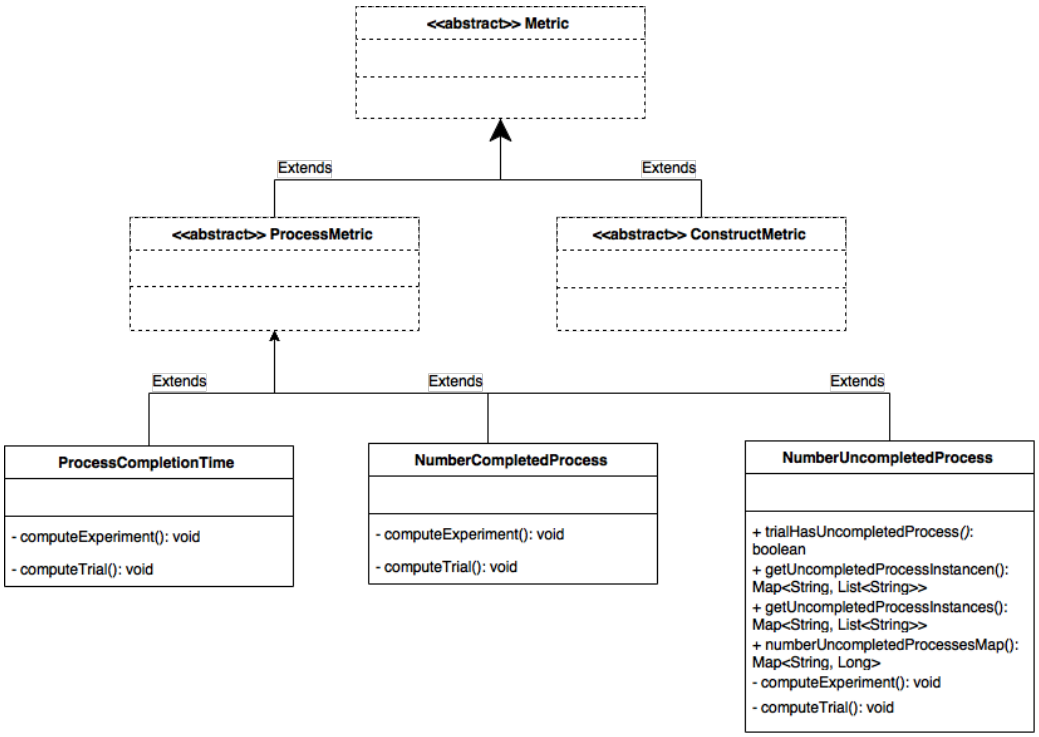


Figure 6.2. Performance meter and data aggregator Java classes - part 2

method. Moreover they have to allow computing the metric for both experiments and trials. Therefore to add new metrics it is necessary to extend one of the two types of metric subclass: ProcessMetric and ConstructMetric. In addition if an already implemented metric requires to be modified to include a new value it is possible to extend directly the metric implemented and extend its compute method.

Finally, MetricCalculator class uses the Metric class to actually instantiate concrete Metric's subclasses to compute their related metric. The MetricCalculator has to instantiate the concrete Metric object and call the compute method, which is defined in the Metric class. It provides data to concrete Metric's subclasses, so it manages interaction functionalities with the CleanRawData database; in particular it establishes the connection with the database and it reads interesting records that are passed to concrete Metric's objects for computations. In addition it determines the list of process models and constructs belonging to process models, so that the concrete Metric's subclasses can have these additional information for their computations. Moreover, MetricCalulator has an additional option that identity if a trial has process instances that are not completed before that metrics computation is performed. This option can be used when there are trials that test WfMS on heavy workload and uncompleted processes instances are unwanted.

6.4 Implementation of the performance meter and data aggregator

The performance meter and data aggregator is implemented in Java. Since it has to interact with the CleanRawData database, as the data cleaner and reconciler, Java Object Oriented Querying framework is used as Object-Oriented Query Language (OQL) technique. In addition the analyser has, also, to interact with the non-relational MongoDB database and for this interaction is used the default MongoDB driver. The connection to the MongoDB database is performed by providing a connection string and a database name. Even if different connection and authentication mechanisms exist, connection string is chosen, because it is the only one that remains consistent for different versions of MongoDB.

About the Metric subclasses, ProcessMetric and ConstructMetric maintain the interesting records for metric computations and additional fields that simplify the metric computations, such as the list of process models belonging to trial or experiment. Moreover they provide the durationArray method to retrieve partially processed data about durations that can be used to simplify metric computation. In addition multiple metric computation can be performed simultaneously, since the Metric class extends the Runnable interface.

Finally the data must be read from the data cleaner and reconciler database only once and used for all metrics without further readings. So that multiple reading of the same data from different metrics are avoided. As a matter of fact if different metrics have to read same data, every metric creates a high overhead and it will cause a reduction of overall performance. For this reason MetricCalculator is the only component that interact with the CleanRawData database.

6.4.1 Performance meter and data aggregator: implemented metrics

Finally some metrics are implemented according to the performed analysis on Chapter 4. They extend the ProcessMetric class (see Figure 6.2) and, in particular, the following classes, which are created in Section 4.1.2, are defined:

- ProcessCompletionTime,
- NumberCompletedProcess,
- NumberUncompletedProcess.

ProcessCompletionTime class computes the completion time metric of process models belonging to trials and experiments. According to completion time specification, the class computes the different metrics values, such as: average, standard deviation, minimum, maximum and median value. The ProcessCompletionTime compute method pseudocode shown in Listing 6.3.

Listing 6.3. Pseudo-code of ProcessCompletionTime compute method

```
if(isTrial()){ //trial aggregation level
  for each process_model {
    duration_array = get_duration(process_model)
    avg = StatUtils.avg(duration_array)
    std_dev = FastMath.sqrt(StatUtils.variance(duration_array))
```

```
    max = StatUtils.max(duration_array)
    min = StatUtils.min(duration_array)
    median = StatUtils.median(duration_array)
    insert(process_model, "avgCompletionTime", avg)
    insert(process_model, "maxCompletionTime", max)
    insert(process_model, "minCompletionTime", min)
    insert(process_model, "stdDevCompletionTime", std_dev)
    insert(process_model, "medianCompletionTime", median)
  }
}else{ //experiment aggregation level
  for each process_model{
    docs = getExperimentDocuments(experiment, process)
    for each docs{
        avg_comp_time.add( doc.get("avgCompletionTime") )
        min_comp_time.add( doc.get("minCompletionTime") )
        max_comp_time.add( doc.get("maxCompletionTime") )
        median_comp_time.add( doc.get("medianCompletionTime") )
    }
    avg = StatUtils.avg(avg_comp_time)
    std_dev = FastMath.sqrt(StatUtils.variance(avg_comp_time))
    max = StatUtils.max(max_comp_time)
    min = StatUtils.min(min_comp_time)
    median = StatUtils.median(median_comp_time)
    insert(process_model, "avgCompletionTime", avg)
    insert(process_model, "maxCompletionTime", max)
    insert(process_model, "minCompletionTime", min)
    insert(process_model, "stdDevCompletionTime", std_dev)
    insert(process_model, "medianCompletionTime", median)
  }
}
```

NumberCompletedProcess class computes the number of process instances that have reached process completion according to process models. The metric aggregation can be performed at trial level or experiment level. At trial aggregation level, the class produces a single value. Whereas in the latter case it is used an aggregation of data retrieved from the performance meter and data aggregator database and it is possible to compute average, standard deviation, minimum, maximum and median value at experiment level. The pseudo-code of compute method in the NumberCompletedProcess class is shown in Listing 6.4.

Listing 6.4. Pseudo-code of NumberCompletedProcess compute method

```
if(isTrial()){ //trial aggregation level
  for each process_model {
    count = 0
    for each processRecord{
      if(processRecord.get("Duration") != NULL){
      count++
    }
```

```
  }
  insert(process_model, "numberCompletedProcess", count)
  }
}else{ //experiment aggregation level
  for each process_model{
    docs = getExperimentDocuments(experiment, process)
    for each docs{
      process_count.add( doc.get("numberCompletedProcess") )
    }
    avg = StatUtils.avg(process_count)
    std_dev = FastMath.sqrt(StatUtils.variance(process_count))
    max = StatUtils.max(process_count)
    min = StatUtils.min(process_count)
    med = StatUtils.median(process_count)
    insert(process_model, "avgNumberCompletedProcess", avg)
    insert(process_model, "maxNumberCompletedProcess", max)
    insert(process_model, "minNumberCompletedProcess", min)
    insert(process_model, "medianNumberCompletedProcess", med)
    insert(process_model, "stdNumberCompletedProcess", std_dev)
  }
}
```

Finally NumberUncompletedProcess class is similar to the NumberCompletedProcess class, but it is used to compute number of process instances that have not reached process completion. In addition to the compute method, it provides additional methods which are used by the MetricCalculator to identity if a trial has process instances that are not completed. The pseudo-code of compute method in the the NumberUncompletedProcess class is shown in Listing 6.5.

Listing 6.5. Pseudo-code of NumberUncompletedProcess compute method

```
if(isTrial()){ //trial aggregation level
  for each process_model {
    count = 0
    for each processRecord{
        if(processRecord.get("Duration") == NULL){
            count++
        }
    }
    insert(process_model, "numberUncompletedProcess", count)
  }
}else{ //experiment aggregation level
  for each process_model{
    docs = getExperimentDocuments(experiment, process)
    for each docs{
        process_count.add( doc.get("numberUncompletedProcess") )
    }
    avg = StatUtils.avg(process_count)
    std = FastMath.sqrt(StatUtils.variance(process_count))
```

```
    max = StatUtils.max(process_count)
    min = StatUtils.min(process_count)
    med = StatUtils.median(process_count)
    insert(process_model, "avgNumberUncompletedProcess", avg)
    insert(process_model, "maxNumberUncompletedProcess", max)
    insert(process_model, "minNumberUncompletedProcess", min)
    insert(process_model, "medianNumberUncompletedProcess", med)
    insert(process_model, "stdNumberUncompletedProcess", std)
  }
}
```

For the computation of statistical value is used the Apache Commons Math, which is a mathematical library. It is used, because it is well-documented, it provides a large set of mathematical operators that can be used for metrics computation and its components are highly reusable.

6.5 Performance meter and data aggregator evaluation

The objective of the performance meter and the data aggregator evaluation is to aggregate standardized data from the CleanRawData databases. To evaluate if the tool is able to reach its objective, the most important characteristics to consider are:

- Performance
- Scalability

The performance meter and the data aggregator must be able to compute metrics requiring the lower time as possible. Effectively it has to process a large amount of data and it produces an less large amount of information. Last, but not least the performance meter and the data aggregator must be able to manage and store the information generated by the standardized data; thus, its scalability in amount of data is evaluated.

The evaluation is performed on MacBook Pro machine with the 10.9.5 OSX version, 8GB of DDR3 RAM and 2.4 GHz quad cores processor. The source databases are 5.5 version of MySQL database, while the performance meter and data aggregator database is 2.6.10 version of MongoDB database. Both databases are run on localhost through Docker containers and Docker is run in a Ubuntu 14.04 LTS Virtual Machine using Virtualbox, whereas the performance meter and data aggregator is run by Eclipse version 4.4.1 with default settings and with Java version 1.8.0_05. For more details about the used infrastructure read Section 7.2.

6.5.1 Performance

To understand performance of the performance meter and the data aggregator, the times to compute the following operations are measured:

- computation of the ProcessCompletionTime,
- computation of the NumberCompletedProcess,

- computation of the NumberUncompletedProcess.

In addition for trial metric computation also the extraction of clean process data and the extraction of clean construct data are considered. For trial metrics computation, these durations are evaluated for all the eight executions that have been performed for the data cleaner and reconciler (see Table 5.1). Instead the evaluation of the performance meter and data aggregator about the computation of the metrics at experiment level of aggregation is performed considering 10 trials of the execution 8. Only 10 trials are chosen, because 10 can be considered a relevant number [61].

The results about the computation of the metrics at trial level of aggregation are shown on Tables 6.1 and 6.2. Whereas the results of the computation at experiment level is shown in Table 6.3. The unit of measurement of all the time data is millisecond.

Table 6.1. Extraction step evaluation

Number	Extract (ms)	
of processes	constructs	processes
10000	3065	491
20000	3864	1102
30000	4629	1401
50000	6870	2138
75000	9428	2796
100000	11058	3165
150000	18018	5299
200000	23383	5979

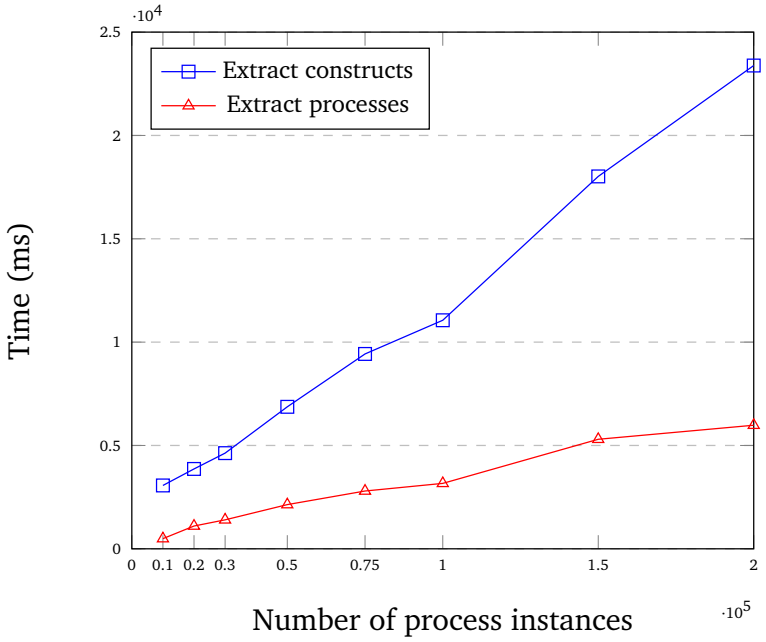
Table 6.2. Evaluation at trial level of metrics computation step

Number	Time to compute (ms)		
of processes	CompletionTime	NumCompletedProcess	NumUncompletedProcess
10000	199	50	42
20000	196	50	39
30000	195	51	50
50000	221	44	30
75000	252	62	35
100000	274	68	40
150000	317	89	51
200000	403	90	69

Table 6.3. Evaluation at experiment level of metric computation step

Number	Time to compute (ms)		
of trial	CompletionTime	NumCompletedProcess	NumUncompletedProcess
10	289	79	59

Figure 6.3. Extraction step evaluation

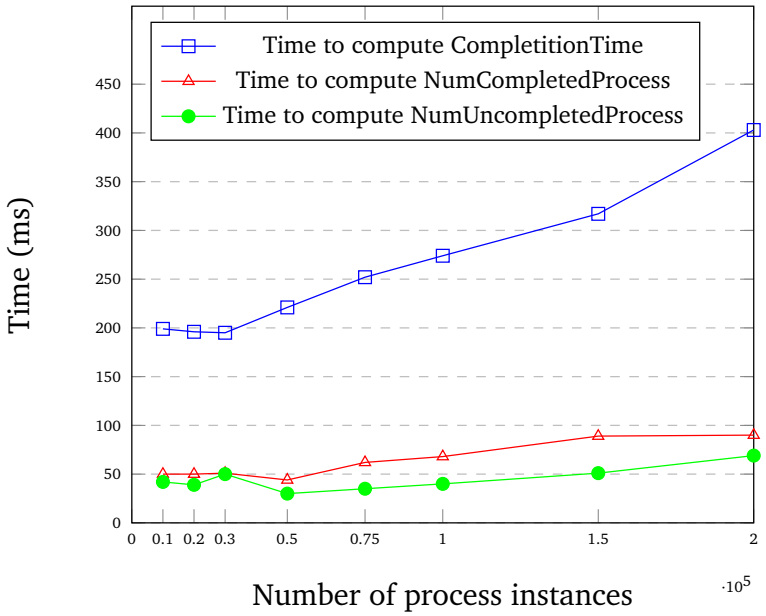


About metrics computation it is possible to see that the time required by the ProcessCompletionTime is about four times the other two metrics computation duration. It can be explained by considering that this metric computes five values more than the other metrics (see Section 6.4.1).

At trial level, the increase of the amount of data for computing metrics can be also analyzed by looking the Figures 6.3 and 6.4. It is possible to understand that the time duration, required to compute the metric and to write the document to the MongoDB, increases approximately linearly with the increase of the data to be processed. In addition metrics computation can be performed simultaneously, since the Metric class extends the Runnable interface. Therefore each metric can be computed on a dedicated thread. Anyway it should be noticed that some metric could be dependent on other metrics, therefore these special cases should be appropriately addressed in the MetricCalculator class by computing the independent metrics first and the dependent metrics later.

Instead, comparing at trial aggregation level, the extraction operations with the metric computation operations, approximately the first operations require more than ten time the time of the latter operations. So it is possible to see that retrieving the clean data from the CleanRawData database is an expensive task in term of resources and time (see Table 6.1). Fortunately data extraction is performed only one time for all metrics, so that each metric does not have to repeat the same operation, and its duration can be split over metric computations. Moreover the one time extraction reduces overheads on the DBMS, which can better exploit the resources available to respond to this single request. In addition connecting to the clean DBMS is one of the most time expensive operation and for this reason it is managed centrally by the MetricCalculator class.

Figure 6.4. Computation step evaluation



Finally the RAM memory used for the trial metric computations of the execution 8 is analyzed by applying VisualVM. This operation is performed for the execution number 8, because it has the highest number of process instances and constructs compared to other executions. The result is that the maximum amount of RAM memory used is 614MB, which is less than half the RAM required by the data cleaner and reconciler (See Section 5.6).

6.5.2 Scalability in amount of metrics

To add a new metric it is required to extend one of the two abstract Metric subclasses: ProcessMetric or ConstructMetric. According to the metric's characteristics, the compute method must be implemented considering that this method calculates both trial metric values and experiment metric values. In addition this new metric must be added to the MetricCalculator in the method for computing metrics by creating a Metric object that instantiates the new metric and calls the compute class. This is necessary, because discovery of subclasses is not implemented and it is left as future work.

Every new metric, added to the tool, increases the number of fields that have to be stored in the database. So it is possible to understand that the performance meter and data aggregator can potentially generate an infinitely large amount of data. Moreover every trial generates an elevate number of new documents that must be stored in the performance meter and data aggregator database. Anyway a non-relational database such as MongoDB ensures the possibility to store an almost infinite amount of data. Indeed MongoDB has a mechanism called sharding[4] that allows to split data among different machines, responding to the increasing demand of

[4]The sharding mechanism has not been applied and the following discussion is based on theoretical deductions [27].

data storage [27].

To apply the sharding mechanism it is required to configure MongoDB and to apply a shard key, which determines how data are split among the different shards. The decision of the shard key is based mainly on four characteristics:

1. shard key should have a high cardinality, so the number of values that shard key can take is large;
2. shard key improves the writing operations if shard key is able to identify different shards on which to write to;
3. shard key improves the querying operations if the documents selected by the query are on the same shard;
4. every document must have the shard key; it is required to write the document on the correct shard.

According to the first and third characteristics of shard key, it is suggested to apply a compound shard key that uses the experiment and the repetition number field. This should not improve write operations, but it should improve query operations.

Chapter 7

Proof of concept

7.1 Disclaimer

This Chapter of the thesis is provided as a proof of concept, with all limitations and simplifications that a proof of concept can have. Only the two tools developed are used and not the entire BenchFlow framework. Hence this Chapter **must not** be considered as part of the BenchFlow project, because only the two tools developed are applied, instead of its entire infrastructure and components. In particular, the infrastructure does not respect the BenchFlow project standards and requirements, in fact the proof of concept is performed on a single low-power machine, instead of using a well controlled Cloud infrastructure [45]. See Section 7.2 for the detailed information about the used infrastructure in the thesis.

In addition, the results obtained are derived from only one trial of the experiment due to time constraint reasons and they are influenced by the used infrastructure, due to the single machine supporting the WfMSs' executions together with the driver and the DBMSs of the WfMSs. Therefore the results obtained are only intended to demonstrate that the types of discussed analyses are feasible.

7.2 Infrastructure and System Under Test

The infrastructure is based on a single machine: one MacBook Pro machine with the 10.9.5 OSX version and 64 bit architecture. It has 8 GigaBytes of DDR3 RAM memory with frequency of 1600 MHz and 2.4 GHz Intel Core i7 quad cores CPU processor.

The WfMSs' DBMSs and also the CleanRawData databases uses of MySQL 5.5.43. Instead the MongoDB 2.6.10 database is used by the performance meter and data aggregator to store data. All the databases run on localhost through Docker containers pulled from the official Docker repository. Since the machine uses OSX operating system, Docker is run in a Ubuntu 14.04 LTS Virtual Machine using Virtualbox. The MySQL Docker image is pulled from the official MySQL repository and the image is called "mysql:5.5"; whereas the MongoDB image is pulled from the tutum repository and the image name is "tutum/mongodb:latest".

The data cleaner and reconciler and the performance meter and data aggregator are run as Java Application through Eclipse Integrated Development Environment (IDE). The Eclipse version used is Luna Service Release 1 (4.4.1) and the Oracle Java version used is the 1.8.0_05.

Instead, Activiti 5.17.0 and Camunda 7.2.0 are deployed on a TomCat 7.0.59 server ran on localhost. For the evaluations, Camunda is used to generate the data that are processed by the data cleaner and reconciler. For both WfMSs data generation is performed by exploiting the REST API services that they provide. Whereas JMeter is used to send request of processes instantiations, in fact it is a load tool that is able to interact through REST requests; in particular after that a request is sent, it waits a reply before that a new request can be issued. The version used is the 2.13 and it is used in its graphical interface mode. After that the data are generated, they are dumped manually from the source database by using the mysqldump command, since the collector component is not yet implemented.

7.2.1 Camunda

Camunda is an open source Business Process Management System and thus a Workflow Management System [2]. According to Camunda terminology, the main components are:

- Process Engine, which is the Workflow Engine in the standard terminology;
- Tasklist, which is the Worklist Handler in the standard terminology;
- Camunda Modeler, which is the Workflow Modeler in the standard terminology;
- Custom Applications, which are the invoked applications in the standard terminology and they interact with the Process Engine by REST API or Java interfaces;
- Database and file repository.

In particular, it has to be noticed that the key component of Camunda is the Process Engine that allows Camunda to support BPMN 2.0 constructs. It is written in Java and it executes process models that respect BPMN 2.0 notation and that are written in XML format. Additionally process models must be transformed into graph structures to be executed by the Process Engine. Specifically Process Engine parses BPMN 2.0 files and each construct is transformed into an object that implements constructs behaviours.

The Process Engine is composed by [2]:

- BPMN 2.0 Core Engine, that is the main part of the Process Engine. It is used to parse process models written in XML format, produces the graph structure and constructs objects, and finally it executes the graph structure that translates the process models instantiated.
- Job Executor, it manages asynchronous executions of constructs.
- Public API, since Camunda is developed in Java, it provides a service oriented API in Java for interactions and, in addition, it provides REST API to connect to its services. Therefore, for interactions it has Java interfaces and REST APIs.
- Persistence Layer, it stores data received from the Process Engine on the database, which is a relational database. In particular, Process Engine maintains the state or process in-

stances in the database and it uses the Persistence Layer to access stored data in the database.

7.2.2 Activiti

Activiti is Java based and it runs BPMN 2.0 process models according to the Process Virtual Machine design pattern, which is used to create executable state machines; so that process models and constructs are translated into graphs and objects [1].

Activiti WfE is composed by a set of services that allow the processes execution and the most important services are:

- Repository service, it manages the process models and their deployment, which consists in inspecting, parsing and storing the process models for later executions.
- Runtime service, it regards starting and execution of process instances from process models previously deployed and it is also responsible of the data required for the process instances execution.
- Task service, it deals with constructs that require a human user's interaction, so it interacts with the Worklist Handler.
- History service, it retrieves data from the Workflow Engine, it stores them on the database and it allows the Activiti WfE to access stored data such as process models. The database is also used as repository, which means that it stores process models in addition to static information, runtime data and other data.

Other Activiti WfE services are: identity service, which manages groups and users; form service, which is an optional functionality that manages forms; and management service that is used for managing the metadata.

7.3 Test design

7.3.1 Performance test and workload

The performances of the WfMSs are the focus of the BenchFlow framework, for this reason the proof of concept applies performance tests and in particular load tests are used, because they simulate real workload applications to the WfMS (see Section 2.3). The workload defines interactions with the WfMS and, according to Menasce and Almeida [68] definition, the workload used for testing can be classified as the kernel one (see Section 2.3). As a matter of fact, it is characterized by requests of process models executions, where process models have similarities with real process models. In fact, the process models used are based on real processes and they have the main BPMN elements and structures of real process models. Specifically processes models are extracted from a collection provided by IBM regarding insurance sector (see Section 7.3.1.1). Moreover, since the process model used are real business process, they are anonymized.

7.3.1.1 Process models derivation

The workload is defined as all the inputs that are sent to the system under test [39] and, in the WfMS context, it is composed by a set of process models execution requests that are sent to WfMS.

In this thesis, the process models are derived from a collection of real insurance processes called "Insurance Process and Service Models" provided by IBM. The collection has 400 process models and, according to Ivanchikj work [53], they are classified in clusters based on their static characteristics [28, 53]: in particular, those 400 process models are divided in six clusters. Executing all process models of the collection is highly expensive and time consuming, because they have to be made executable on the selected Workflow Management Systems. For this reason only four process models from "Insurance Process and Service Models" collection are selected to identify a set of process models for the kernel workload.

Two of the selected process models are the ones Ivanchikj identifies as the representative process models of the cluster 3 and 4 of the "Insurance Process and Service Models" collection. The representative process model of the cluster 3 has distance of 1.049 from the derived model of the cluster itself; whereas the representative process model of the cluster 4 has a 1.273 distance from the derived model of the cluster 4 (see [53] for more details). The distance of a process model to the derived model of the cluster expresses the similarity that the first has with the later and a distance closer to 1 means higher similarity. The structural characteristics of the representative process model of the cluster 3 shows that it can follow a single path without loops; whereas the representative process model of the cluster 4 has a complex structure that allows to loop. In addition to these two process models, arbitrarily other two process models are selected that represent similarities with the discussed process models in term of their structural characteristics.

Table 7.1. Process models characteristics

Process model name	Number of Activities	Number of Exclusive Gateways	Number of Parallel Gateways	Number of End events
processC3P1	10	4	0	1
processC3P2	8	3	0	3
processC4P1	14	3	2	2
processC4P2	14	5	2	2

In the Table 7.1 the process models selected are described according to the static characteristics and, in particular, the number of: activities, start and end events, exclusive and parallel gateways. Instead the process models images are shown in the Figures: 7.1, 7.2, 7.3 and 7.4; where Figures 7.2 and 7.4 are the representative process models of cluster 3 and 4 respectively and the Figures 7.1 and 7.3 are the process models of cluster 3 and 4 respectively that have similar structural characteristics to the representative process models.

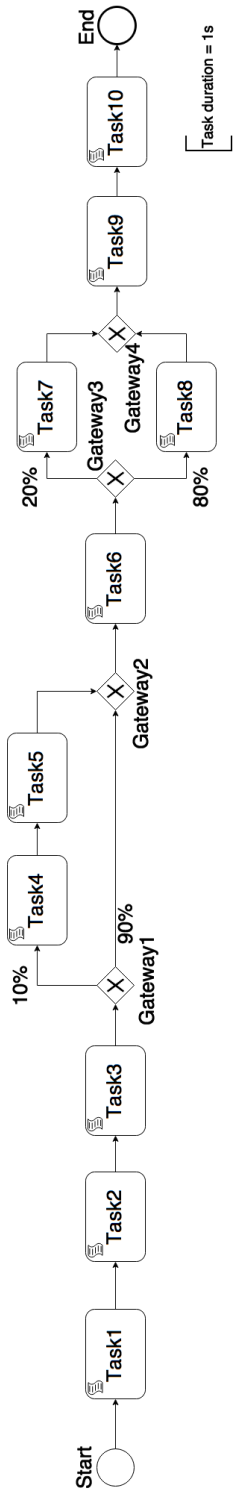


Figure 7.1. Process model 1 of cluster 3

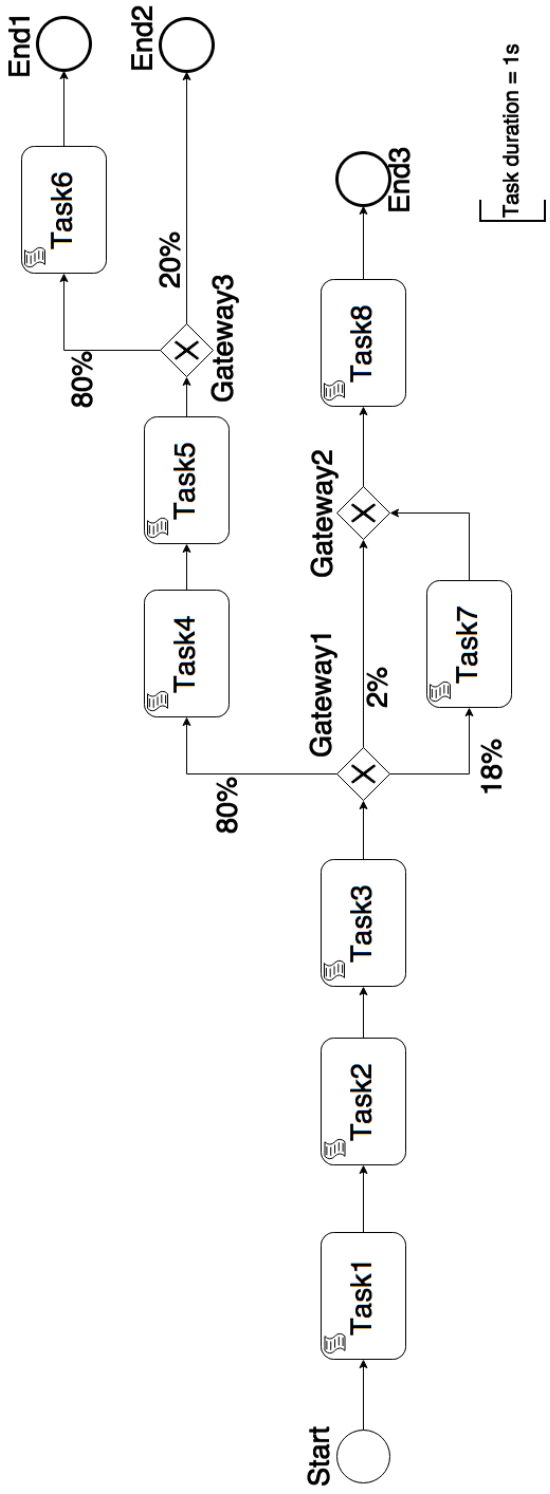


Figure 7.2. Process model 2 of cluster 3

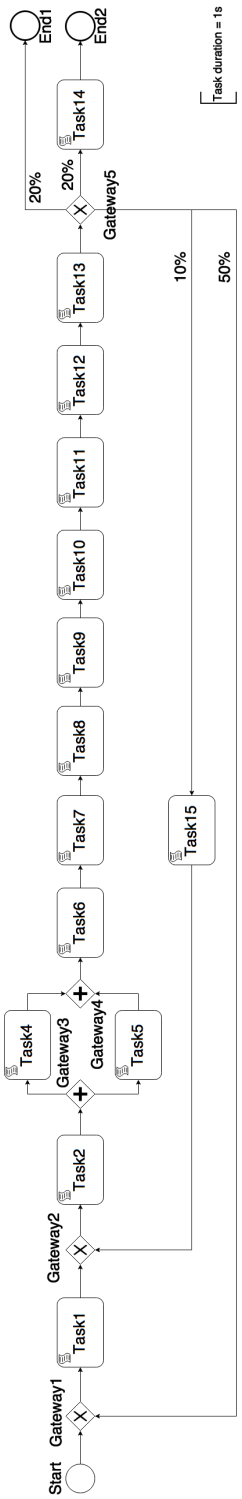


Figure 7.3. Process model 1 of cluster 4

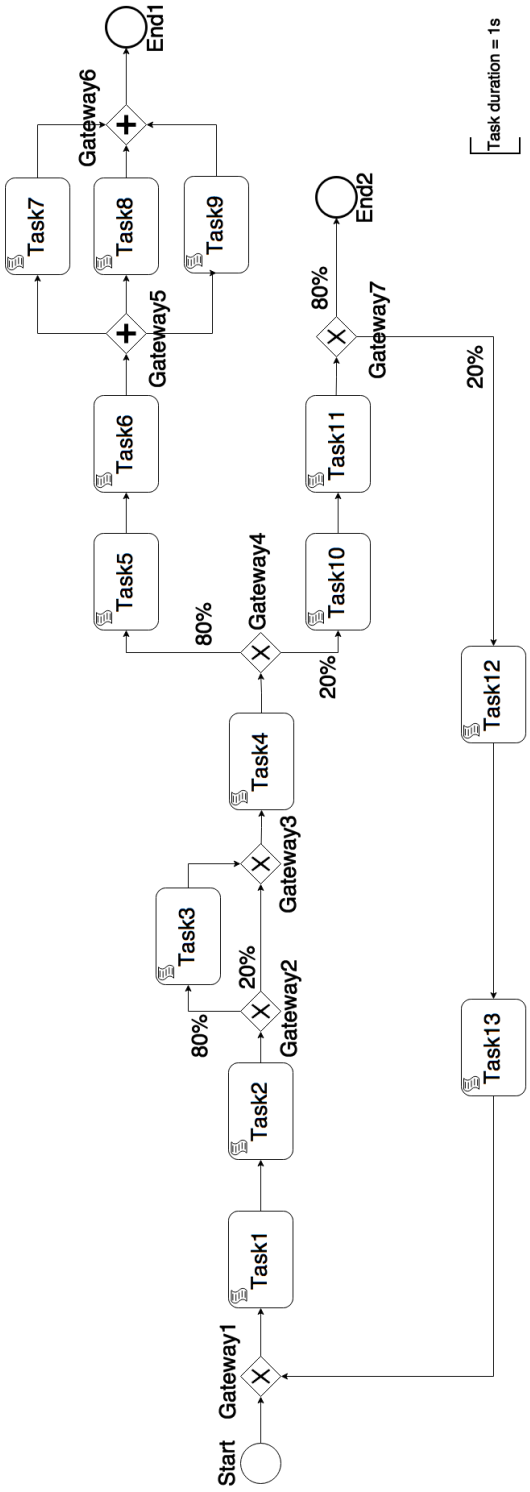


Figure 7.4. Process model 2 of cluster 4

7.3.1.2 Process models transformation

The following transformations are applied to make the process models easily executable on WfMSs without the human interaction:

- Activities are transformed into script tasks which wait 1 second before continuing the execution. Since the execution log of the process models is not available and the real task duration is not known, 1 second is arbitrarily choose as the task duration. This simplification is based on project management studies and in particular on Program Evaluation and Review Technique (PERT) [26], which considers tasks duration as probabilistic and modeled by a Beta distribution or Gaussian distribution. The distribution parameters are determined according to summary descriptors of the distribution, such as mean expected time and time variance; these summary descriptors are computed based on three main values:
  - minimum time A: it is the minimum time required to complete the task and it is an optimistic time estimation based on favorable circumstances. It is impossible that tasks are completed before the minimum time.
  - most likely time M: it is the normal time required to complete the task, which is completed frequently in this time period. In statistical term it is the median time of probability distribution.
  - pessimistic time B: it is the time required to complete the task in the worst case scenario and it considers that all possible delays and inconveniences occur.

  In practice task duration d is enclosed between minimum times A and pessimistic time B, so $A \leq d \leq B$.
- Gateway execution requires to determine which flows should be executed; in particular this problem occurs for exclusive gateways that are almost ubiquitous in the process collection. Since process models in the collection express execution probabilities of flows connected to exclusive gateways, the execution flow is determined at run time according to flow probabilities. Specifically each process has tasks which generate probabilities and the gateways compare the generated probabilities with the flow probabilities and they decide the execution flow accordingly.

According to these transformations, process models can be converted and made easily executable on different WfMSs.

7.3.2 Workload intensity

In this analysis, according to Molyneaux definition [70, pg. 55], workload intensity is based on continuous injection, because determining request arrival rate is difficult without having request logs and because request arrival rate is better applied to really specific cases. For more information read Section 2.3. Moreover once that workload intensity has been determined, using benchmark to compare arrival rates could be difficult for companies who want to use the benchmark to compare WfMSs.

Instead the enterprises have a clear idea of how many users will make requests to the system; i.e: it is possible to approximate how many concurrent users make requests to WfMS depending on the company size. In particular businesses can be divided into: small, medium and big company. Enterprise size is straightforward by definition: small enterprises have maximum 50 employees, medium enterprises have maximum 250 employees and big enterprises have more than 250 employees. Therefore, it has been decided to consider concurrent users as an important parameter to determine the workload intensity. In particular it has been decided to consider that all possible employees of an enterprise interact with WfMS, so there are tests that consider 50 users for small enterprise and 250 users for medium enterprise. Since it is undefined how many employees big companies might have at maximum, it is used the approximation of 400 and 1000 users for big companies, where medium-large enterprises have 400 users and large enterprises have 1000 users.

Moreover, the used continuous injection is stepwise, because it is more likely to happen in real cases. Indeed, concurrent user will start to interact with the system at different times. Finally, also the stopping of requests decreases according to steps, considering that users leave progressively. Especially, users interacting with the WfMS increase according to a ramp up period that is equivalent to the number of users that the test considers. This means that tests use stepwise continuous injection of users and every second, one new user is injected into the system, until the required number of users is reached. The same happens in the ramp down phase, except that every second a single user is removed from the test.

In addition, the continuous injection requires the definition of a plateau and in particular, instead of apply an arbitrary plateau, it has been determined by a test pre-run which did not applied any limit to the number of requests sent to the system. So, through the test pre-run, it has been determined the number of requests to send to the system, which is represented in Table 7.2. Activiti has been used to determine the plateau and it has been chosen at random. For example in the test pre-run the plateau of test with 50 users for cluster 3 is determined by applying the test without specifying the plateau; after that the test pre-run is launched, it is waited until the plateau is stable for approximately 10 minutes, which is arbitrarily chosen to have a stable plateau, and then the test pre-run is stopped and the stable plateau value is used for the tests with 50 users for cluster 3. The same procedure is applied for all the tests of both clusters. Furthermore it should be considered that the plateau might not be reached due to the limitations of the proof of concept (see Sections 7.1 and 7.2). A matter of fact, in general workload generator and the system under study should be located on different machines, so that there are no interference between the workload generator and the system studied, because the system studied might drain resources from workload generator and it might alter how many requests the system has received and viceversa.

Table 7.2. Tests summary

Number of test users	Ramp up duration (s)	Plateau (requests/min)		Load duration (min)	
		Cluster 3	Cluster 4	Cluster 3	Cluster 4
50	50	450	50	90	300
250	250	800	160	90	90
400	400	1100	350	90	90
1000	1000	1650	1250	90	90

7.3.3 Test characteristics

In general, each test maintains its workload for one hour and a half, which is a reasonable time between time required to execute all tests and time required to have an enough large number of process instances. Only the test with 50 users for cluster 4 is run for five hours, instead of the usual one hour and a half of other tests, to have a large number of process instances (see Table 7.2); indeed, if the test runs as the others, the number of process instances started for each process model of cluster 4 is about 2000.

Each test for 50, 250, 400 and 1000 users is executed independently, so that only one test is run at a time, and also cluster 3 and 4 are treated independently. Therefore each test executes both process models together of one single cluster at a time and only one test is executed every time. In addition the load distribution between the requests of instantiation is equally divided between the two process models of a cluster; which means that the number of instantiation requests of one process model belonging to a cluster should be equal to the number of instantiation requests of the other process model of the same cluster.

Finally, each test is executed only once, which means that the proof of concept is based on one single trial. This is reasonable in a proof of concept, however an accurate analysis of WfMSs performance should consist on more than one trial according to statistical sample size [61], because the number of trials increases the reliability of the obtained results.

7.4 Evaluated Metrics

The metrics used to determine the best WfMS are:

- completion time of processes;
- number of completed processes;
- request error percentage.

As Section 6.4.1 describes, completion time is the time required by the WfMS to completely execute process instances. In particular, in the analysis discussion for the completion time of processes the considered values are averages. Instead, the number of completed processes, as the name express, is the number of process instances that are executed and that reach the completion state.

Moreover, request error percentage, computed by the driver, is considered because it indicates how many process execution requests are rejected by the Workflow Management System. Even if it is not a performance metric, it is useful, because WfMSs might loss requests on purpose or not; in effect by rejecting requests WfMSs might increase their performance because they have a lighter load of work. In addition, to interpret the results obtained by the previous metrics analysis average request response time metric is considered.

Therefore, the metrics can be distinguished between those computed by JMeter loader and by the performance meter and data aggregator; in particular completion time of processes and number of completed processes are computed by applying the performance meter and data

aggregator, whereas request error percentage and average request response time is obtained by JMeter.

7.5 Test results discussion and analysis

According to the test results and analysis, as Table 7.3 summarises, the best performer WfMS is not always the same, because the performance depends on the cluster characteristics and the number of users that interact with the WfMS.

Table 7.3. Tests results summary

Test with		Average Completion Time (ms)		Number Completed Processes (#)		Best performed WfMS
		Camunda	Activiti	Camunda	Activiti	
Cluster 3	50 users	∼	∼		✓	Activiti
	250 users	✓		∼	∼	Camunda
	400 users	∼	∼		✓	Activiti
	1000 users	∼	∼		✓	Activiti
Cluster 4	50 users	✓		∼	∼	Camunda
	250 users	✓		∼	∼	Camunda
	400 users	∼	∼		✓	Activiti
	1000 users	∼	∼		✓	Activiti

∼: value difference is not relevant; ✓: value difference is relevant.

As a matter of fact, according to the tests results and analysis discussion of cluster 3, it is possible to state that Activiti has better performances than Camunda executing process models if the number of users interacting with WfMS are 50, 400 and 1000 users; whereas Camunda should be preferred for medium enterprises.

Indeed in the 50 users test the average completion time values of Camunda and Activiti are very close, whereas Activiti outstands Camunda in the number of completed processes due to the higher response time of Camunda. The 250 users test is the only case of the tests of cluster 3 where the average completion time difference between the two WfMSs is relevant in favour of Camunda, even if it is only 0.41% and 0.48% for process 1 and 2 respectively. In this test the number of completed processes are very close. The 400 and 1000 users tests have the same characteristic of the 50 users test: the average completion time values of the two WfMSs are very close, whereas Activiti outstands Camunda in the number of completed processes due to the higher request error percentage of Camunda.

Instead for cluster 4, according to the tests results and analysis discussion, it is possible to state that Camunda has better performances than Activiti executing cluster 4 process models if the number of users interacting with WfMS are 50 and 250 users; whereas Activiti should be preferred if the number of users interacting with WfMS are 400 and 1000 users.

As a matter of fact, in the 50 and 250 users tests Camunda on average executes processes instances faster than Activiti for relevant values, whereas the number of completed processes are

close. Instead, in 400 and 1000 users tests happen the contrary, the two WfMSs average completion times are close whereas Activiti performs better for the number of completed processes. For the 400 users tests the reason is due to the higher request error percentage of Camunda. Whereas for the 1000 users tests it is due to the fact that Activiti has a low response time, because request of execution are rejected before their instantiation or anyway before that Camunda rejects them. For 1000 users test this explains also why the number of completed processes delta between the WfMSs is much higher than the request error percentage delta.

For the request errors they happen more often when the number of users interacting with the WfMS is higher because a greater number of users generate a higher number of requests and the WfE could not be able to manage them all, especially using a single machine (see Section 7.2).

As a matter of fact for the 50 and 250 users tests of cluster 3 the request error percentage is zero and for the same tests for the cluster 4 only Camunda has request errors, but they are 0.05% and 0.04% respectively for the two tests and those request errors are caused by the loop structure of cluster 4 process models. For both clusters in the 400 and 1000 users tests, the request error percentage is relevant. In particular the main Activiti errors consist of socket exceptions and internal servers errors; the first type of error is caused by connection reset by Activiti or by socket closure without communication of its closure to JMeter; whereas the latter type of error is caused by connection establishment between the WfE and the WfMS database. Instead the main Camunda errors consist of socket exceptions, connection exception and internal server errors. Socket exceptions are caused by connection reset and broken pipe, where the first one indicates that the WfE opens the connection, but it communicates that the connection will not be listened, whereas in the second error type the connection is closed while it is used; connection exception is caused by timeout triggered on JMeter. Finally, internal server errors occur only in cluster 4 and they are caused by the high number of loop iterations which uses the whole RAM memory available for the process instances execution and causes a stack overflow.

7.5.1 Cluster 3

Cluster 3 is composed by two process models, which can follow a single execution path without loops. Moreover the process models belonging to cluster 3 are shown on Figures 7.1, 7.2 and their characteristics are described in the Table 7.1. For cluster 3 the test results are shown on tables 7.4, 7.5 and 7.6 and discussed in details for tests with 50, 250, 400 and 1000 users.

7.5.1.1 Test with 50 users

As the Table 7.4 shows, the completion time average values of Activiti and Camunda are very close for every process of this cluster. In particular process 1 and process 2 have a delta of only 0.10% and 0.08%; where the average completion time values for both process models is lower for Activiti.

Number of completed processes metric values show a predominance of Activiti. See Table 7.5. Activiti has executed about 6% process instances more than Camunda for each process model of the cluster (6.48% and 6.51%). This result is motivated by Activiti ability to transmit responses

Table 7.4. Cluster 3 metrics: completion time

Cluster 3					
Test with	Process model	Average completion time (ms)			
		Activiti (A)	Camunda (C)	$\Delta(A-C)$	$\Delta(A-C)$ %
50 users	processC3P1	7230.86	7237.80	**-6.94**	-0.10%
	processC3P2	5603.90	5608.64	**-4.74**	-0.08%
250 users	processC3P1	7262.37	7232.41	**29.95**	0.41%
	processC3P2	5630.96	5603.86	**27.10**	0.48%
400 users	processC3P1	7233.24	7231.95	**1.29**	0.02%
	processC3P2	5601.31	5603.12	**-1.81**	-0.03%
1000 users	processC3P1	7235.27	7234.74	**0.53**	0.01%
	processC3P2	5604.14	5604.88	**-0.74**	-0.01%

Table 7.5. Cluster 3 metrics: number completed processes

Cluster 3					
Test with	Process model	Number completed processes (#)			
		Activiti (A)	Camunda (C)	$\Delta(A-C)$	$\Delta(A-C)$ %
50 users	processC3P1	20115	18811	**1304**	6.48%
	processC3P2	20087	18779	**1308**	6.51%
250 users	processC3P1	38178	37075	**1103**	2.89%
	processC3P2	38058	36923	**1135**	2.98%
400 users	processC3P1	54692	46268	**8424**	15.40%
	processC3P2	54423	47324	**7099**	13.04%
1000 users	processC3P1	73969	42895	**31074**	42.01%
	processC3P2	73523	43810	**29713**	40.41%

Table 7.6. Cluster 3 metrics: request error percentage and response time

Cluster 3					
Test with	Process model	% Request error		Average response time (ms)	
		Activiti	Camunda	Activiti	Camunda
50 users	processC3P1	0.00%	0.00%	7590	7810
	processC3P2	0.00%	0.00%	5933	6162
250 users	processC3P1	0.00%	0.00%	7471	7510
	processC3P2	0.00%	0.00%	5826	5861
400 users	processC3P1	1.30%	14.20%	7894	17429
	processC3P2	1.32%	11.88%	6274	16394
1000 users	processC3P1	25.30%	57.64%	7201	14456
	processC3P2	25.39%	56.51%	5994	13718

in lower time than Camunda, indeed Activiti response time is lower than Camunda response time.

Finally the request error percentage is zero for both Workflow Engines, so it can be neglected in the decision of which is the best WfMS.

In conclusion, according to only completion time values is not possible to prefer one or the other Workflow Management System because the delta are too narrow. However, Activiti has managed the execution of more process instances than Camunda for all process models of cluster 3. Therefore, it is possible to state that Activiti should be used by small enterprises that have process models and workload similar to those expressed in this test.

7.5.1.2 Test with 250 users

The completion time values show that all processes of cluster 3 are executed faster on Camunda, see Table 7.4. Camunda executes process 1 and 2 faster than Activiti by 0.41% and 0.48%.

Whereas the number of completed processes values of Camunda are lower than Activiti by 2.89% and by 2.98% (see Table 7.5), indeed the response time is slightly better for Activiti.

Finally the request error percentage is zero for both Workflow Management Systems, so it can be neglected in the decision of which is the best WfMS.

Considering that the delta between the number of completed processes values is close and that the completion time metric values are more relevant, it is possible to state that Camunda should be used by medium enterprises that have process models and workload similar to those expressed in this test.

7.5.1.3 Test with 400 users

As Table 7.4 displays, the completion time values of the WfMSs are very close for both the process models. Indeed, the difference for process model 1 is 0.02%, where Camunda is faster; and the difference for process model 2 is 0.03%, where Activiti is faster.

Instead, for the number of completed processes values Activiti outstands Camunda. For process 1 and 2, it executes more process instances by 15.40% and 13.04% than Camunda due to the better response time and lower request errors of Activiti.

As a matter of fact also the request error percentage is lower in Activiti, it is 1.30% and 1.32% in Activiti and 14.20% and 11.88% in Camunda. Activiti errors are caused by Internal server errors due to high number of connections with the database; this is a limitation of database manager used by default by Activiti. Instead, Camunda errors are caused by:

- Connection closure after timeout raised and it occurs only 0.63% of times on the total amount of errors;
- Socket exception
  - due to closure of the connection by the WfMS during data communication (26.48%);

  - due to Connection reset by Camunda; in practice the Workflow Engine opens the connection, but it communicates that the connection will not be listened. It is the main error with almost 73.89% occurrences.

In conclusion, given that the completion time value are not distinguishing, there is to consider the number of completed processes values to determinate which is the better WfMS for cluster 3, 400 user test. Thus, considering that Activiti has managed the execution of more process instances than Camunda and it has lower request errors, it is the best Workflow Management System for process models and workload similar to those expressed in this test when there are 400 users that interact with the WfMS.

7.5.1.4 Test with 1000 users

For this test the completion time metric cannot determinate which is the best WfMS because for process 1 is faster Camunda by 0.01% and for process 2 is faster Activiti by 0.01%.

Instead, considering the number of completed processes metric, Activiti is able to execute more process instances than Camunda; it executes 42.01% and 40.41% process instances more than Camunda respectively for process models 1 and 2. This occurs because the response time and request error percentage of Camunda is higher than response time and request error percentage of Activiti.

The request error percentages for the test with 1000 users are different from zero for both WfMSs. Its values for Activiti are 25.30% for process 1 and 25.39% for process 2. The values for Camunda are 57.64% and 56.51% respectively for process 1 and process 2, thus Activiti outstands Camunda.

Camunda errors are of two types:

- Connection exception, in particular it is raised by timeout on JMeter and it causes more than 4% of the total errors;
- Socket exception:
  - due to Broken pipe; with approximately 2.75% occurrences;
  - due to connection closed by Camunda after that the WfMS has send a communication of connection closure. It is the main error with about 93% occurrences.

Instead Activiti errors types are:

- Socket exception raised by socket closed by Activiti providing a communication to JMeter has 10.55% occurrences, whereas Socket exception raised by socket closed by the WfMS without providing a communication to JMeter occurs 1% of the total errors;
- Internal server errors with about the 88.50% occurrences; they are caused by the high number of connections with the database.

Since the request error percentage and the number of completed processes metrics show that Activiti is better than Camunda to execute process models and the completion time values are not distinguishing, it is possible to conclude that Activiti is the best Workflow Management

System for large enterprises that have processes and workload similar to those expressed in this test.

7.5.1.5 Cluster 3 conclusions

Considering the tests results and analysis of cluster 3, it is possible to state that Activiti has better performances than Camunda executing cluster 3 process models if enterprises considered are small, medium-large or large; instead Camunda should be preferred for medium enterprises.

7.5.2 Cluster 4

Cluster 4 is composed by two process models, which have a complex structure that includes loops. Moreover the process models belonging to cluster 4 are shown on Figures 7.3, 7.4 and their characteristics are described in the Table 7.1. For cluster 4 the test results are shown on Tables 7.7, 7.8 and 7.9 and discussed in details for tests with 50, 250, 400 and 1000 users.

Table 7.7. Cluster 4 metrics: completion time

Cluster 4					
Test with	Process model	Average completion time (ms)			
		Activiti (A)	Camunda (C)	$\Delta(A-C)$	$\Delta(A-C)$ %
50 users	processC4P1	31931.27	30869.69	**1061.59**	3.32%
	processC4P2	9694.89	9567.30	**127.59**	1.32%
250 users	processC4P1	31876.77	31407.91	**468.86**	1.47%
	processC4P2	9707.23	9648.15	**59.08**	0.61%
400 users	processC4P1	31129.78	31023.96	**105.82**	0.34%
	processC4P2	9687.33	9613.55	**73.78**	0.76%
1000 users	processC4P1	31370.19	31205.94	**164.25**	0.52%
	processC4P2	9714.87	9643.47	**71.40**	0.73%

Table 7.8. Cluster 4 metrics: number completed processes

Cluster 4					
Test with	Process model	Number completed processes (#)			
		Activiti (A)	Camunda (C)	$\Delta(A-C)$	$\Delta(A-C)$ %
50 users	processC4P1	6327	6295	**32**	0.51%
	processC4P2	6289	6267	**22**	0.35%
250 users	processC4P1	7803	7482	**321**	4.11%
	processC4P2	7619	7283	**336**	4.41%
400 users	processC4P1	17334	14806	**2528**	14.58%
	processC4P2	17076	14998	**2078**	12.17%
1000 users	processC4P1	24580	17763	**6817**	27.73%
	processC4P2	27798	17493	**10305**	37.07%

Table 7.9. Cluster 4 metrics: request error percentage and response time

Cluster 4					
Test with	Process model	% Request error		Average response time (ms)	
		Activiti	Camunda	Activiti	Camunda
50 users	processC4P1	0.00%	0.05%	32446	32808
	processC4P2	0.00%	0.00%	10060	11287
250 users	processC4P1	0.00%	0.04%	32262	32333
	processC4P2	0.00%	0.00%	9956	10306
400 users	processC4P1	0.23%	7.56%	31544	65921
	processC4P2	0.18%	4.84%	10000	47175
1000 users	processC4P1	72.03%	71.16%	9389	49173
	processC4P2	68.16%	71.32%	3532	42983

7.5.2.1 Test with 50 users

The test with 50 users for cluster 4 is run for 5 hours, instead of the usual one hour and a half of the other tests, because extra time is required to have a large number of process instances. Indeed, if the test runs as the others, the number of process instances started for each process model of cluster 4 is about 2000.

Evaluating the completion time values of the test with 50 users of Activiti and Camunda, it is possible to state that Camunda executes process instances faster than Activiti. In particular Camunda completes 3.32% faster than Activiti the execution of process instances of process model 1 and 1.32% faster than Activiti the execution of process instances of process model 2. Thus Camunda performs better than Activiti considering the completion time metric.

The number of completed processes metric, in this test, has values very similar for both Activiti and Camunda. Indeed, the difference of process instances executed is lower than 1% for all process models; in particular, Activiti executes more process instances than Camunda, but the delta has little relevance (0.51% and 0.35%).

Furthermore the request error percentage is zero for Activiti, while Camunda has a 0.05% value for this metric on process model 1. This means that three requests are not satisfied over the total number of requests. Since the value is not significant it can be neglected in the decision of which is the best WfMS.

Anyway, by analyzing the stack trace of these errors, it is possible to determine that they derive from internal server errors. Therefore, it is possible that process instances started might have been suspended or cancelled; thus a further analysis is required. This analysis shows that there are process instances, which have not been completed; specifically they have been suspended during the execution. An additional information obtained by the stack trace is that the internal server errors are caused by stack overflow. This is a common error that occurs during recursions and both process models of cluster 4 have loops. In fact these errors are caused by the two loops that occur in the process model 1; indeed during the test, the suspended process instances of process model 1 take the loops more than 18 times. This problem might be partially overcome by adding additional RAM memory that Camunda can use for the process execution, otherwise it is required to modify the process model structure. In particular additional RAM memory

for Camunda execution is a partial solution because every time that a loop is iterated, the RAM memory used for the execution increases due to the execution of new constructs that are written in memory; when the available RAM memory is finished, stack overflow occurs.

In conclusion Camunda performs better than Activiti and it should be chosen as Workflow Management System from small enterprises that have process models and workload similar to those expressed in this test. However, enterprises must be aware of the limitation in the number of loops that it can enforce. Therefore, they should design process models according to it, or apply one of the solution proposed. Alternatively, they could choose Activiti, but they have to consider that they will have lower performance than by using Camunda.

7.5.2.2 Test with 250 users

The test with 250 users shows similarities with the test with 50 users. Indeed the average completion time values of the test with 250 users display that Camunda executes process instances faster than Activiti (see Table 7.7). The highest completion time delta corresponds to the process model 1 executions, where Camunda completes process instances 1.47% faster than Activiti, whereas the delta completion time of process instances of process model 2 is 0.61%.

In this test, the number of completed processes metric shows similar values, where Activiti executes 4.11% and 4.41% more process instances than Camunda. This difference is due to the fact that Activiti is able to have lower response time, thus in the number of completed processes metric shows a preference for Activiti.

Furthermore the request error percentage is zero for Actitivi. Instead Camunda request error percentage is 0.04% in process 1. Anyway the value is not significant and it can be neglected in the decision of which is the best WfMS as the number of completed processes metric.

As for the previous test, all Camunda errors are derived by internal server errors. By analysing stack trace of these errors, it is possible to determine that errors occur during the execution of tasks and they are derived from process instances that have not been completed. As in the test with 50 users the error is caused by loops executed more than 18 times.

In conclusion medium enterprises, which have the majority of process models and workload similar to those expressed in this test, should prefer Camunda if they know that the loops of their process models do not executed indefinitely. They can apply a partial solution to the problem such as increase the RAM memory available to the JVM, to reach the maximum number of loops required. Alternatively they could choose Activiti, considering that they will have lower performance than by using Camunda, according to the average completion time metric values.

7.5.2.3 Test with 400 users

As Table 7.7 exhibits, the average completion time values of Camunda are slightly better than Activiti; indeed it is 0.34% and 0.76% faster than Activiti in executing process instance of process 1 and 2 respectively. Anyway process average completion time shows that its values

difference between Activiti and Camunda is decreasing comparing it with 50 and 250 users tests.

The number of completed processes metric states the predominance of Activiti over Camunda for more than 10%. In fact, Activiti executes 14.58% more of process instance for process 1 and 12.17% more process instances for process 2 than Camunda. This occurs because the response time of Camunda is higher than the response time of Activiti.

Also for the request error percentage Activiti is better than Camunda, its errors are 0.23% and 0.18%, whereas for Camunda they are 7.56% and 4.84%. Activiti errors are all caused by Activiti limitation in managing connections with database by its default manager. Instead Camunda errors are:

- socket exception, which is caused by connection reset for almost 34% of total errors and by broken pipe for more than 58% of total errors;
- connection exception, which is raised by timeout on JMeter and it occurs 7.60%;
- internal servers error due to stack overflow with about 0.30% occurrences.

Camunda and Activiti performances for cluster 4, 400 users test are very close. Camunda is better according to completion time metrics, whereas Activity has higher number of completed processes values and better request error percentage. Thus, according to the high deltas of the number of completed processes values, the considered best WfMS is Activiti for medium-large enterprises, which have process models and workload similar to those expressed in this test.

7.5.2.4 Test with 1000 users

For cluster 4 all the tests with different number of users show a similar outline about completion time metric, where completion time metric values are lower for Camunda, so Camunda is faster than Activiti for all processes of cluster 4. Anyway only for 50 and 250 users tests the completion time difference between the two WfMSs is relevant.

In particular, the test with 1000 users shows that Camunda is only slightly better than Activiti, which requires 0.52% and 0.73% more time to execute and to complete process instances of the process models.

In addition, Activiti executes 27.73% more process instances of process models 1 than Camunda and, for the process model 2, Activiti executes 37.07% more process instances than Camunda. Thus, it is possible to determine that according to number of completed processes values Activiti is able to manage more processes instances. The number of completed processes values of the two WfMSs are mainly influenced by the request errors and the response time of the WfMSs. As a matter of fact the response time value of Activiti is much lower than the value of Camunda. This might be explained by the fact that Camunda requests reach timeout, as the following request error analysis shows. Whereas Activiti has a low response time, because requests of execution are rejected before their instantiation or anyway before that Camunda rejects them. This explains the reason why the delta between the WfMSs of the number of completed processes metric is much higher than the request error percentage delta.

Request error percentage is high for both Activiti and Camunda. Camunda has about 71% of request error percentage for all process models of cluster 4, whereas Activiti requests of execution of process models 1 and 2 are affected by 72.03% and 68.16% of request error percentage respectively. Therefore Activiti and Camunda have the same level of errors.

Camunda main errors consist of:

- Socket exception due to connection closure by Camunda after a communication of connection closure consisting for the 40.72% of total errors. In addition, also socket exception due to broken pipe error is present with 0.71% occurrences.
- Connection exception; in particular it is raised by timeout on JMeter and it occurs 58.56% of total errors.

Thus Camunda is unable to manage the connection and after that the connection is established, Camunda closes it before that communication is finished. Another problem is that when Camunda does not serve a request, it does not send keep alive messages to JMeter, so the connection must be aborted by JMeter after that JMeter timeout timer triggers. In particular, the Workflow Engine is using almost all the RAM memory available, so probably Camunda is not able to manage additional connections and it needs to close or reject them. In addition, there are few internal server errors that are caused by process instances that exceeded the stack available to Camunda. As previously discussed, it occurs when loops of process instances are executed multiple times.

Activiti errors are:

- Socket exception raised by socket closed;
- Internal server errors; they provide additional information than other errors, because they provide a stack trace of error causes.

The majority of errors are internal server errors (99.98%) and only 26 errors correspond to Socket exception over more than one hundred thousand total errors. A short analysis of the socket exception shows that its errors appear only for the final requests, so they might be caused by the stopping of JMeter load. Therefore it can be considered as a negligible type of error and it will not occur during application production. Instead, the internal server errors are caused by the high number of connections with the database, which Activiti requires to open. This means that for the process models of this test, Activiti requires to open more connections than it requires with process models of other tests.

However this requirement to open a high number of connection hits with the limitation imposed by the Workflow Engine by default. In particular by default Activiti exploits MyBatis [5] framework to interact with the database and to manage the connection pooling. MyBatis is an Object-relation Mapping framework that maps Java methods to SQL statements. This framework limits Activiti and according to Activiti team, users do not care about the default connection pool manager, because they prefer to change the connection pool manager with their own preferred one when they use the WfMS in production [1]. Finally, since internal server error might be caused also after process instantiation, a further analysis is performed to ensure that connection errors with the database do no occurs after that process execution is started. According to this last analysis there is no process instance suspended or cancelled.

In conclusion, even if Camunda has lower completion time values than Activiti, Activiti is able

to better manage process instances; indeed Activiti has high number of completed processes metric values (24580 and 27798). Therefore, if a large company has to choose between the two Workflow Management Systems and it has process models and workload similar to those expressed in this test, it should choose Activiti.

7.5.2.5 Cluster 4 conclusions

Considering the tests results of cluster 4, it is possible to state that Camunda has better performances than Activiti executing cluster 4 process models if enterprises considered are small or medium. However, if enterprise dimension grows, it should be preferred Activiti.

7.6 Limitations and conclusion

The tests and analyses made as proof of concept have all limitations and simplification that a proof of concept can have. In particular, only the two tools developed are used and not the entire BenchFlow framework.

Moreover, the tests are run on a single low-power machine instead of using a well controlled Cloud infrastructure, so the test results are heavily influenced by the infrastructure, especially for the load tests with 1000 users, and they are derived from only one trial of the experiment. In addition, the process models transformations reduce the used constructs and they remove interactions so they are only intended to provide process models usable in the proof of concept. Furthermore, the analysis are performed using the metrics of the performance meter and data aggregator and of the driver, so a limit is the unavailability of the constructs metrics. Finally, the proof of concept limitations are the ones of the two implemented tools, so for additional information about the data cleaner and reconciler and performance meter and data aggregation limitations, read Section 8.2.

Even considering the proof of concept limitations, it is possible to conclude that those types of analyses are feasible and that the best performer WfMS depends on the process models characteristics and the enterprise dimension.

Chapter 8

Conclusion

8.1 Summary and conclusion

As part of the BenchFlow project [13], this thesis focuses on the design, implementation and evaluation of the data cleaner and reconciler and the performance meter and data aggregator components of the BenchFlow framework, to allow the analysis and management of the data from different WfMSs and to be able to examine the data about the performance of Workflow Management Systems. To achieve these goals:

- the data cleaner and reconciler is realized; it uses a common destination database, which contains standardized data generated by applying the ETL process from source databases of different Workflow Management Systems. The design of the destination database is based on a collection of basic requirements according to metrics and context knowledge [99].
- the performance meter and data aggregator is realized; it aggregates data from the data cleaner and reconciler database to obtain information. Data aggregation is performed according to metrics application based on the experiments and on the trials related to a specific experiment.

Both the data cleaner and reconciler and performance meter and data aggregator are developed in Java. Since both components have to interact with the CleanRawData database, Java Object Oriented Querying (jOOQ) framework is used as Object-Oriented Query Language (OQL) technique. Indeed, this technique allows reducing code errors and it facilitates software development. In addition, the performance meter and data aggregator has to interact also with the non-relational MongoDB database, because it needs to have a schema that can be easily extended by adding additional fields, which will contain new metrics values. For this interaction the default MongoDB driver is used.

Thus, the thesis starts with an excursus of the BPMN 2.0 standard, the Workflow Management System architecture and benchmark frameworks. In addition, the BenchFlow framework is presented and the driver, collector, monitor, data cleaner and reconciler and performance meter and data aggregator components are depicted. Then, the SMDM methodology is applied to de-

fine the main WfMSs metrics among which there are the ones implemented in the performance meter and data aggregator: processes completion time metric, number of completed processes metric and number of uncompleted processes metric. Furthermore, the design and implementation of the data cleaner and reconciler and the performance meter and data aggregator are described and their performances are evaluated.

Finally, a proof of concept is provided to demonstrate the feasibility of analysis through the application of the two tools developed and to apply the implemented metrics on two different WfMSs. The performances of the WfMSs are the focus of the BenchFlow framework, for this reason in this thesis the proof of concept applies performance tests and specifically load tests are used. As proof of concept, the tests are performed on a single low-power machine and only one trial of the experiment is obtained for each load test ran. The load tests are based on the number of users interacting with WfMS and they consider: 50 users for small enterprises, 250 users for medium enterprises, 400 users for medium-large enterprises and 1000 users for large enterprises. The workload of the tests are based on two different clusters: cluster 3 and cluster 4, each one contains two process models, for a total of four process models. According to the analysis of the proof of concept, based on the implemented metrics, the best performer WfMS is not always the same, because WfMSs performance depend on the number of users that interact with the WfMS and the cluster characteristics.

Thus, the main contributions of this thesis to the research community are the creation of two tools belonging to the BenchFlow framework [13], one that standardizes data from all the source databases of different WfMSs and one that aggregates stored data to obtain information about WfMS BPMN 2.0 metrics; moreover, first results of the feasibility of performance testing and benchmarking of WfMSs are provided in a proof of concept. The data cleaner and reconciler tool can be further extended to standardize additional WfMS data and the performance meter and data aggregator tool can be further expanded and customised arranging stored data to obtain additional WfMS metrics.

8.2 Current limitations and future work

About the implemented components, a current limitation is the requirement to use the command line to interact. Indeed, components that use a command line interface make automated interactions difficult between them, so a future work is to introduce an interaction paradigm based on socket or a REST API interface. Additionally, new WfMSs should be added to the data cleaner and reconciler and additional metrics should be implemented in the performance meter and data aggregator; in particular the restricted number of implemented metrics is a limitation of the tool and of its evaluation. Another current limitation might be the storage on the databases of the two implemented components; due to the fact that data could grow infinitely. Hence, a future work is to analyze the possibility to partition the MySQL database, used for the data cleaner and reconciler, and to shard the MongoDB database, used by the performance meter and data aggregator.

In addition, a limitation of the data cleaner and reconciler is that it processes only environmental data derived from Docker stats API, since the BenchFlow infrastructure exploits Docker containers. Moreover, the jOOQ framework only supports SQL compliant databases, thus this is a limitation to the databases that the data cleaner and reconciler can support. About the

evaluation of the data cleaner and reconciler, it is limited by the application of a process model with 3 constructs; furthermore, from the evaluation, the data cleaner and reconciler available RAM memory is identified as a bottleneck of the tool.

Moreover, the performance meter and data aggregator limitation is imposed by MongoDB performances compared to other NoSQL databases, especially Cassandra. For this reason, a future work could be to analyze and, in case to implement, the performance meter and data aggregator using a more performant database such as Cassandra [9]. In addition, also reading the data from the CleanRawData database is an expensive operation and a future work could be to discuss the possibility to apply a data caching system. As a matter of fact its application would allow the performance meter and data aggregator to have faster access to data. Furthermore, the performance meter and data aggregator does not support automatic discovery of Metric subclasses and a future work is to analyze and to implement a service locator in the MetricCalculator class.

Last but not least, a future work will consist in the evaluation of the WfMS metrics through their application on different WfMSs by taking advantage of the BenchFlow framework on a real testbed. This will allow to determine which parameters are the most important in the evaluation of WfMSs and it will allow companies to take an informed decision about which Workflow Management System should be used for their business.

Appendix A

Metric theoretical validation

The theoretical validation is conducted applying the SMART framework [32]. See Chapter 4 for more details about the metrics.

The metrics defined for the Workflow Engine entity are divided according to:

- time behaviour in:
  - response time

Metric name	Response time
Specificity	It is useful to determine the behaviour of the WfE regarding how the WfE will respond to requests. So the metric is goal oriented. In any case workers, who do not have this metric clear definition in mind, might misinterpret its meaning with other; for example they might interpret it as residence time [63], which is only the time which the request is into the system, so the time to execute the process. Anyway these two measures are strictly related, therefore the impact of such misinterpretation is reduced. Moreover the definition of this metric has been clearly defined in previous steps (see Section 4.1.2) and this problem is less probable.
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.

Attainability	It can be measured by means of the load driver during benchmarking. Usually workload loaders store into logs files data related to response time; in some case they could store response time directly. Indeed for real world request it might not be easily computed due to lack in the system, anyway this metric might be stored into log file of the systems. Finally it depends also on how workload loader and WfMS interacts, so it depends on how request are sent to Workflow Management System and responses are received.
Relevance	It is relevant, because it represent the time experienced by the user that uses the WfMS.
Timeliness	The time required to compute response time depends on the time required by the Workflow Engine to respond to a request. Therefore it depends on the communication time and the execution time of the process instance requested.

Thus, the response time metric respects all five SMART characteristics and even if it is difficult to obtain data to compute the metric, it is not impossible.

– throughput

Metric name	Throughput
Specificity	It is useful to determine the behaviour of the WfE according to time requirements. So the metric is goal oriented. Moreover workers, usually, have this metric clear definition in mind of throughput and by specifying the correct unit of measurement it is hard to misinterpret it.
Measurability	It is a single value expressed in rates and it can be easily compared to other throughput rates. In particular it is cardinal variable.
Attainability	It can be measured by means of the load driver during benchmarking. Usually workload loaders store into logs files data related to request completed and their time interval; in some case they could store throughput directly. Indeed for real world request it might not be easily computed due to lack in the system. In some case data related to this metric might be stored into log file of the systems. In addition this metric can be computed from stored data by the WfMS.

Relevance	It is relevant, because it represent how many request of the user can be performed by the WfMS. Anyway it might be misleading, due to the intrinsic time required by the process instance to be executed. Therefore a good workload must be applied to have relevant throughput measurement.
Timeliness	The time required to compute throughput depends on how many and how fast the WfE receive requests and execute them. So if the Workflow Engine receives enough requests all at once, the results depends only on the time required by the WfE to execute the requests.

Thus, the throughput time metric respects all five SMART characteristics.

– process execution latency

Metric name	Latency
Specificity	It is useful to determine the behaviour of the WfE according to time requirements and in particular determine if the WfE is managing efficiently time. So the metric is goal oriented. However workers do not have a clear definition in mind and might misinterpret its meaning with other. Anyway the definition of this metric has been clearly defined in previous steps (see Section 4.1.2).
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.
Attainability	It is very difficult to measure it, because it requires correlating data between load driver or the system and Workflow Engine per each request. Usually workload loaders store into logs files data related to request query, but in this data it is necessary to have a correlation key that relates the data of the loader with data of the WfE and it is improbable to have a correlation key directly. Moreover in real world request it might be even harder due to lack in the system logging.
Relevance	It is relevant, because it represent the time inefficiencies of the WfMS. However it is considered as a part of response time metric.

Timeliness	The time required to compute latency depends on the time required by the Workflow Engine to start the execution of a process instance and, if contemplated by the WfE, to interact with the requestor to transmit related information. Therefore it does not strictly require the complete process instance execution, but this depends on the WfE.

Thus, the process execution latency metric respects all five SMART characteristics and even if it is very difficult to obtain data to compute the metric, it is not impossible.

- resource utilization in:
  - database bandwidth usage

Metric name	DB bandwidth usage
Specificity	It is useful to determine external resources usage by the WfE. So the metric is goal oriented. In any case workers might easily misinterpret this metric meaning with other and they might not understand what it actually measures. This problem is important respect to previous, because it might lead to misinterpret the whole metric. Indeed workers might prefer a WfE with lower or higher values according to the point of view that they attribute to the metric. For example two points of view for the metric are: use less communication with the DB as possible to spare bandwidth and store more data persistently as possible, which means increasing communication, to improve reliability.
Measurability	It is a single value expressed in unit of measurement for information size, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It is difficult to measure in real world particularly. Usually databases do not store information on how many data are transmitted and received. Therefore it might be necessary to add network traffic sniffer, that add overhead to the Workflow Engine performance. Moreover it is unreliable, because it might best metric might depends on metric point of view and it might be influenced by hidden variable such as the design of database schemas.
Relevance	It might be relevant in case of restricted network bandwidth between WfMS and DB. In other cases its relevance might be reduced by hidden variables.

Timeliness	The time required to compute DB usage depends on completion of the process instances in the Workflow Engine and how much time after process completion the WfE continues to store data.

Thus, the database bandwidth usage metric respects all five SMART characteristics and even if it is difficult to obtain reliable data to compute the metric, it is not impossible.

– Network usage

Metric name	Network usage
Specificity	It is useful to determine external resources usage by the WfE. So the metric is goal oriented.
Measurability	It is a single value expressed in unit of measurement for information size, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It is difficult to measure in real world particularly. Usually WfE does not store information on how many data are transmitted and received. Therefore it might be necessary to add network traffic sniffer, that add overhead to the Workflow Engine performance. In real world application this might be even more difficult.
Relevance	It might be relevant in case of restricted network bandwidth.
Timeliness	The time required to compute Network usage depends on the interaction of the Workflow Engine with external resources.

Thus, the network usage metric respects all five SMART characteristics and even if it is difficult to obtain data to compute the metric, it is not impossible.

– RAM memory usage

Metric name	RAM memory usage
Specificity	It is useful to determine resources usage by the WfE. So the metric is goal oriented.
Measurability	It is a single value expressed in information size unit of measurement, therefore it can be easily compared to other values. In particular it is cardinal variable.

Attainability	It is difficult to measure the real RAM memory usage due to RAM memory management in systems and due to possible support services used by WfE. Moreover it should be evaluated together with additional information provided by the WfE, such as actual WfE execution load. This might introduce synchronization problem between data.
Relevance	It might be relevant in case of limited RAM memory availability and if the resource is shared with other applications.
Timeliness	It is instantaneously ready by using monitoring tool. Anyway to have realistic metrics might be important to evaluate data together with additional information provided by the WfE, such as actual WfE execution load.

Thus, the RAM memory usage metric respects all five SMART characteristics and even if it is difficult to obtain reliable data to compute the metric, it is not impossible.

- CPU usage

Metric name	CPU usage
Specificity	It is useful to determine resources usage by the WfE. So the metric is goal oriented.
Measurability	It is a percentage, therefore it can be compared to other values.
Attainability	It is difficult to measure the real CPU usage due to overhead introduced by monitoring. Anyway it should be evaluated together with additional information provided by the WfE, such as actual WfE execution load. This might introduce synchronization problem between data.
Relevance	It might be relevant in case of limited processing capacity or if the resource is shared with other applications
Timeliness	It is instantaneously ready by using monitoring tool. Anyway to have realistic metrics might be important to evaluate data together with additional information provided by the WfE, such as actual WfE execution load.

Thus, the CPU usage metric respects all five SMART characteristics and even if it is difficult to obtain reliable data to compute the metric, it is not impossible.

- capacity in:
  - capability

Metric name	Capability
Specificity	It is useful to determine the behaviour of the WfE to manage workload. So the metric is goal oriented. Moreover workers, usually, have at least an approximate idea of capability definition and by specifying the correct unit of measurement it is should not be misinterpreted.
Measurability	It is a value expressed by a quantitative measurement unit, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It can be measured by exploiting stored data by WfE or by load driver during benchmarking or the system. Using stored data by WfE should be more accurate and precise, because this metric is related to the process instances executions and WfE data might contains additional informations. Anyway determine which is the maximum number of process that the WfMS can handle require specific testing.
Relevance	It is relevant, because it represents the ability of the WfMS to process execution request and it is important to know WfE limitation. Anyway this metric might be restricted by resource limitations, but this problem is limited by running benchmarks with the same configuration for all engine under analysis. Therefore the comparison between benchmark results should be applied only for engine that have been runned under the same configuration.
Timeliness	The time required to compute capability depends on the time required by the Workflow Engine to execute process instances. Therefore it requires the completion of process instances execution.

Thus, the capability metric respects all five SMART characteristics and even if it is difficult to obtain data to compute the metric, it is not impossible.

– number of completed processes

Metric name	NumCompletedProcesses
Specificity	It is useful to determine the behaviour of the WfE to manage workload. So the metric is goal oriented. Moreover workers, usually, have an approximate idea of the definition and by specifying the correct unit of measurement it is should not be misinterpreted.

Measurability	It is a value expressed by a quantitative measurement unit, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It can be measured by exploiting stored data by WfE or by load driver during benchmarking or the system. Using stored data by WfE should be more accurate and precise, because this metric is related to the process instances executions and WfE data might contains additional informations.
Relevance	It is relevant, because it represents the ability of the WfMS to process execution request.
Timeliness	The time required to compute the number of completed processes depends on the time required by the Workflow Engine to execute process instances. Therefore it requires the completion of process instances execution.

Thus, the number of completed processes metric does respect all five SMART characteristics.

The metrics that can be defined for the Process entity for the time behaviour quality attribute are:

- completion time

Metric name	Completion time
Specificity	It is useful to determine the time behaviour of the WfE and how process instance are executed by WfE. So the metric is goal oriented. Moreover its name is self-explanatory and workers should not have difficulties to understand its meaning.
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.
Attainability	It can be measured by analysing WfE data. By default WfE should stores into logs files or in databases data related to process instance start time and end time; in some case they could store directly the time required to complete a process instance. If WfE does not store this information by default, it is possible to enable history option to store these data.
Relevance	It is relevant, because it represents the minimum time to fulfil a request by the user that request executions to the WfMS. It is the minimum time, because it does not consider overhead such as request communication overhead.

Timeliness	The time required to compute completion time depends on the time that the Workflow Engine needs to completely execute process instance and after the time required to store those data.

Thus, the completion time metric does respect all five SMART characteristics.

Finally the metrics that can be defined for the Construct entity can be divided according to:

- time behaviour in:
  - completion time

Metric name	Completion time
Specificity	It is useful to determine construct time behaviour and how WfE manages the construct. So the metric is goal oriented. Moreover its name is self-explanatory and workers should not have difficulties to understand its meaning.
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.
Attainability	It can be measured by analysing WfE data. By default WfE should stores into logs files or in databases data related to construct start time and end time; in some case they could store directly the time required to complete a construct. If WfE does not store this information by default, it is possible to enable history option to store these data. Anyway it is a volatile metric, because it might depends on external factors that are not under the control of the WfE; anyway according to the rule of large numbers, its result should be close to the actual value on average.
Relevance	It is relevant, because it represents the time to fulfil an atomic part of process instances requested by the user. Therefore the sum of all construct completion time produces the expected minimum time to execute the process instance without considering the overhead of communication between construct of process instances. In some case it might be misleading and it is required to have enough samples that allow applying the rule of large numbers. Therefore it is necessary to run enough process instance of the same process model to have consistent and valuable data for this metric.

Timeliness	The time required to compute completion time depends on the time that the Workflow Engine needs to execute construct and after the time required to store those data. Moreover in some cases it is necessary to execute enough samples and it should be taken into account when considering the timing of this metric.

Thus, the completion time metric respects all five SMART characteristics and even if it is difficult to reliably compute the metric, it is not impossible.

– delay

Metric name	Delay
Specificity	It is useful to determine construct time behaviour and how WfE manages the construct. So the metric is goal oriented. However workers might not have a clear definition in mind and might misinterpret its meaning with other. Anyway the definition of this metric has been clearly defined in previous steps (see Section 4.1.2).
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.
Attainability	It is difficult to compute, because it is composed by two parts: one part is the ideal expected duration and the other is the completion time. Completion time can be measured by analysing WfE data. By default WfE should stores into logs files or in databases data related to construct start time and end time; in some case they could store directly the time required to complete a construct. If WfE does not store this information by default, it is possible to enable history option to store these data. Moreover this value is a volatile value, because it might depend on external factors that are not under the control of the WfE; therefore according to the rule of large numbers, its value should be evaluated having enough samples. Instead the ideal expected duration is based on the construct. In addition it might be evaluated empirically by a baseline test or basic operation test. In this case the value is related to the WfE.
Relevance	It is relevant, because it represents the time inefficiencies of the WfMS. However it is considered as a part of completion time.

Timeliness	The time required to compute completion time depends on the time that the Workflow Engine needs to execute construct and after the time required to store those data. Moreover in some cases it is necessary to execute enough samples and it should be taken into account when considering the timing of this metric.

Thus, the delay metric respects all five SMART characteristics and even if it is difficult to compute the metric, it is not impossible.

– latency

Metric name	Latency
Specificity	It is useful to determine time behaviour of WfE, how WfE manages the communication among constructs and the execution flow of processes. So the metric is goal oriented. However workers might not have a clear definition in mind and might misinterpret its meaning with other. Anyway the definition of this metric has been clearly defined in previous steps (see Section 4.1.2).
Measurability	It is a single value expressed in time unit, therefore it can be easily compared to other response time values. In particular it is cardinal variable.
Attainability	It requires knowing exactly process model to determine the execution path and which construct come before and after the analysed one. Moreover additional problem are introduced considering process model with loops. Because it might cause error in identify the correct relationship between constructs. Regarding the data, WfE should stores into logs files or in databases data related to construct start time and end time. If WfE does not store this information by default, it is possible to enable history option to store these data. Anyway there might be precision problem, which means that the unit of measurement of stored data might not allow determining a precise and valuable metric.
Relevance	It is relevant, because it represents the time inefficiencies of the WfMS and in particular it affects the completion time of process instances. Moreover it is considered as a part of completion time of process instances.

Timeliness	The time required to compute completion time depends on the time that the Workflow Engine needs to execute construct and after the time required to store those data. Moreover in some cases it is necessary to execute enough samples and it should be taken into account when considering the timing of this metric.

Thus, the latency metric respects all five SMART characteristics and even if it is difficult to compute the metric, it is not impossible.

- capacity in:
  - construct capability

Metric name	Construct capability
Specificity	It is useful to determine the behaviour of the WfE to manage workload. So the metric is goal oriented. Moreover workers, usually, have at least an approximate idea of capability definition and by specifying the correct unit of measurement it is should not be misinterpreted.
Measurability	It is a value expressed by a quantitative measurement unit, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It can be measured by exploiting stored data by WfE. If WfE does not store this information by default, it is possible to enable history option to store these data. Anyway determine which is the maximum number of constructs that the WfMS can handle require specific testing and it might be influenced by the process.
Relevance	It is relevant, because it represents the ability to identify WfE limitation in the execution of particular constructs. Anyway this metric might be restricted by resource limitations, but this problem is limited by running benchmarks with the same configuration for all engines under analysis. Therefore the comparison between benchmark results should be applied only for engine that have been run under the same configuration.
Timeliness	The time required to compute capability depends on the time required by the Workflow Engine to execute process instances. Therefore it requires the completion of process instances execution, or at least the completion of the analysed construct of the process instances in execution.

Thus, the construct capability metric respects all five SMART characteristics and even if it is difficult to obtain data to compute the metric, it is not impossible.

– number of completed constructs

Metric name	NumCompletedConstructs
Specificity	It is useful to determine the behaviour of the WfE to manage workload. So the metric is goal oriented. Moreover workers, usually, have an approximate idea of the definition and by specifying the correct unit of measurement it is should not be misinterpreted.
Measurability	It is a value expressed by a quantitative measurement unit, therefore it can be easily compared to other values. In particular it is cardinal variable.
Attainability	It can be measured by exploiting stored data by WfMS. If WfMS does not store this information by default, it is possible to enable history option to store these data.
Relevance	It is relevant, because it represents the ability of the WfMS to process execution request of constructs.
Timeliness	The time required to compute number of completed constructs depends on the time required by the Workflow Engine to execute the constructs..

Thus, the number of completed constructs metric does respect all five SMART characteristics.

Bibliography

[1] Activiti. URL http://forums.activiti.org/.

[2] Camunda BPM docs. URL http://docs.camunda.org/.

[3] Docker. URL https://www.docker.com/.

[4] Apache jmeter. URL http://jmeter.apache.org/.

[5] MyBatis, . URL https://mybatis.github.io/mybatis-3/.

[6] MySQL Reference Manual, . URL https://dev.mysql.com/doc/refman/5.5/en/index.html.

[7] IEEE Standard for a Software Quality Metrics Methodology. *IEEE Std 1061-1992*, 1993. doi: 10.1109/IEEESTD.1993.115124.

[8] Abdul Azim Abd Ghani, Tieng Wei Koh, Geoffrey Muchiri Muketha, and Pei Wen Wong. Complexity metrics for measuring the understandability and maintainability of Business Process Models using Goal-Question-Metric (GQM). *International Journal of Computer Science and Network Security*, 8(5):219–225, 2008. ISSN 1738-7906. URL http://psasir.upm.edu.my/13726/.

[9] Veronika Abramova, Jorge Bernardino, and Pedro Furtado. Which nosql database? a performance overview. *Open Journal of Databases (OJDB)*, pages 17–24, 2014.

[10] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. *IEEE Expert*, 12, 1997.

[11] Arvind Arasu, Mitch Cherniack, Eduardo Galvez, David Maier, Anurag S. Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. Linear road: a stream data management benchmark. In *Proc. of the Thirtieth international conference on Very large data bases (VLDB 2004)*, VLDB 2004, pages 480–491. VLDB Endowment, 2004. ISBN 0-12-088469-0. URL http://dl.acm.org/citation.cfm?id=1316689.1316732.

[12] Victor R Basili, Gianluigi Caldeira, and H Dieter Rombach. The Goal Question Metric Approach. *Encyclopedia of Software Engineering*, 2006.

[13] BenchFlow. BenchFlow - a benchmark for workflow management systems. http://benchflow.inf.usi.ch, August 2014. URL http://benchflow.inf.usi.ch. http://benchflow.inf.usi.ch.

[14] Domenico Bianculli, Walter Binder, and Mauro Luigi Drago. Automated performance assessment for service-oriented middleware: A case study on BPEL engines. In *Proc. of the 19th International World Wide Web Conference (WWW '10)*, WWW '10, pages 141–150, 2010.

[15] Domenico Bianculli, Walter Binder, and Mauro Luigi Drago. SOABench: Performance evaluation of service-oriented middleware made easy. In *Proc. of the 32nd International Conference on Software Engineering (ICSE '10) - Volume 2*, ICSE'10, pages 301–302, 2010. ISBN 978-1-60558-719-6. doi: 10.1145/1810295.1810361. URL http://doi.acm.org/10.1145/1810295.1810361.

[16] Dina Bitton, Mark Brown, Rick Catell, Stefano Ceri, Tim Chou, Dave DeWitt, Dieter Gawlick, Hector Garcia-Molina, Bob Good, Jim Gray, et al. A measure of transaction processing power. *Datamation*, 31(7):112–118, 1985.

[17] Anthony J. Bonner, Adel Shrufi, and Steve Rozen. Labflow-1: A database benchmark for high-throughput workflow management. In *Proc. of the 5th International Conference on Extending Database Technology (EDBT '96)*, EDBT '96, pages 463–478, 1996. ISBN 3-540-61057-X.

[18] E. Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 45(2): 23–29, February 2012. ISSN 0018-9162.

[19] Coral Calero, Mario Piattini, and Marcela Genero. Method for Obtaining Correct Metrics. In *ICEIS (2)*, pages 779–784, 2001.

[20] Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011. ISSN 0163-5808. doi: 10.1145/1978915.1978919.

[21] Akmal B. Chaudhri. An annotated bibliography of benchmarks for object databases. *SIGMOD Rec.*, 24(1):50–57, 1995. ISSN 0163-5808. doi: 10.1145/202660.202668. URL http://doi.acm.org/10.1145/202660.202668.

[22] R. S. Chen, P. Nadkarni, L. Marenco, F. Levin, J. Erdos, and P. L. Miller. Exploring performance issues for a clinical database organized using an entity-attribute-value representation. *Journal of the American Medical Informatics Association: JAMIA*, 7(5):475–487, October 2000. ISSN 1067-5027.

[23] Michele Chinosi and Alberto Trombetta. Bpmn: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.

[24] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, June 1970. ISSN 0001-0782. doi: 10.1145/362384.362685. URL http://doi.acm.org/10.1145/362384.362685.

[25] Edgar F Codd. Further normalization of the data base relational model. *Data base systems*, pages 33–64, 1972.

[26] Desmond Lawrence Cook. *Program Evaluation and Review Technique: Applications in Education*. University Press of America, 1966. ISBN 978-0-8191-0657-5.

[27] Rick Copeland. *MongoDB Applied Design Patterns*. O'Reilly Media, Inc., March 2013. ISBN 978-1-4493-4004-9.

[28] Carlo Corti. Bpmetrics: a software system for the evaluation of some metrics for business process. Master's thesis, Politecnico di Milano, 2012.

[29] Florian Daniel, Giuseppe Pozzi, and Ye Zhang. Workflow engine performance evaluation by a black-box approach. In *Proc. of the International Conference on Informatics Engineering & Information Science (ICIEIS '11)*, ICIEIS '11, pages 189–203. Springer, November 2011.

[30] Marc Demarest. The politics of data warehousing. *Retrieved January*, 2:2008, 1997.

[31] Valentin Dinu and Prakash Nadkarni. Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases. *International journal of medical informatics*, 76(11-12):769–779, 2007. ISSN 1386-5056. doi: 10.1016/j.ijmedinf.2006.09.023. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2110957/`.

[32] George T Doran. There'sa SMART way to write management's goals and objectives. *Management review*, 70(11):35–36, 1981.

[33] Paul Dourish. Process Descriptions As Organisational Accounting Devices: The Dual Use of Workflow Technologies. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '01, pages 52–60, New York, NY, USA, 2001. ACM. ISBN 1-58113-294-8. doi: 10.1145/500286.500297. URL `http://doi.acm.org/10.1145/500286.500297`.

[34] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A Reijers. *Fundamentals of business process management*. Springer, 2013.

[35] Clarence A Ellis. Information control nets: A mathematical model of office information flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, volume 3670. Boulder, CO, 1979.

[36] Norman Fenton and James Bieman. *Software metrics: a rigorous and practical approach*. CRC Press, 3rd edition, 2014.

[37] Vincenzo Ferme, Ana Ivanchikj, and Cesare Pautasso. A framework for benchmarking BPMN 2.0 workflow management systems. In *Proc. of the 13th International Conference on Business Process Management (BPM '15)*, BPM '15. Springer, 2015.

[38] Vincenzo Ferme, Ana Ivanchikj, and Cesare Pautasso. Performance metrics for benchmarking BPMN 2.0 workflow management systems. 2015.

[39] Domenico Ferrari. *Computer systems performance evaluation*. Prentice Hall, 1978.

[40] Mark L Fussell. Foundations of Object-Relational Mapping. URL: `http://www.chimu.com/publications/objectRelational/index.html`, 1997.

[41] Michael Gillmann, Ralf Mindermann, and Gerhard Weikum. Benchmarking and configuration of workflow management systems. In *Proc. of the 7th International Conference on Cooperative Information Systems (CoopIS '00)*, CoopIS '00, pages 186–197, 2000.

[42] Jim Gray. *The Benchmark Handbook for Database and Transaction Systems*. Morgan Kaufmann, 2nd edition, 1992.

[43] Reijersa Hajo A. and Wil van der Aalst. The effectiveness of worflow management systems:predictions and lessons learned. *International Journal of Information Management*, 25:458–472, 2005.

[44] Paul Harmon and Business Process Trends. *Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals*. Morgan Kaufmann, July 2010. ISBN 978-0-08-055367-2.

[45] Paul Harmon and Celia Wolf. The state of business process management 2014. http://www.bptrends.com/bpt/wp-content/uploads/BPTrends-State-of-BPM-Survey-Report.pdf, March 2014.

[46] S. Harrer, C. Rock, and G. Wirtz. Automated and Isolated Tests for Complex Middleware Products: The Case of BPEL Engines. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 390–398, March 2014. doi: 10.1109/ICSTW.2014.45.

[47] K. Hayes and K. Lavery. *Workflow management software: the business opportunity*. Ovum Ltd, London, 1991.

[48] Thomas Herrmann and Marcel Hoffmann. The Metamorphoses of Workflow Projects in their Early Stages. *Computer Supported Cooperative Work (CSCW)*, 14(5):399–432, November 2005. ISSN 0925-9724, 1573-7551. doi: 10.1007/s10606-005-9006-8. URL `http://link.springer.com/article/10.1007/s10606-005-9006-8`.

[49] David Hollingsworth. Workflow management coalition the workflow reference model. *Workflow Management Coalition*, 68, 1995.

[50] William H Inmon. *Building the data warehouse*. John Wiley & Sons, 2005.

[51] International Organization for Standardization. ISO/IEC 25010".2011. *Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-System and software quality models*, 2011.

[52] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. Understanding object-relational mapping: A framework based approach. *International Journal on Advances in Software*, 1, Numbers 2&3, 2009.

[53] Ana Ivanchikj. Characterising representative models for BPMN 2.0 workflow engine performance evaluation. Master's thesis, Università della Svizzera Italiana, September 2014.

[54] S B Johnson. Generic data modeling for clinical repositories. *Journal of the American Medical Informatics Association*, 3(5):328–339, 1996. ISSN 1067-5027. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC116317/`.

[55] Diane Jordan and John Evdemon. Business Process Model And Notation (BPMN) version 2.0. Object Management Group, Inc, January 2011. `http://www.omg.org/spec/BPMN/2.0/`.

[56] Josh Juneau. Object-Relational Mapping. *Beginning Database-Driven Application Development in Java EE: Using GlassFish*, pages 369–408, 2013. URL http://link.springer.com/chapter/10.1007/978-1-4302-4426-4_8.

[57] Stylianos Kavadias and Svenja C. Sommer. The Effects of Problem Structure and Team Diversity on Brainstorming Effectiveness. *Management Science*, 55(12):1899–1913, September 2009. ISSN 0025-1909. doi: 10.1287/mnsc.1090.1079. URL http://pubsonline.informs.org/doi/abs/10.1287/mnsc.1090.1079.

[58] R. Khalaf, A. Keller, and F. Leymann. Business processes for Web Services: Principles and applications. *IBM Systems Journal*, 45(2):425–446, 2006. ISSN 0018-8670. doi: 10.1147/sj.452.0425.

[59] Ralph Kimball and Joe Caserta. *The data warehouse ETL toolkit*. John Wiley & Sons, 2004.

[60] Joakim v Kistowski, Nikolas Herbst, Daniel Zoller, Samuel Kounev, and Andreas Hotho. Modeling and extracting load intensity profiles. 2015. URL http://se2.informatik.uni-wuerzburg.de/pa/uploads/papers/paper-780.pdf.

[61] Robert V. Krejcie and Daryle W. Morgan. Determining Sample Size for Research Activities. *Educ Psychol Meas*, January 1970.

[62] Peter Kueng. The Effects of Workflow Systems on Organizations: A Qualitative Study. In Wil van der Aalst, Jorg Desel, and Andreas Oberweis, editors, *Business Process Management*, number 1806 in Lecture Notes in Computer Science, pages 301–316. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67454-2 978-3-540-45594-3. URL http://link.springer.com/chapter/10.1007/3-540-45594-9_19.

[63] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1984. ISBN 0-13-746975-6.

[64] Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. A distributed service-oriented architecture for business process execution. *ACM Transactions on the Web*, 4(1):2:1–2:33, January 2010. ISSN 1559-1131. doi: 10.1145/1658373.1658375.

[65] Zhongling Li. Object-Oriented Query Language. *IEEE A Technical White Paper*, 2006.

[66] M Sultan Mahmud, Saad Abdullah, and Shazzad Hosain. Gwdl: A graphical workflow definition language for business workflows. In *Recent Progress in Data Engineering and Internet Technology*, pages 205–210. Springer, 2013.

[67] J. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. *Performance Testing Guidance for Web Applications: Patterns & Practices*. Microsoft Press. ISBN 978-0-7356-2570-9.

[68] Daniel A. Menasce and Virgilio Almeida. *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall, 1st edition, 2001. ISBN 0130659037.

[69] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO' 89 Proceedings*, number 435 in Lecture Notes in Computer Science, pages 428–446. Springer New York, 1990. ISBN 978-0-387-97317-3 978-0-387-34805-6. URL http://link.springer.com/chapter/10.1007/0-387-34805-0_40.

[70] Ian Molyneaux. *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media, Inc., 2nd edition, 2014.

[71] Prakash M. Nadkarni. QAV: querying entity-attribute-value metadata in a biomedical database. *Computer Methods and Programs in Biomedicine*, 53(2):93–103, June 1997. ISSN 0169-2607. doi: 10.1016/S0169-2607(97)01815-4. URL http://www.cmpbjournal.com/article/S0169260797018154/abstract.

[72] Ted Neward. The vietnam of computer science. The Blog Ride, Ted Neward's Technical Blog, 2006.

[73] Matthias Nicola, Irina Kogan, and Berni Schiefer. An xml transaction processing benchmark. In *Proc. of the 2007 ACM SIGMOD international conference on Management of (SIGMOD/PODS '07)*, SIGMOD/PODS '07, pages 937–948, 2007. ISBN 978-1-59593-686-8. doi: 10.1145/1247480.1247590.

[74] M. Oba, S. Onoda, and N. Komoda. Evaluating the quantitative effects of workflow systems based on real cases. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000*, volume 2, January 2000. doi: 10.1109/HICSS.2000.926852.

[75] Business Process Model OMG. Notation BPMN Version 2.0. *OMG Specification, Object Management Group*, 2011. URL http://www.omg.org/spec/BPMN/2.0.

[76] Alison Parkes. Critical success factors in workflow implementation. In *Proceedings of the Sixth Pacific Asia Conference on Information Systems, Jasmin*, pages 363–380, 2002.

[77] Giuseppe Paterno. NoSQL Tutorial: A Comprehensive Look at the NoSQL Database. *Linux J.*, 1999(67es), November 1999. ISSN 1075-3583. URL http://dl.acm.org/citation.cfm?id=328036.328059.

[78] Cesare Pautasso, Vincenzo Ferme, Dieter Roller, Frank Leymann, and Marigianna Skouradaki. Towards workflow benchmarking: Open research challenges. In *Proc. of the 16th conference on Database Systems for Business, Technology, and Web (BTW 2015)*, BTW 2015, pages 331–350, 2015.

[79] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[80] Alan Ramias and Cherie Wilkins. Building metrics for a process, 2010. URL http://www.performancedesignlab.com/building-metrics-for-a-process.

[81] Eric Redmond, Jim R. Wilson, and Jacquelyn Carter. *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement*. Pragmatic Bookshelf, Dallas, Tex, 2012.

[82] Spiridon Reveliotis. *Real-Time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. Springer Science & Business Media, 2006. ISBN 978-0-387-23967-5.

[83] Luis Reynoso, Marcela Genero, and Mario Piattini. Refinement and extension of smdm, a method for defining valid measures. *J. UCS*, 16(21):3210–3244, 2010.

[84] Ronald L. Rivest. The md5 message-digest algorithm. *Internet activities board*, 1992.

[85] Cedric Röck and Simon Harrer. Literature survey of performance benchmarking approaches of BPEL engines. Technical report, Otto-Friedrich University of Bamberg, 2014.

[86] Geary A. Rummler and Alan P. Brache. *Improving Performance: How To Manage the White Space on the Organization Chart*. Jossey-Bass, January 1990.

[87] Marigianna Skouradaki, Dieter H. Roller, Leymann Frank, Vincenzo Ferme, and Cesare Pautasso. On the road to benchmarking BPMN 2.0 workflow engines. In *Proc. of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE '15)*, ICPE '15, pages 301–304, 2015. doi: 10.1145/2668930.2695527. URL `http://dx.doi.org/10.1145/2668930.2695527`.

[88] Standard Performance Evaluation Corporation. SPEC CPU2006 Version 1.2, September 2011.

[89] G. Stark, Robert C. Durst, and C.W. Vowell. Using metrics in management decision making. *Computer*, 27(9):42–48, September 1994. ISSN 0018-9162. doi: 10.1109/2.312037.

[90] Marc Stevens. Fast Collision Attack on MD5. *IACR Cryptology ePrint Archive*, 2006:104, 2006.

[91] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, number 5677 in Lecture Notes in Computer Science, pages 55–69. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03355-1 978-3-642-03356-8. URL `http://link.springer.com/chapter/10.1007/978-3-642-03356-8_4`.

[92] Transaction Processing Council (TPC). TPC Benchmark C (Online Transaction Processing Benchmark) Version 5.11, February 1997.

[93] B.G. Tudorica and C. Bucur. A comparison between several nosql databases with comments and notes. In *Proc. of the 10th Roedunet International Conference (RoEduNet 2011)*, RoEduNet 2011, pages 1–5, 2011. doi: 10.1109/RoEduNet.2011.5993686.

[94] Gaurav Vaish. *Getting Started with Nosql*. Packt Publishing Ltd, March 2013. ISBN 978-1-84969-499-5.

[95] Wil van der Aalst. Three Good Reasons for Using a Petri-Net-Based Workflow Management System. In Toshiro Wakayama, Srikanth Kannapan, Chan Meng Khoong, Shamkant Navathe, and JoAnne Yates, editors, *Information and Process Integration in Enterprises*, number 428 in The Springer International Series in Engineering and Computer Science,

pages 161–182. Springer US, 1998. ISBN 978-1-4613-7512-8 978-1-4615-5499-8. URL http://link.springer.com/chapter/10.1007/978-1-4615-5499-8_10.

[96] Wil van der Aalst and Kees Max van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2004. ISBN 978-0-262-72046-5.

[97] Wil van der Aalst, Arthur HM Ter Hofstede, and Mathias Weske. Business process management: A survey. In *Business Process Management (BPM 2003)*, BPM 2003, pages 1–12. Springer, 2003.

[98] Ian Thomas Varley, Adnan Aziz, Co-supervisors Adnan Aziz, and Daniel Miranker. No relation: the mixed blessings of non-relational databases. Master's thesis, The University of Texas at Austin, August 2009.

[99] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual Modeling for ETL Processes. In *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP*, DOLAP '02, pages 14–21, New York, NY, USA, 2002. ACM. ISBN 1-58113-590-4. doi: 10.1145/583890.583893. URL http://doi.acm.org/10.1145/583890.583893.

[100] Branimir Wetzstein, Philipp Leitner, Florian Rosenberg, Ivona Brandic, Schahram Dustdar, and Frank Leymann. Monitoring and analyzing influential factors of business process performance. In *Proc. of the IEEE International on Enterprise Distributed Object Computing Conference (EDOC '09)*, EDOC '09, pages 141–150, 2009. doi: 10.1109/EDOC.2009.18.

[101] Stephen White. Process modeling notations and workflow patterns. *Workflow handbook*, 2004:265–294, 2004.

[102] Stephen White. Introduction to BPMN. *IBM Cooperation*, 2, 2004. URL http://www.omg.org/bpmn/Documents/Introduction_to_BPMN.pdf.

[103] Benjamin B Yao, M Tamer Özsu, and John Keenleyside. Xbench-a family of benchmarks for XML DBMSs. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web*, pages 162–164. Springer, 2003.