



UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea in Informatica (Crema)

**Report Generator
per Dati di Processo
basato su .Net**

RELATORE
Prof. Ernesto Damiani

CORRELATORE
Dr. Ing. Sergio Oltolina

TESI DI LAUREA DI
Salvatore Camodeca
Matricola 648378

Anno Accademico 2010/2011

RINGRAZIAMENTI

Desidero esprimere con sincera gratitudine i ringraziamenti a tutti coloro hanno contribuito al raggiungimento di questo mio grande traguardo culminato da questa tesi.

In particolare ringrazio tutti i professori del corso **d'Informatica Industriale del Polo di Crema**, i quali puntualmente in questi anni mi hanno sempre sostenuto ad affrontare la mia particolare condizione, dandomi la possibilità di adoperare gli strumenti idonei e le modalità più comode per svolgere i corsi e superare tutti gli esami: in alcune occasioni recandosi amichevolmente al mio domicilio con totale abnegazione e merito professionale oltre che generosità.

Voglio rendere sentitamente grazie in modo speciale al **Prof. Ernesto Damiani** fraterno maestro, al **Dr. Fulvio Frati** formidabile guida e al **Dr. Sergio Oltolina (Engineering Group)** disponibilissimo esperto, per i quali ho potuto svolgere questa tesi con uno stage a distanza e che mi hanno seguito costantemente nella stesura dell'elaborato fornendomi ogni materiale ed informazione necessaria, rispondendo sempre ai miei quesiti e pazientando amichevolmente per le mie numerose richieste.

Porgo i miei ringraziamenti anche **all'ufficio Disabilità ed Handicap dell'Università Statale di Milano** attraverso il quale ho potuto intraprendere il mio percorso accademico e grazie al quale ho potuto usufruire di un prezioso supporto, composto anche dall'acquisto di sussidi tecnologici indispensabili per le mie gravi impossibilità fisiche.

In fine, ma non per meno importanza, ringrazio la mia famiglia, i miei genitori che mi hanno sostenuto ed accompagnato, gli amici per l'incoraggiamento continuo e la mia compagna **Mirtha** che mi ha donato spesso il motivo per continuare.

A Cinzia che il tempo non ha donato tali gioie ma che è in ogni mia vittoria.

Indice generale

SOMMARIO.....	4
INTRODUZIONE.....	5
ENGINEERING INGEGNERIA INFORMATICA.....	8
PROBLEMA E NECESSITA' AZIENDALI.....	10
2.1 SCENARI REALI.....	11
2.2 SOLUZIONI ESISTENTI (descrizione breve).....	12
SPAGO4Q.....	14
3.1 STRUTTURA E FUNZIONAMENTO.....	14
3.2 IL META-MODELLO.....	17
3.3 METRICHE UTILIZZATE.....	18
3.4 ESEMPIO STRUTTURA ISTANZA MODELLO.....	20
WEB SERVICE.....	24
4.1 DESCRIZIONE E FUNZIONAMENTO GENERALE.....	25
4.2 ESEMPIO STRUTTURA FILE WSDL.....	27
4.3 I WEB SERVICE DI SPAGO4Q.....	30
4.4 CREAZIONE WEB REQUEST HTTP.....	33
Spago4Q .NET Dashboard (soluzione SW creata).....	37
5.1 CARATTERISTICHE DASHBOARD E REPORT WINDOWS.....	38
5.2 TOOLS DI SVILUPPO UTILIZZATI.....	39
5.3 SPECIFICHE RICHIESTE.....	40
5.4 ALGORITMO DI ESTRAZIONE ED ANALISI XML.....	41
5.5 UTILIZZO COME DLL.....	46
DESCRIZIONE TECNICA DETTAGLIATA.....	48
6.1 GRAFICI UML.....	50
6.2 GRAFICI UML RELATIVI ALLA DLL.....	65
6.3 CODICE SORGENTE DLL E SUO FUNZIONAMENTO.....	67
6.4 CODICE SORGENTE (parti salienti Windows Forms Application).....	82
CONCLUSIONI.....	96
7.1 SVILUPPI FUTURI.....	96
7.2 LIMITAZIONI DA RISOLVERE.....	97
RIFERIMENTI.....	98

SOMMARIO

La tesi in oggetto riguarda la creazione di un software specifico per le piattaforme **Windows** che generi *dashboard e report* per la visualizzazione all'utente dei valori indicanti lo stato di evoluzione e/o della qualità di un prodotto, di un servizio o di un processo.

Per il raggiungimento dell'obiettivo è stato possibile utilizzare la piattaforma **Spago4Q** che consente la misurazione dei parametri secondo un *framework* di metriche prestabilite, mettendo a disposizione le collezioni dei dati da prelevare e da poter mostrare agli utenti con una grafica idonea.

Il mio lavoro è stato possibile con il contributo di **Engineering** sede di Assago Milanofiori (Milano), e che ha permesso di svolgere uno stage a distanza, fornendomi inoltre tutte le specifiche tecniche per l'interfacciamento con i loro server e *web services*.

La creazione del SW è stata, per le mie particolari necessità, eseguita con il totale utilizzo di **Microsoft Visual Studio 2010** e dell'uso del linguaggio di programmazione ad oggetti **C# .Net**, in quanto era per me indispensabile un tool di sviluppo dotato di strumenti di creazione visuali efficaci e che rendesse fattibile unire comodità, potenza e affidabilità per la stesura del codice applicativo in ambiente **Windows**.

Scopo della tesi è stato l'implementazione di un *framework* per generare dei *report Windows-Style* elaborando in remoto i files di risorsa in formato **XML**, estraendo da essi un set di valori selezionabili dall'utente secondo le sue esigenze.

INTRODUZIONE

Engineering Group fra le sue molteplici attività nel settore dei prodotti informatici ha creato e continua ad impegnarsi nell'evoluzione della piattaforma *open source* chiamata *Spago4Q*, gestendo la comunità e guidando il team di sviluppatori rivolta principalmente a chiunque voglia costantemente monitorare l'andamento e la qualità dei propri lavori, in particolare per quelli di software.

Questa piattaforma è in grado di acquisire i dati necessari alle elaborazioni di misurazione con metriche dedicate e di posizionarli in *Data warehouse[1]* remoti, così da renderli disponibili a chi vuole usufruire del servizio di monitoraggio in ogni istante e da ogni luogo.

Nella creazione di un prodotto, oppure nell'offerta di un servizio di natura **ICT**, nelle aziende e nelle imprese più professionalmente esperte si segue un processo di sviluppo ben definito, così come nell'ingegneria del software si segue un modello di sviluppo come ad es. il *Visual Modeling* o *Waterfall Model[2]*.

In ogni fase dell'evoluzione del prodotto è dunque fondamentale per chi lo sviluppa conoscere esattamente gli obiettivi da raggiungere ed in quale stato attuale ci si trova, al fine di poter stabilire il prossimo passo da compiere e se i livelli di soddisfacimento sono sufficientemente positivi.

Soprattutto per progetti molto complessi le fasi hanno una durata temporale maggiore e si prendono in considerazione molteplici proprietà descrittive per poter valutare i risultati auspicati, pertanto è necessario programmare anche con quale metodologia è opportuno verificare il manufatto o il **SW** realizzabile.

Un altro aspetto complesso è che nello sviluppo di software e nella produzione di servizi, i quali molto soventemente richiedono l'utilizzo di molteplici *devices* programmabili, per la grossa mole di lavoro richiesta vengono impegnate numerose risorse fisiche (operatori umani) ed anche strumenti che vanno tutti coordinati e controllati come in un unico sistema, quindi è difficile seguire parallelamente tutti gli stati svolgendo le verifiche costantemente senza l'utilizzo di supporti che aiutino in modo semplice e non esageratamente dispendioso all'analisi qualitativa e quantitativa di ciò che è in atto.

Il software implementato per la tesi come descritto dettagliatamente nel seguito, in sintesi è in grado di prelevare da un *Data warehouse Spago4Q* dedicato attraverso *web service* i dati in un formato **XML**, successivamente di elaborarli e di mostrarli all'utente con una visualizzazione testuale e grafica intuitiva.

Le risorse **XML** contenenti i dati suddivisi in diverse chiavi che assumono precisi significati, sono dunque la parte fondamentale sulla quale si concentra la tesi nei metodi di ricerca delle risorse e della loro estrazione mirata.

Le finalità della soluzione creata consistono nel poter importare e consultare i dati di **Spago4Q** anche nelle applicazioni **.Net**, per la quale molte aziende partecipano all'ascesa della sua diffusione dovuta al fatto che è installabile su differenti tipi di dispositivi compresi quelli di *user mobile*, oltre che sui classici server e PC fissi.

L'applicativo è stato costruito dando inoltre rilievo alle proprietà di personalizzazione ed integrazione con altri ambienti derivati dalla piattaforma **Microsoft** come ad esempio **Sharepoint**.

Con questa prima release è stato dunque possibile ottenere una *Windows Forms Application* di base con tutte le funzioni disponibili, ma è stata anche costruita una **DLL**

inseribile in ogni nuovo progetto **.Net** con ogni suo linguaggio di programmazione: **C#**, **Visual Basic**, **J#** e quelli compatibili forniti da produttori terzi.

E stato quindi pensato il tutto nella prospettiva di derivazione delle funzioni e nel miglioramento dei modi di consultazione delle collezioni dei dati di misura per sistemi

O.S. Windows.

Capitolo 1

ENGINEERING INGEGNERIA INFORMATICA

Engineering Group è un *player* mondiale del ramo informatica e software con un'esperienza trentennale in diversi ambiti applicativi. Lavora in diversi paesi con oltre 40 sedi italiane ed estere, attestandosi così fra i leader nello sviluppo e nella ricerca di prodotti informatici per la diffusione della conoscenza e l'automazione di processi aziendali.

Il modello di business è dato dalla ricerca ed innovazione nel mercato **ICT** che rende tale azienda la 3° per produttività in Italia e fra i partner strategici in Europa con un'offerta integrata che svolge attività di: consulenza, system & business integration, servizi di outsourcing e soluzioni di servizi verticali.

Grazie ai suoi oltre 6000 dipendenti ci sono numerosi professionisti esperti che si adoperano in tutta l'offerta che l'azienda offre: progetti di ricerca, finanza, PA e sanità, telecomunicazioni, multimedia ecc., per soddisfare oltre i 1000 clienti serviti fra cui i prestigiosi gruppi come: **BNL, ENI, FIAT Group, INPS, Telecom** e altri ancora importanti.

E' stata anche una delle prime aziende a ricevere le certificazioni internazionali quando ancora non erano obbligatorie, ed attualmente tutte le attività del gruppo sono certificate secondo i più moderni standard come ad es.: **ISO9001-2000[3]**, **Nato AQAP 2110/160[4]**, **ISO 14001:2004[5]** e altri mirati a differenti segmenti.

Fra le molteplici attività, **Engineering** lavora anche nel campo *Open Source* con più di 50 risorse attive che costituiscono il “Centro di competenza per l'*Open Source*,

Business Intelligence e SOA[6]?. Le soluzioni di questo tipo riguardano principalmente i domini di:

- Business Intelligence
- Business Activity Monitoring
- Business Process Management
- Architetture e servizi XaaS
- Cloud
- Qualità di prodotti e servizi
- Sviluppi in *Java Enterprise*
- Interoperabilità

Fra tali attività quella per lo sviluppo di soluzioni ***Open Source*** è ben consolidata con una gamma di prodotti molto utili che fanno parte della suite **SpagoWorld**: (***SpagoBI, Spagic, Spago e Spago4Q***)[7], quest'ultimo in particolare come oggetto della tesi.

Oltre ad impegnarsi direttamente nella creazione di software del tipo open source è anche partner delle comunità più famose al mondo fra cui: ***OW2 Consortium, Eclipse*** ed il **Centro di Competenza Italiano per l'Open Source (Qualipso)**[8].

Capitolo 2

PROBLEMA E NECESSITA' AZIENDALI

Nell'era attuale in cui l'evoluzione tecnologica progredisce a ritmi rapidissimi, soprattutto grazie alle tecnologie informatiche con l'importanza sempre più predominante dei software di controllo di tutti i dispositivi hardware, l'offerta di prodotti destinati sia allo *user consumer* che alle grosse imprese commerciali è vastissima ed in costante ampliamento.

E' necessario quindi che i produttori che competono nel campo della vendita di prodotti software raggiungano livelli qualitativi ben prefissati oltre che specifiche certificazioni, pertanto devono seguire processi di sviluppo precisi nei quali molti fattori vanno analizzati per stimare i significativi valori di stato.

Ovviamente la precisione richiede metodo, dunque è indispensabile disporre anche di strumenti automatici di monitoraggio della qualità e di cattura ed analisi dei dati.

Per questo scopo esistono in commercio diversi applicativi professionali con diverse caratteristiche, ma sovente sono alquanto costosi e questo non li rende facilmente accessibili alle imprese di minori dimensioni o che comunque dispongono di risorse più esigue. Fortunatamente l'*Open Source* è attivo anche nel settore della *Business Intelligence* e fra i molteplici applicativi disponibili in rete è possibile trovare buoni prodotti adattabili alle diverse esigenze oltre che adoperabili con basse, e spesso nulle, spese di licenza e adozione. Fra queste la piattaforma *Spago4Q* è assolutamente una delle migliori e delle più funzionali soluzioni per chiunque con delle personalizzazioni voglia disporre di un sistema molto complesso e rigoroso di misurazione dei propri lavori.

Caratteristica principale di *Spago4Q* è quella di poter gestire monitoraggi della qualità del processo di sviluppo del software eseguendo misurazioni multi-progetto, quindi applicabili a tutto il bouquet di progetti in lavorazione, e multi-processo, ossia gestendo progetti che adottano cicli di vita del software completamente diversi da loro e apparentemente semanticamente non paragonabili.

Tale caratteristica rende la piattaforma unica nel panorama delle applicazioni *open source* di *business intelligence*.

2.1 SCENARI REALI

I fornitori di software e di servizi fanno oggi affidamento su sistemi capaci di trattare il *Software Quality Model*, attraverso il quale si definisce la conformità dei requisiti e l'idoneità per l'uso dell'applicazione. Riguardo specialmente la qualità del codice sorgente nell'organizzazione di un progetto da implementare, ogni operatore infatti, in termini pratici, deve poter conoscere costantemente tutte le specifiche aggiornate, quali sono gli obiettivi da raggiungere, quali sono i criteri evolutivi e quali contributi deve dare per ottenere gli obiettivi prefissati.

Per perseguire una buona direzione occorre contemplare alcune proprietà comuni per un buon prodotto come ad esempio la facile testabilità oppure la semplice manutenzione.

Un altro step fondamentale è anche quello d'individuare e catalogare i fattori variabili più importanti per i risultati finali, i quali vengono solitamente chiamati predicati degli obiettivi e possono consistere in nomi comuni o essere dei valori numerici calcolati con precise metriche.

Soprattutto nei lavori in team ogni programmatore ad esempio necessita di conoscere lo stato a cui si è giunti considerando anche il lavoro di tutti gli altri colleghi, quindi non soltanto della propria visione, e questo magari da un luogo distante e senza che si ricorra sempre al dialogo diretto.

Occorre quindi saper pianificare quali strumenti e come adoperarli per tali misurazioni e fra quelle disponibili come *Swat4j*[9] (*quality model for Java*) oppure *Squale*[10], certamente *Spago4Q*[7] è già parecchio diffusa ed utilizzata, inoltre grazie ai diversi *partners* sono possibili anche corsi di formazione per consentire la diffusione maggiore.

2.2 SOLUZIONI ESISTENTI (descrizione breve)

Il prodotto *Swat4J* è un pacchetto dedicato ai programmatori *Java* in quanto consente una verifica della qualità del codice sorgente prodotto, basandosi sul modello “*goal oriented*”. Esso è stato progettato seguendo i principi delle certificazioni **ISO 9126-1 (Quality Model)** e **ISO 9126-3 (Software Product Quality)**. La sua caratteristica è quella di offrire una garanzia di qualità controllando la “*Code Entropy*”, ovvero una produzione di codice che segua metriche prestabilite e le “*best practice rules*” delle aziende più esperte. Il sistema di auditing del codice è automatico e si integra perfettamente con gli strumenti di sviluppo *Java* con un *plug-in* per il diffuso IDE *Eclipse*, inoltre la soluzione è personalizzabile per mezzo di parametri di analisi, di metriche da adottare e dei modelli di report per la visualizzazione. Per utilizzare tale strumento è quindi necessario che ogni *stakeholder* del gruppo di lavoro stabilisca a priori quali obiettivi si vogliono raggiungere (i goal del modello), quindi impostare l'analisi per verificare gli aspetti scelti e monitorarli durante la stesura del codice, ad esempio la testabilità o l'affidabilità. Il passo successivo è di stabilire i predicati che

occorreranno alle misure degli obiettivi che costituiranno le metriche software di riferimento. E' dunque una soluzione anch'essa improntata sul “*Quality Model*”, ma che è esclusiva per la creazione dei programmi *Java*. La sua licenza è di tipo commerciale a pagamento ed è possibile scaricare ed installare il *plug-in* per poterlo provare per un periodo limitato, perciò a differenza di *Spago4Q* richiede l'esborso di denaro per un completo utilizzo.

Il software *Squale* è un progetto di tipo *Open Source* fruibile pienamente secondo la filosofia *freeware* e installabile con licenza **GNU**. Si basa anch'esso sul paradigma “*Quality Model*” ed è ispirato agli standard **ISO 9126**. Permette tuttavia un'analisi multi-linguaggio, cioè consente di essere adoperato nei progetti di sviluppo svolti con differenti linguaggi come: *C++*, *Java*, *C#*, *PHP*, *.Net*, *Cobol* ed altri.

Il suo ciclo di vita è composto da un'estrazione del codice sorgente, dalla memorizzazione di tale codice in un *repository*, dall'analisi di copertura del codice, dal monitoraggio dei *bug* ed infine dalla reportistica dei risultati mediante *Dashboard* o portale web, infatti una sua utile peculiarità è che permette di accedere ai risultati automaticamente attraverso un sito web dedicato. Dispone inoltre di un *plug-in* per **Eclipse** ed è pienamente personalizzabile attraverso un pannello con i vari menu'.

In comune con i precedenti prodotti è l'utilizzo dello stesso meta-modello, ma a proprio vantaggio non si limita all'uso con un solo linguaggio ed è inoltre *free* per consentire di essere utilizzato, studiato ed ampliato da chiunque lo desideri.

Capitolo 3

SPAGO4Q

Per lo svolgimento della tesi inerente il controllo della qualità di un processo e/o di un prodotto software, è stato scelto di riferirsi allo strumento *Spago4Q (SpagoBI for Quality)* della suite *SpagoWorld*.

Come per le soluzioni precedentemente menzionate anche questa è di tipo *freeware* nata secondo la filosofia *Open Source* e si occupa della generazione, del monitoraggio e dell'analisi dei dati per la misurazione qualità servizi, processi e software.

Tale piattaforma è personalizzabile per ogni tipo di organizzazione indipendentemente dai metodi e dagli strumenti di sviluppo usati, inoltre consente di ottenere livelli di qualità basati sui modelli più famosi di valutazione: **ISO 9001**[3], **ITIL**[11], **CMMI**[12], **GQM**[13] etc..

L'utilizzo in fase di lavoro da parte dei programmatori non è invasivo in quanto procede a prelevare i dati, anche contemporaneamente da più sorgenti, senza interferire con gli altri *tool* adoperati in un modo del tutto trasparente. Nasce inoltre come verticalizzazione del progetto *SpagoBI*, dunque dispone di un meta-modello specificatamente dedicato alla valutazione della qualità del software.

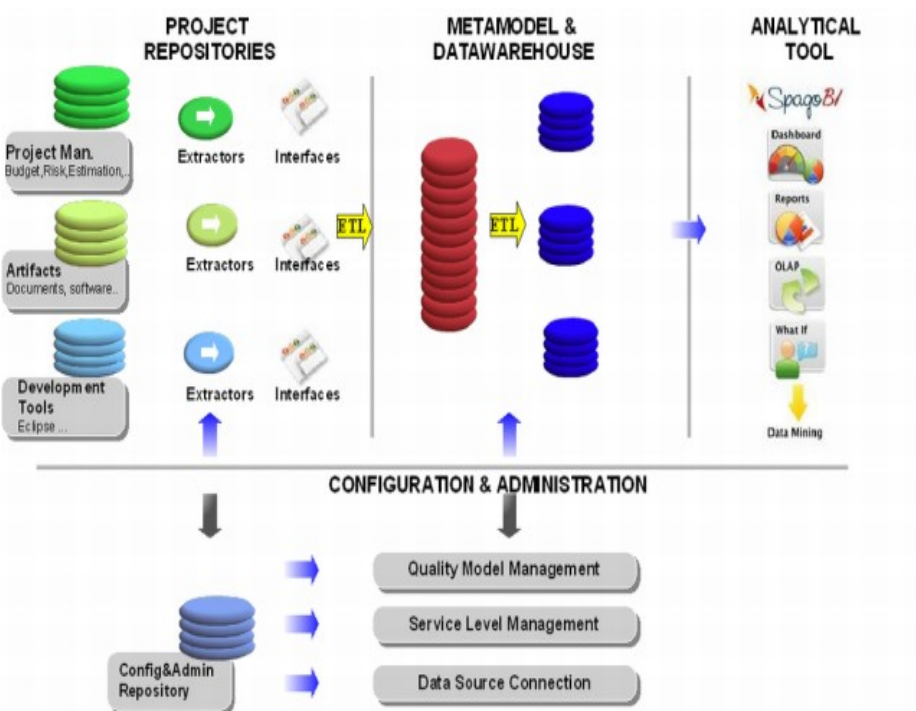
3.1 STRUTTURA E FUNZIONAMENTO

L'architettura della piattaforma è composta da 3 parti collegate:

- Modulo di estrazione **ETL (Extract Transform Load)** il quale si occupa di prelevare i dati processo, servizio o software dai *tool* infrastrutturali

- Modulo di inserimento per la popolazione di un *Data warehouse* dedicato di *Spago4Q* secondo l'implementazione di uno specifico meta-modello
- Modulo di analisi, gestione, amministrazione e sicurezza dei dati collezionati nel *Data warehouse*

figura 1 - Struttura piattaforma Spago4Q



L'estrazione avviene con l'ausilio di *repository* e la conservazione dei dati raccolti nel sistema *Data warehouse* godono d'integrità protetta ed inoltre vengono suddivisi in base ad ogni *work-product*. Nella fase di analisi si applica il meta-modello basilare della piattaforma per cui si considerano gli attributi, i predicati e le proprietà proprie del modello. Le funzioni di visualizzazione offrono invece la generazione di *report*, *dashboard* e cruscotti personalizzati attraverso il componente di amministrazione e configurazione.

Nella definizione del modello è possibile impostare alcuni parametri significativi come ad esempio il tipo di struttura organizzativa (azienda, *Business Unit*, progetti di servizio), oppure quale metodologia di sviluppo verrà seguita fra ad es. *waterfall*, *evolutionary*, *SCRUM*, *UP* o gli altri proposti nell'ingegneria del software. Si può definire inoltre a quale *Measurement framework* riferirsi come ad esempio il modello **GQM**, oppure l'*assessment framework* da considerare come **CMMI**, **ISO9001-2000** ecc..

Per la rappresentazione dei dati elaborati al termine del processo di funzionamento viene utilizzato il formato **XML**, poiché tale standard de facto universale consente l'interoperabilità totale su ogni sistema. La potenza del metalinguaggio di *mark-up* intrinseca in **XML** permette di definire la sintassi utile all'estensione o al controllo di linguaggi marcatori derivati da esso. Nel contesto della creazione di una risorsa **Spago4Q** è stato implementato attraverso una serie di *tag* un sottolinguaggio dedicato in grado di esporre in forma testuale tutte le dimensioni, gli attributi, i valori e le relative soglie di ogni *work-product* in esame.

In questo senso il software realizzato analizza tali file **XML** dopo il loro download per la successiva interpretazione e visualizzazione.

3.2 IL META-MODELLO

Alla base dunque di ogni soluzione troviamo la necessità di formulare ed applicare un meta-modello[14] idoneo al dominio applicativo. Questo servirà al controllo di qualità, ovvero una proprietà che può essere definita come l'insieme delle caratteristiche di un'entità che portano alla soddisfazione delle esigenze espresse.

Per la piattaforma *Spago4Q* è stato usato il tipo **MOF (Meta-Object Facility)** proposto da **OMG (Object Management Group)**[15] il quale è strutturato in 3 componenti:

- **Process**
 - Descrive un processo di sviluppo ed è stato definito secondo una versione semplificata del **OMG's SPEM (Software Process Engineering Meta-model)**
- **Measurement**
 - E' strutturato secondo il paradigma **GQM (Goal Question Metric)** ed è composto da 3 entità
 - **MeasurableConcept** che definisce l'obiettivo di misura
 - **MeasurableAttribute** che rappresenta gli attributi presi in esame per la valutazione del raggiungimento dell'obiettivo
 - **"KPI" (Key Process Indicator)** e metriche per i valori quantitativi utili al calcolo dei risultati di stima degli obiettivi
- **Assessment meta-model**
 - Si occupa del controllo del raggiungimento degli obiettivi ed è conforme, in quanto **framework** di valutazione, agli standard **CMMI (Capability Maturity Model Integration)** e **ISO9001-2000**

Tramite questo modello si possono rappresentare principalmente 3 oggetti astratti:

- Tipo del processo di sviluppo (es. *waterfall*, *UP*, *SCRUM*, *evolutionary*)
- *Framework* di valutazione (ad es. *QualiPSo*, *ISO9000-2001*, *ISO9126*, *CMMI* ecc.)
- Le "**KPI**" che rappresentano gli attributi con valori quantitativi per il monitoraggio effettivo della qualità sotto osservazione

3.3 METRICHE UTILIZZATE

Molto importanti per la loro semantica sono tutte le metriche che istanziano il meta-modello e che seguono il modello **GQM** (*Global Question Metric*). In sintesi tale approccio indica di procedere secondo la seguente modalità : definire un obiettivo importante per il business aziendale, stabilire una domanda che abbia come risposta un valore quantitativo per stabilire l'esito del raggiungimento, in fine definire la metrica che soddisfi la domanda.

In *Spago4Q* vengono prettamente individuate delle dimensioni su cui indirizzare le analisi ed ogni dimensione si riferisce ad un obiettivo (*goal*). Ognuna di esse inoltre è composta da più componenti da monitorare, le quali corrispondono a più attributi e formano l'intero sistema di metriche che concorrono all'obiettivo da cui sono derivate, dunque si rappresenta l'intera struttura dati di un documento risorsa estratto di tipo ad albero.

Per un esempio pratico si consideri la valutazione qualitativa di un nuovo software per cui è disponibile un file **XML** di risorsa elaborato da *Spago4Q* che mostri lo stato dell'opera, quindi possiamo immaginare un nodo principale nel quale si definisce

l'obiettivo generale come performance del prodotto, ed in un secondo livello sarà possibile definire le dimensioni che costituiscono le parti concorrenti all'obiettivo generale e che a loro volta avranno ognuna una "**KPI**" (*Key Process Indicator*) *performance* associata. Nei successivi sotto-nodi saranno poi indicati i *goal* che dovranno corrispondere a più *questions* per le quali potranno esserci più metriche. Ogni "**KPI**" di livello superiore in ogni dimensione, sarà il risultato della media delle "**KPI**" di tutte le metriche associate ad essa attraverso le *questions goal* le quali saranno comunque definite ad un livello inferiore e le metriche collegate al livello successivo.

3.4 ESEMPIO STRUTTURA ISTANZA MODELLO

Analisi e descrizione di una risorsa XML.

listato XML 1 – esempio file di risorsa Spago4Q (nodi primari)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DOCKPI>
  <TITLE/>
  <SUBTITLE/>
  <RESOURCE name="BS01">
    <MODELINSTANCENODE code="BS-1" description="Instance of the QESTnD model for Business Service"
      name="QEST nD for Business Service">
      <KPI code="QEST-KPI-BS" description="Indicatore Performance Globale per il modello Qest
        Business Service" interpretation="" name="QEST-KPI-BS"/>
      <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:46:00.0"
        target="" thresholdid="6" value="0.9995" weight="" weightedvalue=""/>
      <MODELINSTANCENODE code="BS-CS" description="" name="Customers Satis">...</MODELINSTANCENODE>
      <MODELINSTANCENODE code="BS-EC" description="" name="Economic Perfor">...</MODELINSTANCENODE>
      <MODELINSTANCENODE code="BS-RS" description="" name="Resources Perfo">...</MODELINSTANCENODE>
      <MODELINSTANCENODE code="BS-TE" description="" name="Technical Perfo">...</MODELINSTANCENODE>
    </MODELINSTANCENODE>
  </RESOURCE>
  <THRESHOLDS>...</THRESHOLDS>
</DOCKPI>
```

Nel **listato XML 1** precedente si nota un nodo **"MODELINSTANCENODE"** con una **"KPI"** che indica il valore di performance globale, poi al suo interno sono presenti i sotto-nodi **"MODELINSTANCENODE"** ognuno rappresentante le dimensioni considerate: *Customer satisfaction, Economic, Resource e Technical performance*.

listato XML 2 – esempio file di risorsa Spago4Q (sottonodi)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DOCKPI>
  <TITLE/>
  <SUBTITLE/>
  <RESOURCE name="BS01">
    <MODELINSTANCENODE code="BS-1" description="Instance of the QESTnD model for Business Service"
      name="QEST nD for Business Service">
      <KPI code="QEST-KPI-BS" description="Indicatore Performance Globale per il modello Qest
        Business Service" interpretation="" name="QEST-KPI-BS"/>
      <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:46:00.0"
        target="" thresholdid="6" value="0.9995" weight="" weightedvalue=""/>
    <MODELINSTANCENODE code="BS-CS" description="" name="Customers Satisfaction Performance
      Indicator">
      <KPI code="QEST-CS" description="Valore" interpretation="" name="QEST-CS"/>
      <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:45:59.0"
        target="" thresholdid="33" value="0.9033" weight="1.0" weightedvalue="0.9033"/>
      <MODELINSTANCENODE code="BS-CS-G1" description="" name="Training">
      <KPI code="QEST-GOAL-CS-1" description="" interpretation="" name="QEST-GOAL-CS-1"/>
      <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:45:59.0"
        target="" thresholdid="" value="0.89" weight="" weightedvalue=""/>
    <MODELINSTANCENODE code="BS-CS-Q1.1" description="" name="Is the training">...</MODELINSTANCENODE>
  </MODELINSTANCENODE>
  <MODELINSTANCENODE code="BS-CS-G2" description="" name="Customers Satis">...</MODELINSTANCENODE>
  <MODELINSTANCENODE code="BS-CS-G3" description="" name="Usability">...</MODELINSTANCENODE>
</MODELINSTANCENODE>
```

Nel **listato XML 2** si può notare come al terzo livello si trovano i nodi **"MODELINSTANCENODE"** che sono i **goals: Training, Customer, Usability**, e per ogni **goal** esiste una **"KPI"** come valore medio di tutte le **"KPI"** del livello successivo.

listato XML 3 – esempio file di risorsa Spago4Q (nodi livello 4°)

```

<RESOURCE name="BS01">
  <MODELINSTANCENODE code="BS-1" description="Instance of the QESTnD model for Business Service"
name="QEST nD for Business Service">
    <KPI code="QEST-KPI-BS" description="Indicatore Performance Globale per il modello Qest
Business Service" interpretation="" name="QEST-KPI-BS"/>
    <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:46:00.0"
target="" thresholdid="6" value="0.9995" weight="" weightedvalue=""/>
  <MODELINSTANCENODE code="BS-CS" description="" name="Customers Satisfaction Performance
Indicator">
    <KPI code="QEST-CS" description="Valore" interpretation="" name="QEST-CS"/>
    <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:45:59.0"
target="" thresholdid="33" value="0.9033" weight="1.0" weightedvalue="0.9033"/>
  <MODELINSTANCENODE code="BS-CS-G1" description="" name="Training">
    <KPI code="QEST-GOAL-CS-1" description="" interpretation="" name="QEST-GOAL-CS-1"/>
    <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31 13:45:59.0"
target="" thresholdid="" value="0.89" weight="" weightedvalue=""/>
  <MODELINSTANCENODE code="BS-CS-Q1.1" description="" name="Is the training sufficient in
relation to needs">
    <MODELINSTANCENODE code="BS-CS-M1.1.1" description="" name="Training ratio">
      <KPI code="QEST-CS-1" description="" interpretation="" name="QEST-CS-1"/>
      <KPIVALUE begindate="2010-06-07 13:45:58.0" description="" enddate="9999-12-31
13:45:59.0" target="" thresholdid="40" value="0.89" weight="1.0" weightedvalue="0.89"/>
    </MODELINSTANCENODE>
  </MODELINSTANCENODE>
</MODELINSTANCENODE>
<MODELINSTANCENODE code="BS-CS-G2" description="" name="Customers Satis">...</MODELINSTANCENODE>
<MODELINSTANCENODE code="BS-CS-G3" description="" name="Usability">...</MODELINSTANCENODE>
</MODELINSTANCENODE>

```

Nel **listato XML 3** sopra raffigurato si mostra come nel 4° livello è espressa la **question**, in questo caso per la performance del **Training** che nel nodo **"MODELINSTANCENODE"** in esame possiede l'attributo **name** con valore *"Is the training sufficient..."*.

Si vede inoltre al livello 5 quale sia la metrica usata, in questo caso il **"MODELINSTANCENODE"** con valore *"Training ratio"*.

Nelle misurazioni espresse nel file XML sono inoltre comprese le soglie di riferimento quantitative con i range di stato per i significati di **bad**, **warning** **good level**, come mostrato nella seguente immagine.

listato XML 4 – esempio file di risorsa Spago4Q (sezione range)

```
<THRESHOLDS>
  <THRESHOLD code="QEST-KPI" id="6" type="RANGE">
    <RANGE color="#FF3300" label="Bad" max="0.5999" min="0.0"/>
    <RANGE color="#FFFF00" label="Warning" max="0.8499" min="0.6"/>
    <RANGE color="#00FF00" label="Good" max="1.0" min="0.85"/>
  </THRESHOLD>
  <THRESHOLD code="QEST-CS" id="33" type="RANGE">
    <RANGE color="#FF3300" label="Bad" max="0.4999" min="0.0"/>
    <RANGE color="#FFFF00" label="Warning" max="0.7499" min="0.5"/>
    <RANGE color="#00FF00" label="Good" max="1.0" min="0.75"/>
  </THRESHOLD>
  <THRESHOLD code="QEST-CS-1 " id="40" type="RANGE">
    <RANGE color="#FF3300" label="Training scarso" max="0.6999" min="0.0"/>
    <RANGE color="#FFFF00" label="Training accettabile" max="0.8999" min="0.7"/>
    <RANGE color="#00FF00" label="Training in linea" max="1.0" min="0.9"/>
  </THRESHOLD>
  <THRESHOLD code="QEST-CS-2" id="41" type="RANGE">
    <RANGE color="#FF3300" label="Scarsa soddisfazione" max="0.6999" min="0.0"/>
    <RANGE color="#FFFF00" label="Media soddisfazione" max="0.8999" min="0.7"/>
    <RANGE color="#00FF00" label="Alta soddisfazione" max="1.0" min="0.9"/>
  </THRESHOLD>
  <THRESHOLD code="QEST-CS-3" id="42" type="RANGE">
    <RANGE color="#FF3300" label="Bassa usabilita" max="0.8499" min="0.0"/>
    <RANGE color="#FFFF00" label="Media usabilita" max="0.9499" min="0.85"/>
    <RANGE color="#00FF00" label="Usabilita in linea" max="1.0" min="0.95"/>
  </THRESHOLD>
</THRESHOLDS>
```

Nello specifico in ogni nodo "KPIVALUE" è presente un attributo *thresholdid* di corrispondenza con l'attributo "id" dell'elemento "THRESHOLD" utile all'identificazione del range.

Capitolo 4

WEB SERVICE

Nell'applicazione **Spago4Q .Net Dashboard** oggetto di tale tesi, per la quale è necessario interfacciarsi con le risorse della piattaforma **SpagoWorld**, la modalità di connessione è consentita dalla presenza dei *web services*[16] dedicati ed installati sui loro server con i relativi **WSDL** in grado di offrire i servizi in rete ad applicativi eseguibili su macchine con diversi **O.S.** e *tool* di sviluppo.

Negli ultimi anni per la rapida evoluzione dei siti e dei servizi web, molte risorse sono disponibili in remoto ed esistono degli strumenti applicativi molto potenti. I *web service* costituiscono dei componenti software che offrono dei servizi alle applicazioni che si interfacciano con essi e che vengono eseguite dai PC collegati su una medesima rete, solitamente attraverso la porta di comunicazione dedicata al protocollo **HTTP 80**.

Ogni *web service* mette a disposizione delle funzioni di vario utilizzo che le *software house* trattano nella loro attività, come ad esempio: servizi di consultazione tempo atmosferico, servizi di geolocalizzazione **IP**, servizi di localizzazione **GPS**, servizi di traduzione, servizi finanziari ecc., tutti offerti sia a pagamento che nella filosofia *freeware*.

La comodità di questi sistemi è che si basano su interfacce semplici e standard. In pratica non occorre sapere come implementano le specifiche, ma semplicemente richiamare metodi con i loro parametri come indicato dai loro descrittori, e mediante l'utilizzo di protocolli compatibili con tutti i sistemi operativi, oltre che dai browser web e dai linguaggi di programmazione. Grazie a queste caratteristiche, la compatibilità fra

un *client* utilizzatore di un determinato servizio con un server sul quale è installato il *web service*, è garantita dai protocolli “aperti”, standard e predefiniti.

Un'altra peculiarità positiva di questi *service* è che sono “auto-descrittivi” ed “auto-contenuti”, ovvero è possibile comprendere quali funzioni sono disponibili ed anche come usarle: in pratica è facile conoscere quali sono i parametri d'ingresso e quali valori sono restituiti oltre ai tipi di valori accettati.

Un'altra qualità di questi oggetti è che sono ricercabili consultando l'**UDDI (Universal Description Discovery and Integration)**[17]. Queste *web directory* visualizzano liste di indirizzi **URL** corrispondenti solitamente ai descrittori **WSDL** e sono suddivise in base ai domini di utilità.

4.1 DESCRIZIONE E FUNZIONAMENTO GENERALE

I *web service* permettono una comunicazione fra un *client* ed il servizio offerto tramite il protocollo **HTTP**, tuttavia nel dettaglio utilizza anche altri standard basati tutti sul formato **XML**, tra cui:

- **XML Schema**
- **UDDI (Universal Description Discoverey and Integration)**
- **SOAP (Simple Object Access Protocol)**
- **WSDL (Web Service Description Language)**

E' importante notare che mediante l'uso congiunto di **XML-HTTP** è possibile far funzionare i servizi anche fra piattaforme e sistemi operativi differenti: *Linux, Unix, Windows, MAC OS, Solaris* ecc..

Tramite **XML Schema**, così come fa un documento **DTD** nella costruzione di pagine **web**, si definisce la costruzione legale di un documento **XML** in questo caso per l'auto-descrizione del **service** e precisamente imposta le seguenti caratteristiche: quali elementi possono apparire (**tag**), quali attributi possiede ogni elemento, quali e quanti elementi **child** sono presenti, se e quando un **element** può essere vuoto o contenere solo testo, quali sono i tipi degli elementi e degli attributi ammessi e quali valori di default inserire. Riguardo il **WSDL** esso è il linguaggio che descrive dove si trova il **web service** e quali operazioni espone tale servizio. All'interno di un file di questo linguaggio ci sono 4 elementi fondamentali che sinteticamente hanno i seguenti significati:

- ◆ **<types>**
 - ◆ Tipi di dato accettabili come parametro sia in entrata che in uscita
- ◆ **<message>**
 - ◆ Definisce gli elementi che compongono i dati di un'operazione. Ogni messaggio è formato da una o più parti e si possono paragonare alle chiamate di funzioni di un linguaggio di programmazione
- ◆ **<portType>**
 - ◆ Questo è uno degli elementi più salienti perché descrive il servizio, le operazioni eseguibili ed i messaggi coinvolti nelle operazioni. In pratica questo **tag** mostra similmente ciò che è una libreria di funzioni oppure un modulo in un linguaggio di programmazione
- ◆ **<binding>**
 - ◆ Il **binding** definisce il formato di ogni messaggio ed il collegamento di ognuno con il protocollo **SOAP** specificando ogni parametro di input ed output

4.2 ESEMPIO STRUTTURA FILE WSDL

listato XML 5 – esempio file descrittore WSDL 1° parte

```
<definitions>
  <types>
    definizione dei tipi....
  </types>
  <message>
    definizione di un messaggio....
  </message>
  <portType>
    definizione di un port....
  </portType>
  <binding>
    definifinizione di un binding...
  </binding>
</definitions>
```

DETTAGLI PER I 2 ELEMENTI PIU' RILEVANTI

listato XML 6 – esempio file descrittore WSDL 2° parte

Esempio di WSDL

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

Nell'esempio del **listato XML 6** si vede come l'elemento *message* definisce “*glossaryTerm*” come nome del *port* e “*getTerm*” come nome dell'operazione. L'operazione è composta da due parti: una di *input* che è “*getTermRequest*” ed una di

output che si incarica della risposta che nell'esempio è *“getTermResponse”*. Gli elementi **<message>**, come si può notare, definiscono precisamente le parti singole, come il nome ed il tipo di dato delle funzioni inserite nel **<portType>**.

Nella struttura di un documento **WSDL** si possono inserire inoltre altri elementi più complessi che consentono di creare dei tipi di dato definiti dall'utente.

Riguardo il protocollo **SOAP** esso è fondamentale per scambiare le informazioni tra il *web service* ed il *client*, ossia le applicazioni utente che vi accedono. Fondamentalmente questo protocollo è il mezzo con cui è possibile comunicare fra applicazioni eseguite su differenti sistemi operativi, create con differenti tecnologie o linguaggi di programmazione, ma sempre attraverso **HTTP** e **l'XML**.

Dettagliatamente un messaggio **SOAP** è un documento **XML** costituito da 4 parti principali:

- **Envelope** che identifica il tipo di documento (**SOAP**)
- Un **HEADER** opzionale contenente delle mirate informazioni per una singola applicazione, oppure più indirizzi di destinatari se nella comunicazione dovessero esserci più punti d'arrivo
- Un **Body** che è indispensabile poiché contiene i dati di scambio per le *request* e i *response*
- **Fault** anch'esso opzionale poiché usato per fornire eventuali informazioni d'errore avvenuto nello scambio dati

Le importanti regole per la costruzione di un messaggio **SOAP** sono quindi: specificare la codifica **XML**, usare il **SOAP envelope namespace**, usare il **SOAP encoding**

namespace ed infine non inserire alcun collegamento ad un file **DTD**, né inserire istruzioni per il processore **XML**.

Quello più interessante è senza dubbio il campo **Body**, dato che in esso è contenuto il messaggio che verrà consegnato al destinatario. All'interno di esso ogni elemento deve essere dichiarato con un *namespace* ed è previsto soltanto un elemento predefinito: il **Fault**. Tutti gli elementi presenti nel *tag body* sono dunque quelli definiti dagli utenti e visibili nel file **WSDL** i quali si occupano di inviare parametri relativi ai metodi e di ricevere risposte come nell'esempio seguente:

figura 6.a

```
<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body xmlns:tns="http://www.unsitoqualunque.it/un_po_sui_Web_services">
    <tns:getUserFunction SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope/">
      <id xsi:type="xsd:integer">24</id>
    </tns:getUserFunction>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In figura 6.a si può notare infatti che nel *tag Envelope* sono stati caricati 3 *namespace*: uno per **XML-Schema** istanza, uno per **XMLSchema** ed uno per **SOAP** e tutti occorrono a descrivere a fondo il documento. Nel *tag Body* si può invece vedere che viene inserito il *namespace* costruito sulla base del **WSDL** e dei tipi di dati personalizzati: in particolare "http://www.unsitoqualunque.it/un_po_sui_Web_services".

Questo *namespace* fa sì che quando il messaggio giunge al *web service*, oppure viceversa verso il *client*, tutti gli elementi possano essere correttamente decodificati e si comprende anche che vengono inviati i messaggi corrispondenti alle operazioni, cioè le funzioni che erano state scritte anche nel **<portType>** del **WSDL**: nell'esempio *getUserFunction*.

Oltre la struttura di un *web service* disponibile, occorre indicare anche a quale indirizzo fisico inviare la richiesta per utilizzarlo, e come collegare il **SOAP** con **HTTP** per raggiungere il servizio richiesto. Per il primo obiettivo è necessario indicare nel file **WSDL** all'interno del *tag* **<service>** e con l'attributo *location* l'esatta **URL** del servizio, quindi l'intero messaggio potrà essere incapsulato nel protocollo **HTTP** e giungere a destinazione dove il *web service* è perennemente in ascolto, il quale però per comprendere quale azione deve intraprendere necessita di ricevere un'intestazione specifica attraverso un campo *Header* nella *request web*.

4.3 I WEB SERVICE DI SPAGO4Q

Per poter accedere ai servizi disponibili della piattaforma *Spago4Q* per il monitoraggio della qualità dei prodotti è disponibile *SpagoBI SDK*, che comprende di una collezione di *web service*, *API* e *tags* in grado di far interagire gli utenti con il server *SpagoBI* dal quale attingere per le risorse.

Principalmente le classi sono disponibili per il linguaggio *Java*, tuttavia mediante altri *web service* con descrittori **WSDL** è possibile interfacciarsi con essi mediante altri linguaggi di sviluppo come il **C# .Net**, o in alternativa con alcuni *plug-in* per i browser come ad esempio *Firefox*, così come nel seguito della tesi vado a spiegare.

Lavorando con *Java* o altri linguaggi, ma necessariamente implementando la principale interfaccia chiamata *DocumentsService*, si possono usare delle funzioni molto utili come:

- Ricevere una lista di risorse (file **XML**) che un utente è abilitato alla visione
- Ricevere una lista di regole permesse sull'esecuzione di un documento di risorsa
- Ricevere una lista dei parametri di un documento di risorsa
- Ricevere una lista dei parametri consentiti nell'esecuzione di un file di risorsa
- Eseguire una risorsa ed ottenere la risposta in diversi formati come **XLS**, **PDF** ecc.

Inoltre permette le principali funzioni di manipolazione:

- Creazione di un nuovo documento
- Ricezione di un *template* di un documento esistente
- Aggiornamento di un *template* esistente di un documento esistente

Esempio codice interfaccia DocumentsService:

```
package it.eng.spagobi.sdk.documents;

import it.eng.spagobi.sdk.documents.bo.SDKDocument;
import it.eng.spagobi.sdk.documents.bo.SDKDocumentParameter;
import it.eng.spagobi.sdk.documents.bo.SDKExecutedDocumentContent;
import it.eng.spagobi.sdk.documents.bo.SDKFunctionality;
import it.eng.spagobi.sdk.documents.bo.SDKTemplate;
import it.eng.spagobi.sdk.exceptions.InvalidParameterValue;
import it.eng.spagobi.sdk.exceptions.MissingParameterValue;
import it.eng.spagobi.sdk.exceptions.NonExecutableDocumentException;
import it.eng.spagobi.sdk.exceptions.NotAllowedOperationException;

import java.util.HashMap;

public interface DocumentsService {
```

```

        SDKDocument[] getDocumentsAsList(String type, String state,
String folderPath);

        SDKFunctionality getDocumentsAsTree(String initialPath);

        String[] getCorrectRolesForExecution(Integer documentId)
throws NonExecutableDocumentException;

        SDKDocumentParameter[] getDocumentParameters(Integer
documentId, String roleName) throws NonExecutableDocumentException;

        HashMap<String, String> getAdmissibleValues(Integer
documentParameterId, String roleName) throws
NonExecutableDocumentException;

        SDKExecutedDocumentContent executeDocument(SDKDocument
document, SDKDocumentParameter[] parameters, String roleName, String
outputType) throws NonExecutableDocumentException,
NotAllowedOperationException, InvalidParameterValue,
MissingParameterValue;

        SDKTemplate downloadTemplate(Integer documentId) throws
NotAllowedOperationException;

        void uploadTemplate(Integer documentId, SDKTemplate template)
throws NotAllowedOperationException;

        Integer saveNewDocument(SDKDocument document, SDKTemplate
template, Integer functionalityId) throws
NotAllowedOperationException;
    }

```

Anche a prima vista si può dedurre che l'interfaccia è abbastanza intuitiva oltre che discretamente semplice da implementare, in quanto le classi disponibili sono conformi e funzionali.

Per accedere alle risorse nel formato nativo in **XML** con linguaggi diversi da *Java*, ad esempio come nel mio caso in cui ho utilizzato il linguaggio di *.Net C#*, è invece più opportuno collegarsi ai *web service* descritti con **WSDL** al quale ci si può interfacciare con **SOAP** incapsulato in **HTTP**.

In questo modo è facile richiedere le stesse funzioni di quelle ottenibili in *Java*, ma con strumenti diversi dato che ormai quasi tutti i linguaggi gestiscono pienamente con opportune librerie il protocollo **SOAP** e generano applicazioni *web* o di tipo **Windows**

Forms che creino in automatico o con poche aggiunte le richieste web corredate dagli **Headers** prestabiliti e con le chiamate dei metodi contenuti nel **WSDL**.

Attraverso questa tecnica è dunque possibile leggere e/o scaricare i file **XML** disponibili sul server, inviando le esatte credenziali e successivamente manipolare tali file per una visualizzazione *customizzabile*, dato che anche le classi per il trattamento dei testi in **XML** ormai proliferano.

All'indirizzo <http://spago4q.org/Spago4Q/services> gli sviluppatori di *SpagoWorld* mettono a disposizione la consultazione e le **URL** dei vari *web service* con i sorgenti **WSDL** e le principali operazioni che ognuno di essi può eseguire, pertanto così come ho potuto personalmente sperimentare questo sistema è un ottimo ausilio per creare, manipolare e visualizzare i dati che la piattaforma ha estratto nei processi di misurazione.

4.4 CREAZIONE WEB REQUEST HTTP

Nel dettaglio per adoperare tutti i metodi prima descritti è necessario seguire una serie di passaggi implementativi così com'è elencato:

- Stabilire una connessione **HTTP** inoltrando una richiesta web con un *header* impostato con tutti i necessari parametri
- Nel **body** della richiesta web occorre inviare un documento strutturato con i vari parametri ed in formato **XML**
- Richiedere il servizio denominato *getDocumentsAsList* il quale restituisce come risposta un file **XML** contenente l'elenco di tutti i documenti disponibili, di quale tipologia appartengono e quali identificativi univoci possiedono

- Richiedere il servizio *getDocumentParameters* il quale passandogli come parametri un **ID** e gli identificativi utente, estrapolati dalla risposta del metodo precedente, ci restituisce una lista di parametri necessaria per l'esecuzione di un documento risorsa
- Richiedere il servizio *getAdmissibleValues* che in ingresso necessita del parametro **ID** fornito dal precedente metodo ed ancora dell'identificativo ruolo utente, così da ottenere in risposta una lista di valori che ogni parametro può assumere
- Richiedere infine il servizio *executeDocument* che raccoglie nella sua costruzione tutti i dati forniti dai precedenti servizi. Questo metodo richiede 4 parametri d'ingresso che sono: l'**ID** di un documento ricavato dal metodo *getDocumentsAsList*, l'*array* di parametri ricavati dai metodi *getDocumentParameters* (per conoscere quali parametri usare) e *getAdmissibleValues* (per conoscere quali valori possono assumere i parametri), la stringa identificativa del ruolo utente e per ultimo la stringa che determina in quale formato eseguire il documento, ad esempio per ricevere una risorsa in **XML**, **PDF**, **HTML** ecc.. La risposta di questo servizio è quella finale e fondamentale perché contiene esattamente tutti i valori misurati corredati dei rispetti significati, attributi, soglie di riferimento e tutto ciò che indica i livelli di qualità globali e parziali da passare poi ad un *parser* per la successiva visualizzazione.

Esempio di richiesta del servizio executeDocument passato come body alla web

request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<soapenv:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
```

```

<wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-27">
  <wsse:Username>s4quser</wsse:Username>
  <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">s4quser</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
  <ns1:executeDocument xmlns:ns1="urn:spagobisdskdocuments"
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <in0 href="#id0"/>
    <in1 xmlns:ns2="http://bo.documents.sdk.spagobi.eng.it"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" soapenc:arrayType="ns2:SDKDocumentParameter[4]"
  xsi:type="soapenc:Array">
      <in1 href="#id1"/>
      <in1 href="#id2"/>
      <in1 href="#id3"/>
      <in1 href="#id4"/>
    </in1>
    <in2 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:string">/spagobi/user</in2>
    <in3 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:string">XML</in3>
  </ns1:executeDocument>
  <multiRef xmlns:ns3="http://bo.documents.sdk.spagobi.eng.it"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id1" soapenc:root="0"
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns3:SDKDocumentParameter">
    <constraints soapenc:arrayType="ns3:SDKConstraint[0]" xsi:type="soapenc:Array"/>
    <id href="#id5"/>
    <label xsi:type="soapenc:string">resource</label>
    <type xsi:nil="true" xsi:type="soapenc:string"/>
    <urlName xsi:type="soapenc:string">ParKpiResources</urlName>
    <values soapenc:arrayType="soapenc:string[1]" xsi:type="soapenc:Array">
    <values xsi:type="soapenc:string">5</values>
  </values>
</multiRef>

  <multiRef xmlns:ns4="http://bo.documents.sdk.spagobi.eng.it"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id3" soapenc:root="0"
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns4:SDKDocumentParameter">
    <constraints soapenc:arrayType="ns4:SDKConstraint[0]" xsi:type="soapenc:Array"/>
    <id href="#id6"/>
    <label xsi:type="soapenc:string">register values</label>
    <type xsi:nil="true" xsi:type="soapenc:string"/>
    <urlName xsi:type="soapenc:string">register_values</urlName>
    <values soapenc:arrayType="soapenc:string[1]" xsi:type="soapenc:Array">
    <values xsi:type="soapenc:string">>false</values>
  </values>
</multiRef>

  <multiRef xmlns:ns5="http://bo.documents.sdk.spagobi.eng.it"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id0" soapenc:root="0"
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns5:SDKDocument">
    <dataSetId xsi:nil="true" xsi:type="soapenc:int"/>
    <dataSourceId xsi:nil="true" xsi:type="soapenc:int"/>
    <description xsi:type="soapenc:string">QEST Case Study for Application Management Service</description>
    <engineId href="#id7"/>
    <id href="#id8"/>
    <label xsi:type="soapenc:string">QEST-AM-SERVICE</label>
    <name xsi:type="soapenc:string">QEST-AM-SERVICE</name>
    <state xsi:type="soapenc:string">REL</state>
    <type xsi:type="soapenc:string">"KPI"</type>
  </multiRef>

```

```

<multiRef xmlns:ns6="http://bo.documents.sdk.spagobi.eng.it"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id4" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns6:SDKDocumentParameter">
<constraints soapenc:arrayType="ns6:SDKConstraint[0]" xsi:type="soapenc:Array"/>
<id href="#id9"/>
<label xsi:type="soapenc:string">behaviour</label>
<type xsi:nil="true" xsi:type="soapenc:string"/>
<urlName xsi:type="soapenc:string">behaviour</urlName>
<values soapenc:arrayType="soapenc:string[1]" xsi:type="soapenc:Array">
<values xsi:type="soapenc:string">Default</values>
</values>
</multiRef>

<multiRef xmlns:ns7="http://bo.documents.sdk.spagobi.eng.it"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id2" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns7:SDKDocumentParameter">
<constraints soapenc:arrayType="ns7:SDKConstraint[0]" xsi:type="soapenc:Array"/>
<id href="#id10"/>
<label xsi:type="soapenc:string">date</label>
<type xsi:nil="true" xsi:type="soapenc:string"/>
<urlName xsi:type="soapenc:string">ParKpiDate</urlName>
<values soapenc:arrayType="soapenc:string[1]" xsi:type="soapenc:Array">
<values xsi:type="soapenc:string"/>
</values>
</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id6" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">49</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id10" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">48</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id5" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">47</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id8" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">14</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id7" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">6</multiRef>

<multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id9" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int">50</multiRef>

</soapenv:Body>
</soapenv:Envelope>

```

La parte di colore **fucsia** è relativa all'oggetto Documento. Questa parte viene ricavata tramite il servizio **getDocumentAsList**.

La parte di colore **verde** è relativa all'oggetto Parametro. Questa parte viene ricavata tramite l'esecuzione del servizio **getDocumentParameters**. La parte della busta **Values** viene ricavata tramite il servizio **getAdmissibleValues**.

La parte **azzurra** è relativa al terzo parametro (Stringa che rappresenta il ruolo dell'utente ("/spagobi/user")).

La parte **gialla** è relativa al quarto parametro (Stringa che rappresenta il formato con cui verrà salvato il documento ("XML")).

NB. Esempio fornitomi gentilmente da **Engineering Group**.

Capitolo 5

Spago4Q .NET Dashboard (soluzione SW creata)

Le funzionalità del software creato come progetto di tesi in collaborazione con **Engineering Group** e la supervisione del **Dr. Oltolina** oltre che del **Dr. Frati** ricercatore del **Polo di Crema**, permettono di accedere facilmente alle risorse di analisi della piattaforma **Spago4Q** e di mostrarne il loro contenuto in forma grafica e testuale sul computer dell'utente.

In questo modo si risolve efficacemente il problema di dover stabilire con esattezza la qualità di un prodotto di natura software in fase di sviluppo, tutto in modo *user-friendly* su un qualsiasi **PC** dotato di sistema operativo **Windows** con piattaforma **.Net** e la relativa **CLR Common Language Runtime**.

Questo strumento modesto e *freeware* può integrarsi ed aggiungersi all'ambiente **SpagoWorld** che già possiede numerose e utili applicazioni, pertanto la base primaria del mio lavoro sono le interfacce che i server **Spago4Q** offrono, mentre tutto il meccanismo di collegamento, reperimento, estrazione e visualizzazione è stato sviluppato secondo il paradigma della **OOP (Object Oriented Programming)** scrivendo le opportune classi.

Pur se con **.Net** sarebbe stato possibile sviluppare un applicativo eseguibile su macchine con **Operating System** di diverso genere, a patto ovviamente che possiedano le librerie di tale *framework*, questa utility è stata progettata e costruita per l'esecuzione in ambiente **MS Windows** con una grafica di tipo **Windows Forms** e con caratteristiche tipo *dashboard*, seguendo l'intenzione originaria di generare dei *report* completi con

poche difficoltà di consultazione e poche risorse hardware necessarie, a parte un'indispensabile connessione Internet con un'ampiezza di banda non per forza ampia.

5.1 CARATTERISTICHE DASHBOARD E REPORT WINDOWS

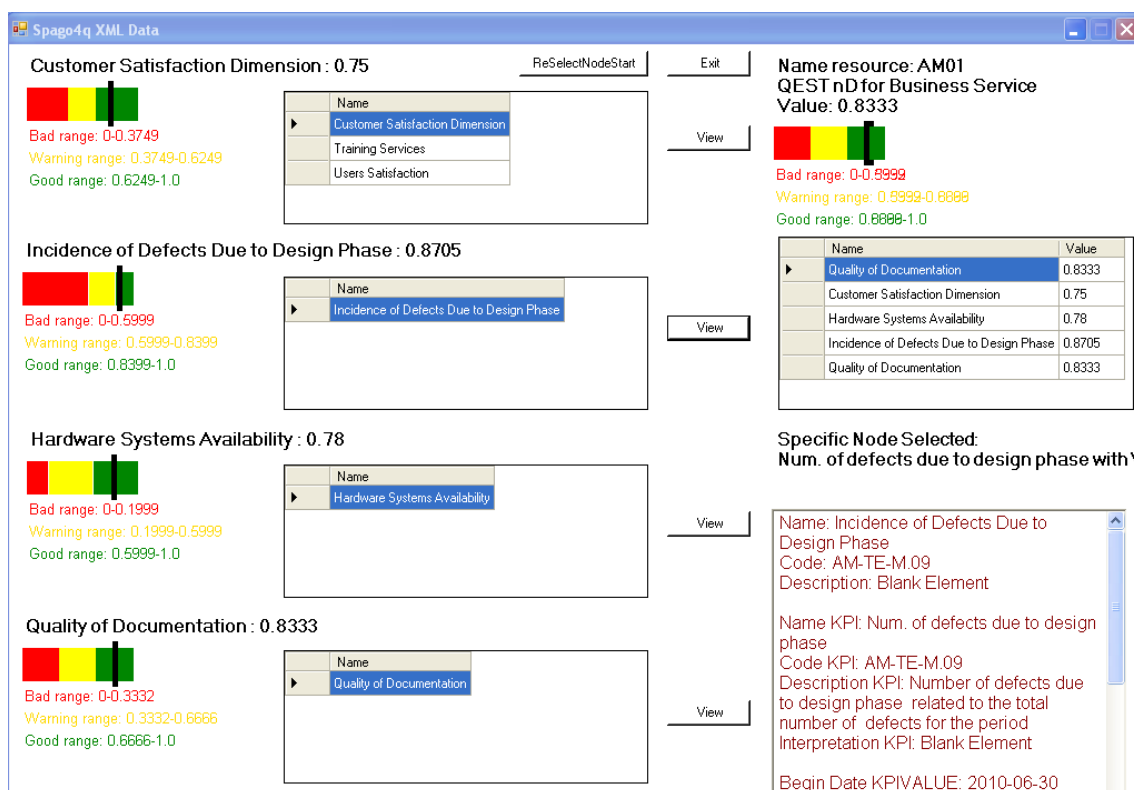
Nel progetto software di tesi si vuole costruire una visualizzazione automatica, eseguibile possibilmente con pochi comandi e che non fosse estremamente sofisticata da assorbire troppe risorse grafiche.

In **OS Windows**, come in tutti i sistemi operativi ad interfaccia grafica, di solito per ottenere tali peculiarità si adoperano i cosiddetti *dashboard*, i quali sono dei piccoli dispositivi grafici visibili sul desktop fin dall'avvio del sistema dai quali è possibile eseguire altri programmi chiamati *widget*, che a loro volta servono per applicazioni modeste caricate in memoria ed eseguite fin dall'avvio del PC, ma che appaiono silenti per non ingombrare costantemente lo schermo come ad esempio: l'orologio avanzato, la calcolatrice, le previsioni meteo o le *news*, ecc..

I *report* invece si occupano esclusivamente di mostrare in forma testuale e/o grafica i valori dei dati selezionati, pertanto possono unirsi ad una *dashboard* qualora sia dispendioso avere sul monitor una *Windows Forms* sempre *on-top*.

Nell'utility creata si è preferito di unire le due proprietà con una schermata discretamente semplice, abbastanza ampia per una lettura non stancante, ma eseguibile e ridimensionabile come una comune applicazione, che tuttavia nelle versioni future potrà con delle opportune migliorie disporre di funzioni per *dashboard* più avanzate.

Figura 2 – Schermata principale Spago4Q .Net Dashboard



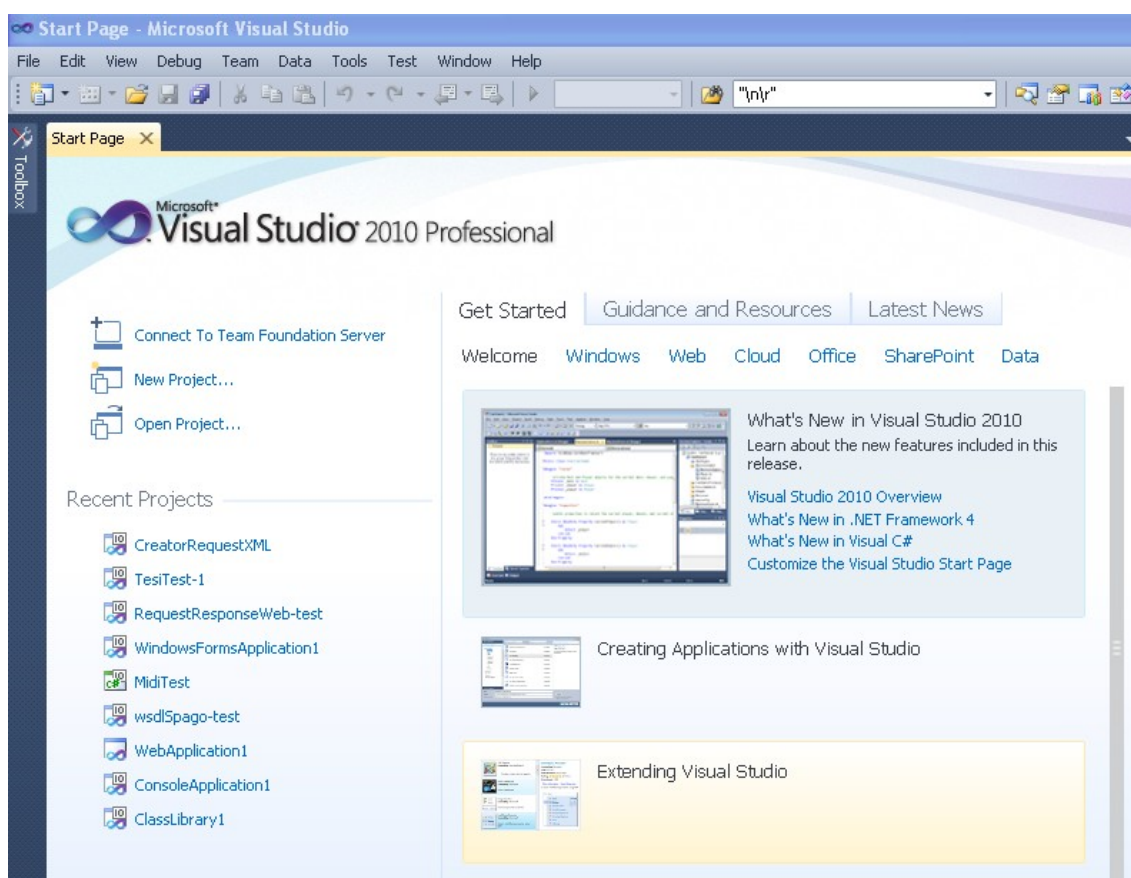
5.2 TOOLS DI SVILUPPO UTILIZZATI

Lo strumento impiegato per l'implementazione del software è stato prettamente il **Microsoft Visual Studio 2010 PRO .Net** con cui è stato possibile disegnare efficientemente ed assemblare la parte grafica inserendo gli opportuni oggetti: *textbox*, *buttons*, *label* e le loro proprietà di dimensioni e aspetto.

Riguardo alla stesura del codice, per permette la piena compatibilità con il sistema operativo **Windows** è stato utilizzato il linguaggio **C# di .Net[18]** con il paradigma della **OOP**.

Le librerie adoperate sono tutte quelle comprese nel *framework* e costruite principalmente per il trattamento dei file **XML** oltre che per l'invio delle richieste web con il protocollo **HTTP**.

Figura 3 – Esecuzione Microsoft Visual Studio 2010 Pro



5.3 SPECIFICHE RICHIESTE

Per la proposta di tesi di **Engineering Group** con la finalità di integrare al meglio l'applicazione **MS Windows** nell'ambiente *SpagoWorld* le specifiche più importanti richieste sono:

- Collegamento automatico al *web service* preposto
- Selezione del documento di risorsa appropriato fra una lista di quelli disponibili per uno user account

- Download in locale del documento di risorsa selezionato in formato **XML** per un'elaborazione da effettuare in *off-line*
- Pulizia e normalizzazione del file scaricato soggetto a possibili inserimenti di parti aggiuntive non conformi ad un formato **XML** corretto
- Analisi del documento ed estrazione dei nodi elemento e dei relativi attributi con tutti i rispettivi valori
- Personalizzazione della visualizzazione principale mediante la scelta dei nodi primari da cui mostrare i sotto-nodi
- Permettere la *view* dei dettagli di ogni singolo nodo secondo selezione con eventi mouse-click
- Consentire una rifelezione degli elementi principali da prendere in considerazione
- Consentire la scelta di documenti differenti disponibili sul *web service*

5.4 ALGORITMO DI ESTRAZIONE ED ANALISI XML

Fra le funzioni più importanti del programma una è certamente quella di estrapolazione dei dati contenuti nelle risorse in formato **XML** e per questo è stato implementato un semplice algoritmo di scansione dell'albero **XML** contenuto nel file.

Lo pseudo-codice dell'algoritmo implementato è il seguente:

- Istanziare oggetto *readerXML*
- Caricare in memoria il file **XML**
- Finché ci sono nodi da leggere avanzare nella lettura

- Se un nodo letto è un elemento con nome "**MODELINSTANCENODE**" ed è anche un elemento di partenza spostarsi all'attributo specificato
- Se l'attributo ha un certo valore indicato posizionare il *reader* sull'elemento a cui appartiene l'attributo
- Assegnare ad una variabile il valore dell'elemento su cui il *reader* è posizionato
- Finché ci sono attributi per l'elemento spostarsi su di essi
- Assegnare una variabile che contenga il nome attributo
- Assegnare una variabile che contenga il valore attributo
- Se il nome dell'attributo corrisponde ad uno di quelli ricercati nel file allora assegnare il suo nome e il suo valore alle variabili che poi serviranno alla visualizzazione
- Spostarsi all'elemento successivo e verificare il nome
- Ripetere il ciclo *while* per tutti i sotto-elementi dell'elemento "**MODELINSTANCENODE**"

In pratica questo algoritmo prende in esame ogni nodo padre "**MODELINSTANCENODE**" ed i suoi due nodi figli "**KPI**" e "**KPIVALUE**" che a loro volta contengono gli attributi di significato per le metriche, tuttavia il metodo progettato per una classe **ParsexmlDoc**, come nel seguito spiegata nei dettagli, può essere impostato per la lettura di altri diversi nodi cambiando semplicemente i parametri come si evince dal listato del codice sorgente sotto indicato.

```

    /// <summary>
    /// This method read a complete element MODELINSTANCENODE with two
    subelement kpi and kpivalue. This method also sets all variables to store each
    value.
    /// </summary>

```

```

    /// <param name="elemsearch">Name search item</param>
    /// <param name="valuesearch">Value name search item</param>
    public void ReadElemGlobal(string elemsearch, string valuesearch)
    {
        try
        {
            XmlTextReader xrd = new XmlTextReader(pathdocxml);
            //Creating an object for reading XML file
            XmlNodeType tn = xrd.NodeType;
            //Creating an object to determine XML node type

            while (xrd.Read())
            //Loop for reading all elements until the method is true
            {
                if (xrd.Name == "MODELINSTANCENODE" && xrd.IsStartElement())
            //Checks if the name element is given and if is an start element
                {
                    xrd.MoveToElement();
            //Moves to the element that contains the current attribute node
                    xrd.MoveToAttribute(elemsearch);
            //Moves to the attribute with the specified name
                    if (xrd.Value.ToString() == valuesearch)
            //Checks if the value of the attribute is given
                    {
                        xrd.MoveToElement();
                        string valuecoderoot = xrd.Value;
            //Assigned to the variable string attribute value

                        while (xrd.MoveToNextAttribute())
            //Read all the attributes of current element
                        {
                            //Assigned name and value attributes to variables
                            string nameattr = xrd.Name;
                            string valueattr = xrd.Value;
                            //analysis and extraction of each attribute
                            switch (nameattr)
                            {
                                case "name":
                                    name_node_global = "";
                                    name_node_global = valueattr;
                                    if (valueattr == "")
                                        name_node_global = "Blank Element";
                                    break;
                                case "code":
                                    code_node_global = "";
                                    code_node_global = valueattr;
                                    if (valueattr == "")
                                        code_node_global = "Blank Element";
                                    break;
                                case "description":
                                    description_node_global = "";
                                    description_node_global = valueattr;
                                    if (valueattr == "")
                                        description_node_global = "Blank
Element";
                                    break;
                                default:
                                    break;
                            }
                        }
                    }
                }
            }
        }
    }

```

```

xrd.MoveToElement();
//repeat analysis for the KPI item
if (xrd.ReadToFollowing("KPI"))
{
    while (xrd.MoveToNextAttribute() && xrd.Name !=
"MODELINSTANCENODE")
    {
        string nameattr2 = xrd.Name;
        string valueattr2 = xrd.Value;
        switch (nameattr2)
        {
            case "description":
                kpi_description_global = "";
                kpi_description_global = valueattr2;
                if (valueattr2 == "")
                    kpi_description_global = "Blank
Element";

                break;
            case "name":
                kpi_name_global = "";
                kpi_name_global = valueattr2;
                if (valueattr2 == "")
                    kpi_name_global = "Blank Element";
                break;
            case "interpretation":
                kpi_interpretation_global = "";
                kpi_interpretation_global =
valueattr2;

                if (valueattr2 == "")
                    kpi_interpretation_global = "Blank
Element";

                break;
            case "code":
                kpi_code_global = "";
                kpi_code_global = valueattr2;
                if (valueattr2 == "")
                    kpi_code_global = "Blank Element";
                break;
            default:
                break;
        }
    }

xrd.MoveToElement();
//repeat analysis for the KPIVALUE item
if (xrd.ReadToFollowing("KPIVALUE"))
{
    while (xrd.MoveToNextAttribute() && xrd.Name !=
"MODELINSTANCENODE")
    {
        string nameattr3 = xrd.Name;
        string valueattr3 = xrd.Value;
        switch (nameattr3)
        {
            case "begindate":
                kpivalue_bgdate_global = "";
                kpivalue_bgdate_global = valueattr3;

```

```

        if (valueattr3 == "")
            kpivaluedate_global = "Blank
Element";
        break;
    case "enddate":
        kpivaluedate_global = "";
        kpivaluedate_global = valueattr3;
        if (valueattr3 == "")
            kpivaluedate_global = "Blank
Element";
        break;
    case "target":
        kpivaluetarget_global = "";
        kpivaluetarget_global = valueattr3;
        if (valueattr3 == "")
            kpivaluetarget_global = "Blank
Element";
        break;
    case "description":
        kpivaluedescription_global = "";
        kpivaluedescription_global =
valueattr3;
        if (valueattr3 == "")
            kpivaluedescription_global =
"Blank Element";
        break;
    case "weight":
        kpivalueweight_global = "";
        kpivalueweight_global = valueattr3;
        if (valueattr3 == "")
            kpivalueweight_global = "Blank
Element";
        break;
    case "thresholdid":
        kpivaluethresholdid_global = "";
        kpivaluethresholdid_global =
valueattr3;
        if (valueattr3 == "")
            kpivaluethresholdid_global =
"Blank Element";
        break;
    case "weightedvalue":
        kpivalueweightedvalue_global = "";
        kpivalueweightedvalue_global =
valueattr3;
        if (valueattr3 == "")
            kpivalueweightedvalue_global =
"Blank Element";
        break;
    case "value":
        kpivaluevalue_global = "";
        kpivaluevalue_global = valueattr3;
        if (valueattr3 == "")
            kpivaluevalue_global = "Blank
Element";
        break;
    default:
        break;
}
}

```


gli oggetti delle classi opportune per poter poi richiamare i vari metodi con i relativi parametri al fine d'implementare ogni specifica: tutto questo principalmente potrà servire a comporre delle maschere complesse di *report* più complete.

In questo contesto inoltre una libreria dinamica può rappresentare un buon strumento per costruire integrazioni con altre soluzioni di scambio dati in rete, LAN o WAN che siano, ad esempio quella commerciale di **MS Sharepoint** molto diffusa in ambienti **Windows** che non sono **Open Source**, ma che possono così essere maggiormente personalizzate.

Nel dettaglio la classe **Parsexmldoc** che implementa l'interfaccia **Iparsexml**, al suo interno dispone di alcuni metodi che accettano i parametri che specificano il risultato dell'estrazione, sottolineando ulteriormente che tali metodi sono comodamente richiamabili aggiungendo un **reference** d'importazione della **DLL** in un qualunque progetto.

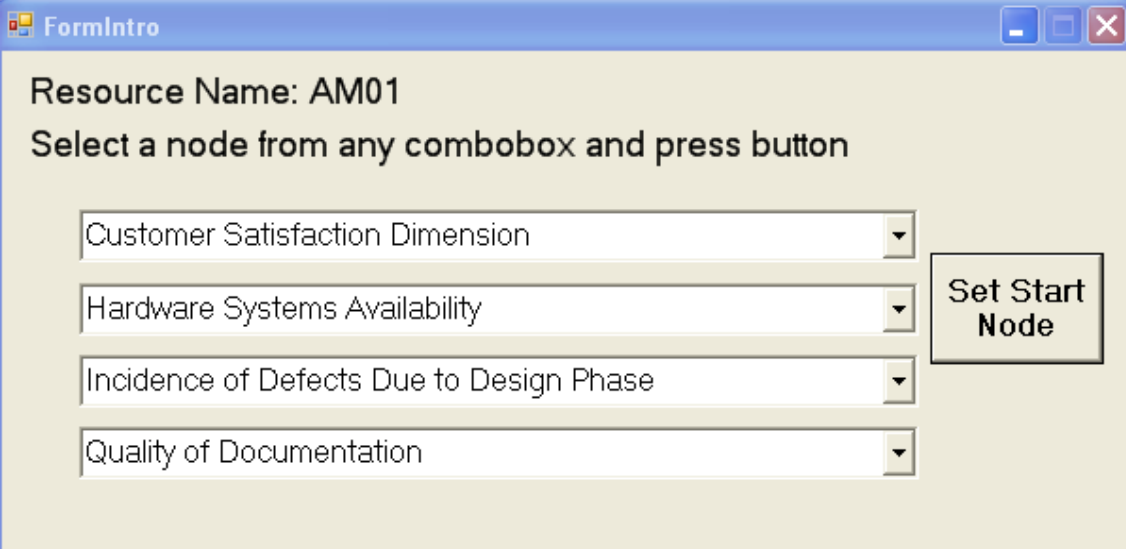
Capitolo 6

DESCRIZIONE TECNICA DETTAGLIATA

Per esporre ad un miglior livello la struttura ed il funzionamento del software, nel seguito vengono mostrate con l'aiuto del linguaggio **UML** le parti sorgenti e come interagiscono fra di loro.

La prima descrizione riguarda l'interfaccia utente principale, ed è utile a spiegare in modo pratico come svolge le funzioni per gli eventi generati.

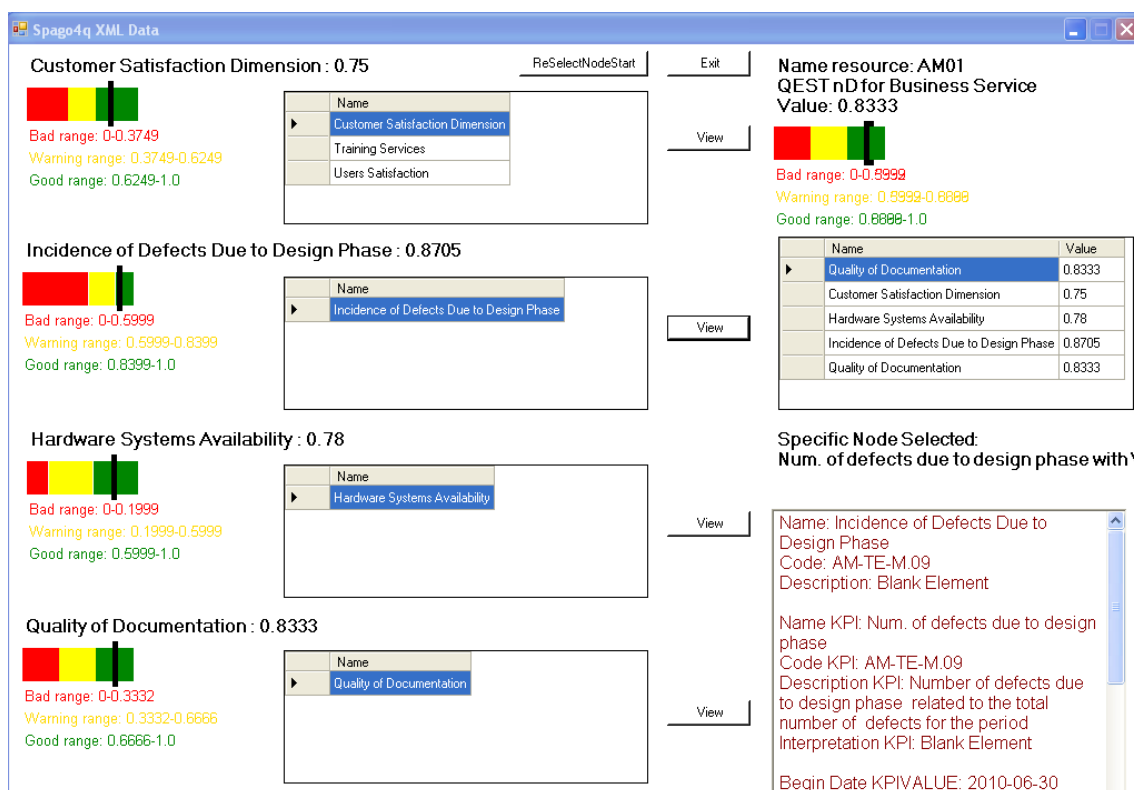
Figura 4 - Form d'avvio Spago4Q .Net Dashboard



The screenshot shows a window titled "FormIntro" with a light beige background. At the top left, it says "Resource Name: AM01". Below that, the instruction "Select a node from any combobox and press button" is displayed. There are four vertical comboboxes, each containing a text label and a downward-pointing arrow. The labels are: "Customer Satisfaction Dimension", "Hardware Systems Availability", "Incidence of Defects Due to Design Phase", and "Quality of Documentation". To the right of these comboboxes is a rectangular button with the text "Set Start Node".

All'avvio viene immediatamente estratto il nome del documento di risorsa e le prime 4 misure (nodi prescelti) che si vogliono mostrare nel dettaglio, e questa scelta si può personalizzare scegliendo da una lista di ogni **combobox** il valore desiderato. Successivamente alle scelte dei nodi e premendo il pulsante **"Set Start Node"**, verrà visualizzato sul desktop il pannello completo con tutti i dettagli ed i valori di misura.

Figura 5 – Schermata principale Spago4Q .Net Dashboard



In questa videata il pannello indica il valore globale (in alto a destra) con i nomi ed i valori di qualità raggiunti dai sotto-nodi primari scelti nella schermata precedente.

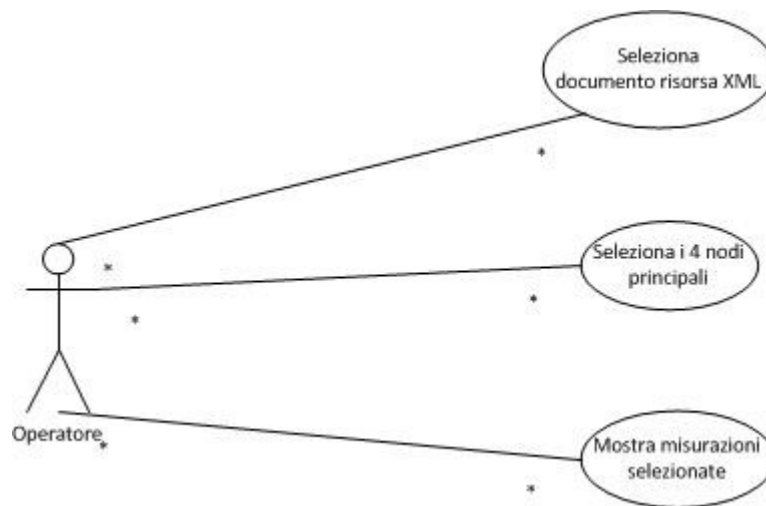
Per ogni sotto-nodo viene inoltre mostrato un indicatore grafico per una visione rapida, ma con ogni range di riferimento, inoltre compare la lista di tutti gli altri sotto-nodi dell'elemento.

Selezionando uno di questi elementi del *DataGridView* e premendo il **button** “View”, nella **textbox** presente in basso a destra si ottengono in forma testuale i dettagli più capillari.

E' anche possibile reimpostare tutta la schermata rifelezionando i nodi principali premendo il pulsante “ReSelectNodeStart” per ritornare alla **Form** iniziale, infine semplicemente premendo il pulsante “Exit” si termina l'esecuzione del programma”.

6.1 GRAFICI UML

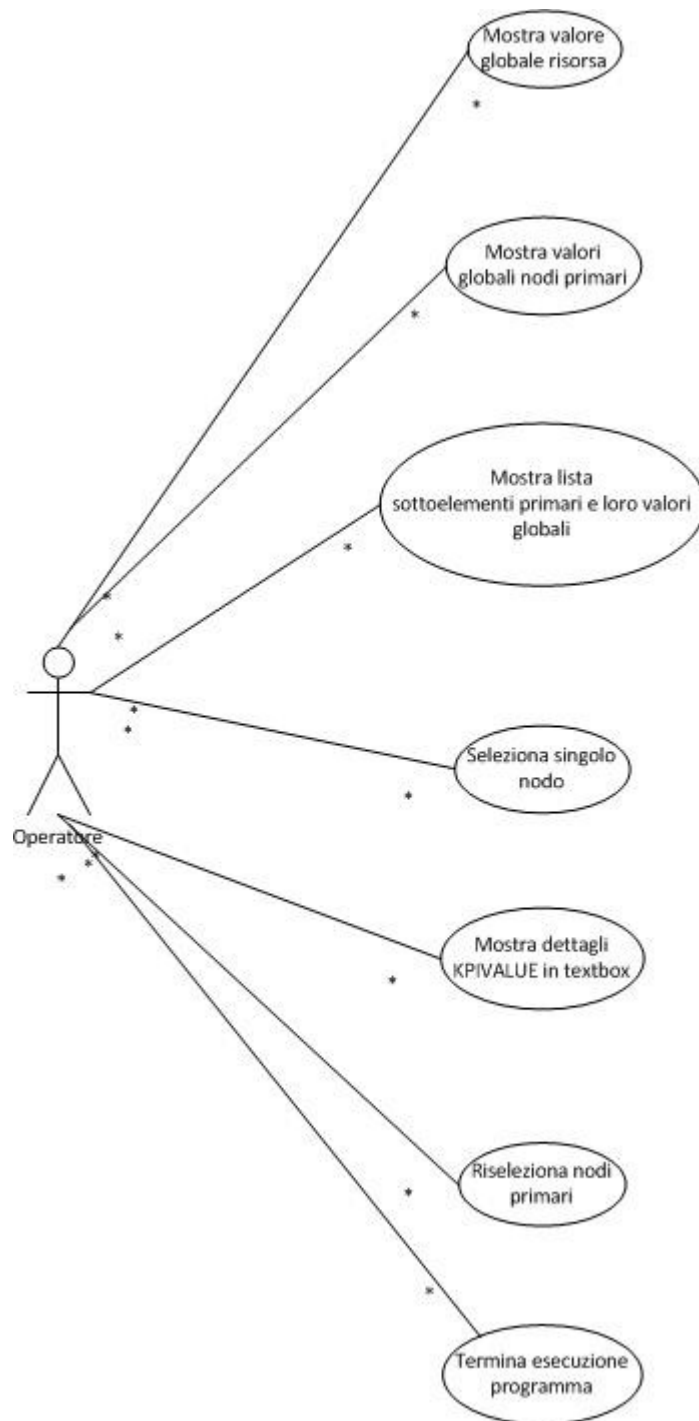
USE CASE FORM 1



Spiegazione grafico:

L'utente può compiere all'avvio del programma attraverso la **Form** principale le funzioni di: scelta e selezione di una risorsa disponibile fra quelle presenti sul server relative ai prodotti in costruzione, selezione di 4 elementi principali corrispondenti a 4 dimensioni da osservare, infine mostrare attraverso la **Form** che sarà lanciata successivamente i valori primari dei 4 nodi scelti mediante le **combobox**.

USE CASE FORM 2 (SCHERMATA PRINCIPALE)

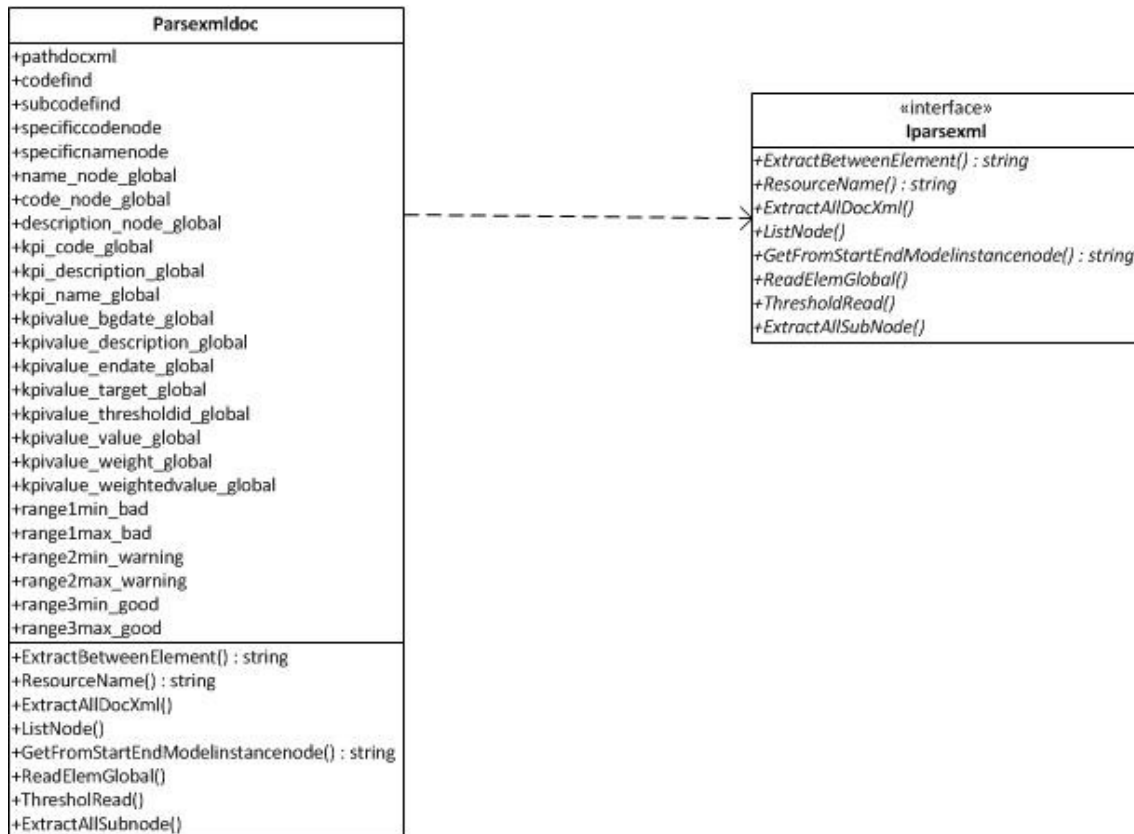


Nel diagramma relativo al **USE CASE FORM 2** sono elencate le funzioni disponibili nelle 2° videata del **SW**. Dopo aver selezionato sia la risorsa che i suoi principali nodi da esaminare come inizio, l'utente può accedere ad altre risorse come nel dettaglio seguente:

- Ogni documento contiene un valore globale stimato che è visualizzato nel pannello
- Vengono visualizzati i valori globali dei singoli nodi principali scelti
- Per ogni nodo è possibile elencare tutti i nodi discendenti da essi con i loro valori globali nella lista degli *items* delle *combobox*
- E' possibile selezionare un singolo nodo come *SelectedItem*
- E' possibile dopo la selezione singola di un nodo mostrare i dettagli del corrispondente sotto-nodo “**KPIVALUE**” all'interno di una *textbox* con i nomi e valore attributi
- E' possibile rizelezionare i nodi primari di partenza reimpostandoli attraverso il metodo *Show()* della 1° *Form*
- E' possibile terminare l'esecuzione del programma e dei sottoprocessi collegati

CLASS DIAGRAM

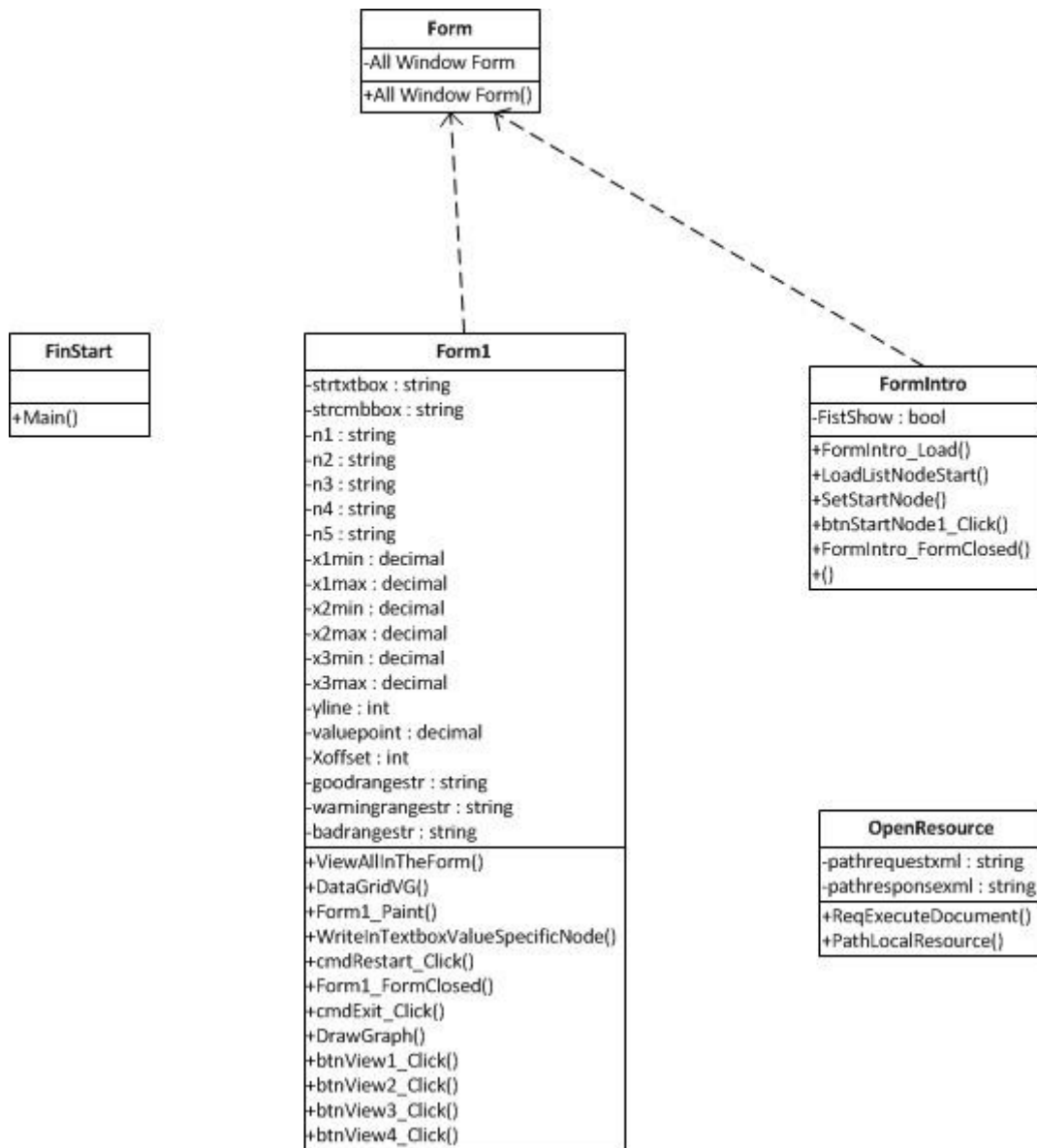
INTERFACCIA & CLASSE CHE LA IMPLEMENTA



Il **SW** è composto da una interfaccia principale **Iparsexml** che viene implementata da una classe **Parsexml doc** la quale mette a disposizione i metodi e le funzioni più importanti anche per la **DLL**.

I metodi si occupano esclusivamente dell'analisi del documento **XML** di risorsa e dell'estrazione di alcuni precisi nodi con attributi.

CLASSI AGGIUNTIVE



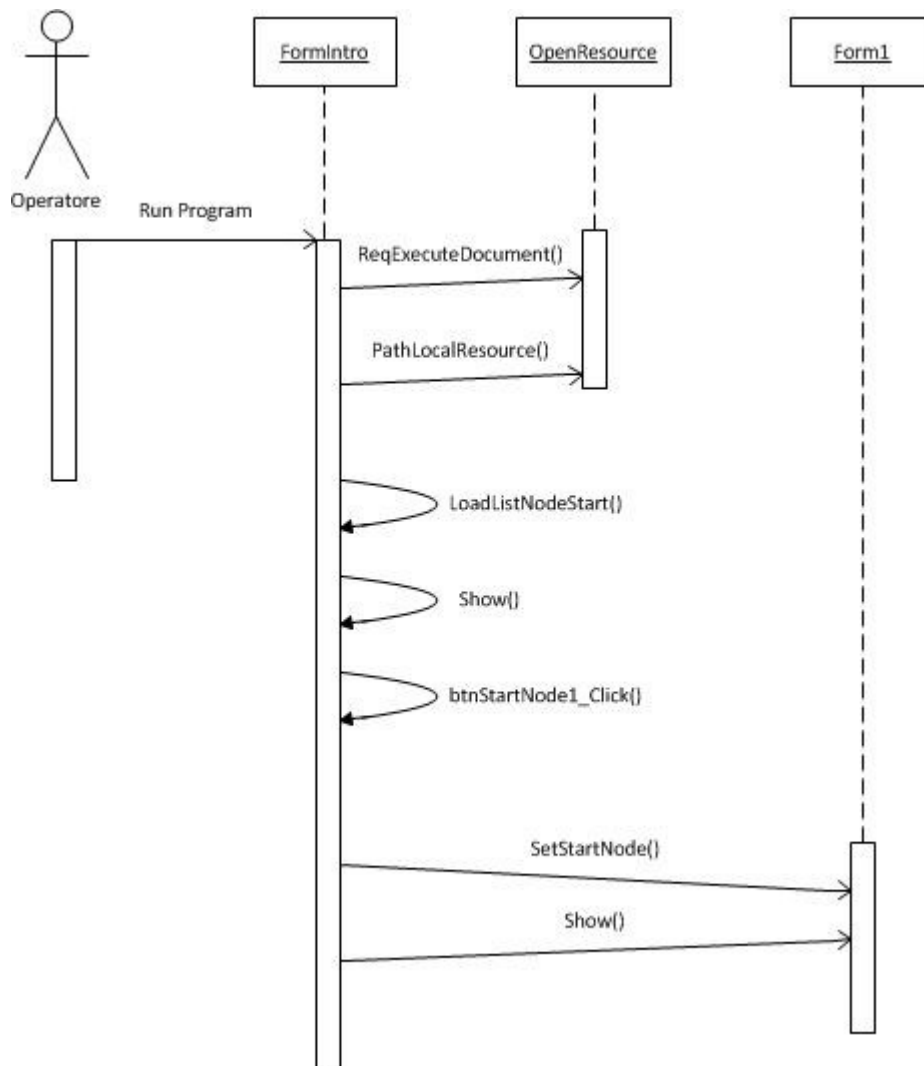
Il **SW** è composto da altre classi dedicate particolarmente all'applicazione **Windows Form**. La classe padre **Form** generata direttamente da **VS** viene derivata dalle 3 sottoclassi che creano l'interfaccia utente grafica. La classe **FinStart** si occupa di

stabilire quale schermata con relativa classe eseguire all'avvio ed è questa una escamotage tipica per la modifica dei parametri dei *default* d'avvio.

La classe **FormIntro** è relativa allo **use case 1** che permette all'utente di selezionare la risorsa e i nodi primari d'analisi, mentre la classe **Form1** si occupa della creazione del pannello principale con tutte le specifiche dello **use case 2**.

La classe **OpenResource** si incarica della connessione al *web service* e del download della risorsa da utilizzare. Attualmente i parametri di selezione file e delle **URL**, sia del server che di memorizzazione in locale sono predefiniti, pertanto sarà oggetto di un miglioramento in tempi abbastanza rapidi al fine di consentire una multiselezione dei documenti **XML** oltre che l'impostazione di differenti percorsi remoti e locali.

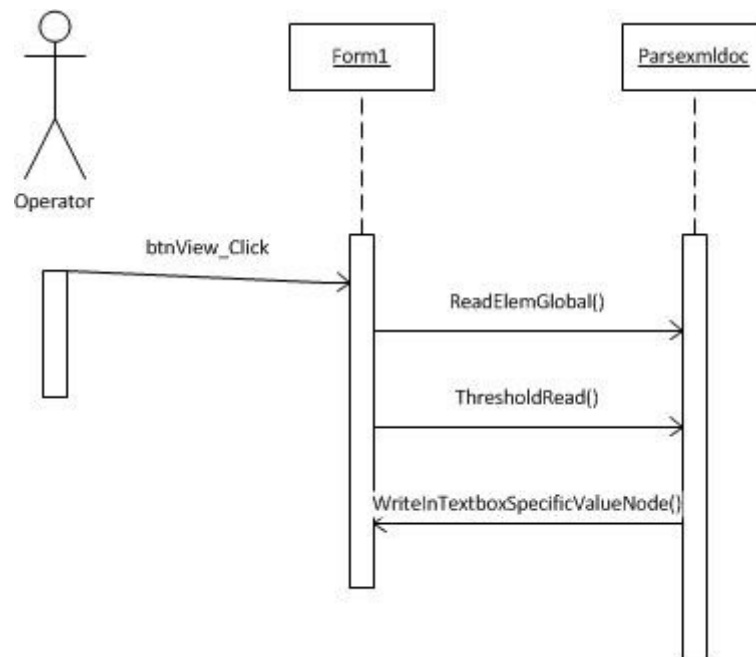
SEQUENCE DIAGRAM (Avvio e selezione elementi di partenza)



La sequenza di evoluzione del programma ha inizio con la classe **FormIntro** che richiama il metodo **ReqExecuteDocument()** in grado di *downloadare* la risorsa mediante la classe **OpenResource**. Questa classe permette inoltre di riempire gli *ArrayList* e creare così le liste contenenti tutti gli elementi estratti dal file **XML**, quindi

procede ad impostare 4 nodi fra questi per una visualizzazione iniziale che sarà ottenuta richiamando la classe **Form1**.

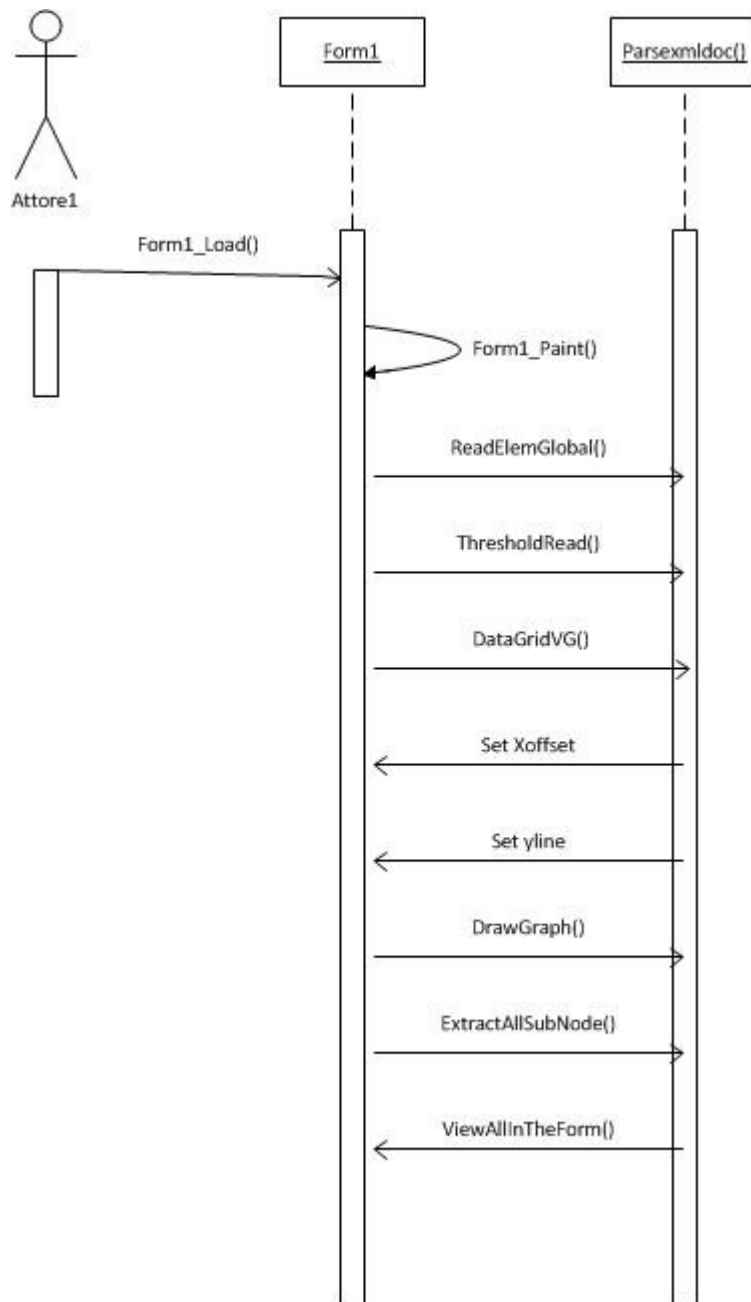
SEQUENCE DIAGRAM (selezione singolo nodo)



Riguardo un singolo nodo mostrato nelle liste delle *combobox* della classe **Form1**, è possibile ottenere su di esso maggiore informazioni richiamando prima il metodo **ReadElemGlobal()** per l'estrazione dati: nodo, attributi, valori del nodo dall'albero XML e poi il metodo **ThresholdRead()** che ricerca le soglie definite per i valori trovati ed entrambi i 2 metodi fanno parte della classe **Parsexml doc**.

Successivamente dalla classe **Parsexml doc** viene richiamato il metodo **WriteInTextboxSpecificValueNode()** della classe **Form1** per mostrare in forma testuale i dati ricavati e scambiati tra le *form* attraverso variabili statiche.

SEQUENCE DIAGRAM (view schermata principale)



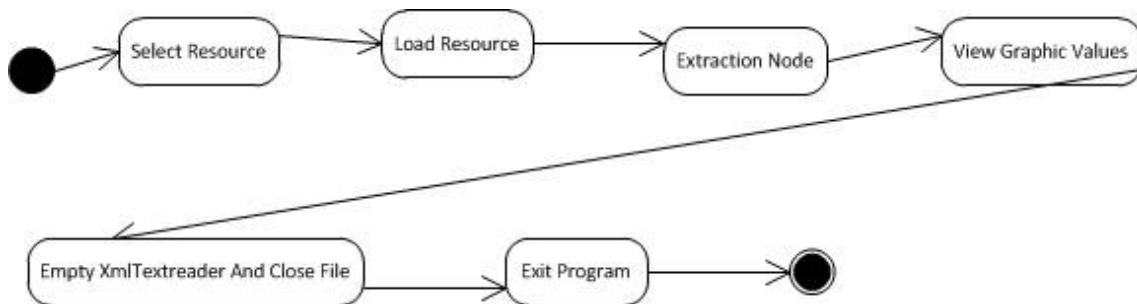
Nel diagramma sono indicate le sequenze di azioni svolte per l'esecuzione del cruscotto principale gestito dalla classe *Form1*. . Dopo il caricamento mediante l'evento *Load* di

tale classe, un altro suo evento *Paint* si occupa di disegnare a video tutti gli oggetti grafici compresi gli indicatori tipo *level bar* ottenuti con una libreria grafica, i quali necessitano del *refresh* possibile solo se effettuato in tale evento.

Si istanzia quindi la classe *Parsexmldoc* per eseguire in serie i suoi metodi *ReadElemGlobal()*, *ThresholdRead()* e *DataGridVG()*, i quali servono ad estrarre i valori dei nodi, i loro valori dei range ed a popolare gli oggetti *DataGridView* che ne mostrano i risultati. Successivamente la classe *Parsexmldoc* imposta le variabili per l'offset di distanza verticale ed orizzontale degli indicatori grafici presenti nella *Form1*, quindi viene eseguito il metodo *DrawGraph()* che li disegna.

Si richiamano in fine il metodo *ExtractAllSubNode()* della classe *Parsexmldoc* incaricato di estrarre i nodi figli di quelli prescelti, ed in ultimo, dopo che si ha a disposizione ogni informazione, si richiama il metodo *ViewAllInTheForm()* della classe *Form1* che visualizza ogni dato ed oggetto nella *form*.

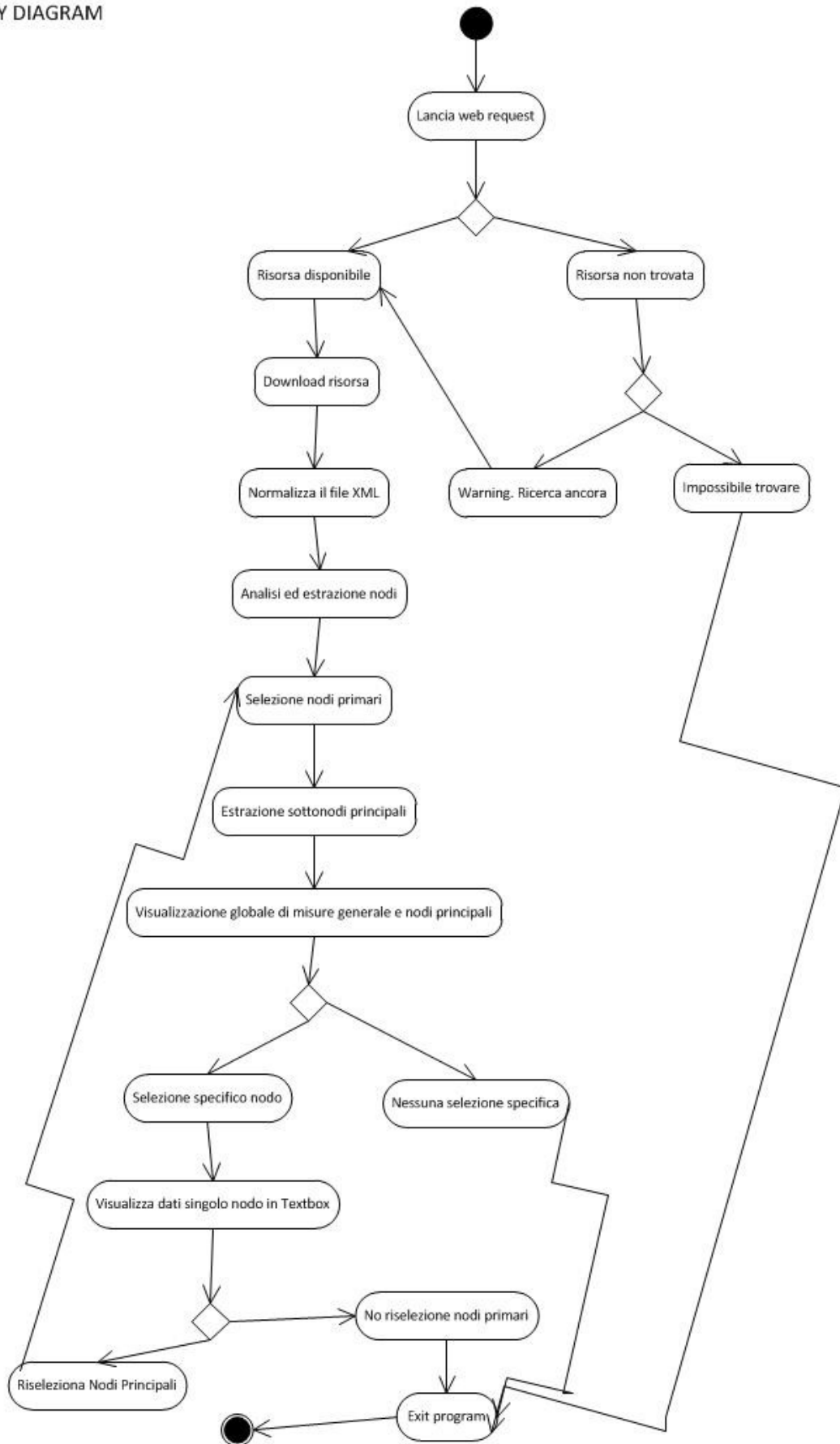
STATE DIAGRAM (livello generale)



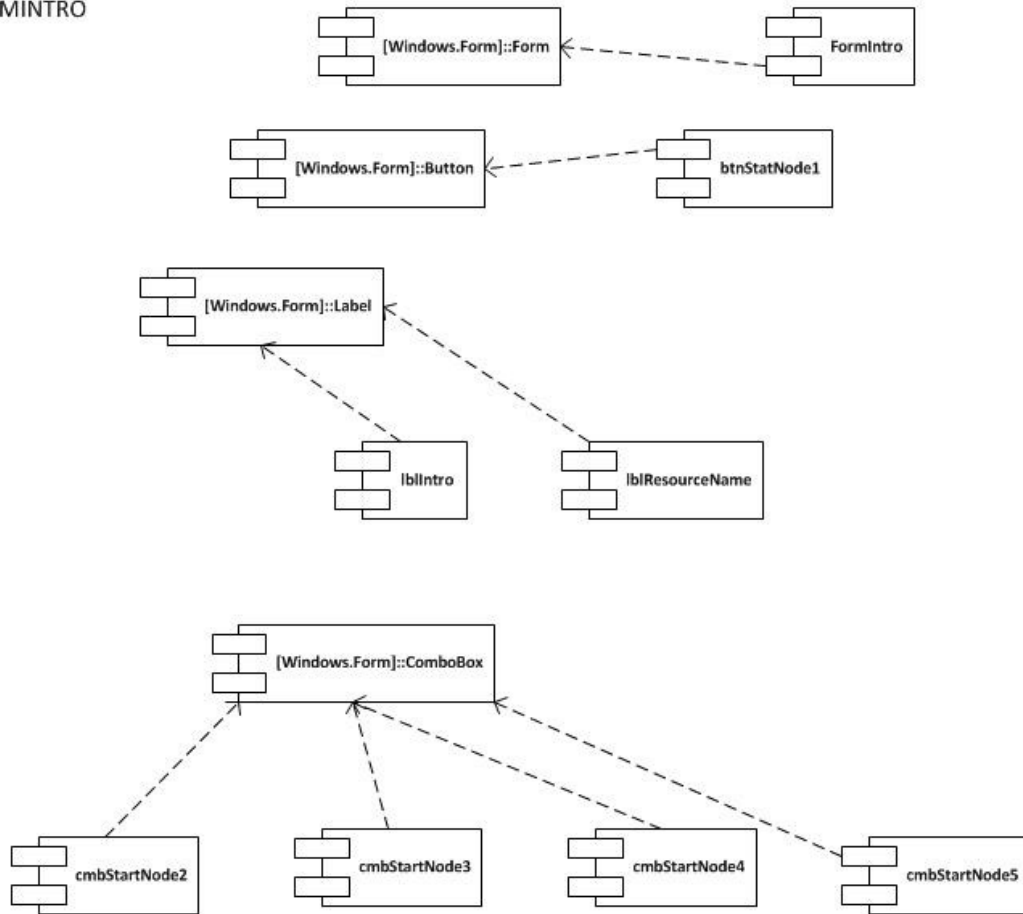
Gli stati più importanti che il **sw** può assumere sono principalmente di:

- Selezione risorsa
- *Download* e salvataggio della risorsa rappresentata dal file **XML**
- Analisi ed estrazione degli elementi nodi contenuti nel file
- Visualizzazione grafica dei risultati dell'estrazione
- Svuotamento e chiusura degli oggetti *XmlTextReader()* e *stream* di lettura
- Chiusura del programma e liberazione delle risorse impegnate fra quelle anche grafiche

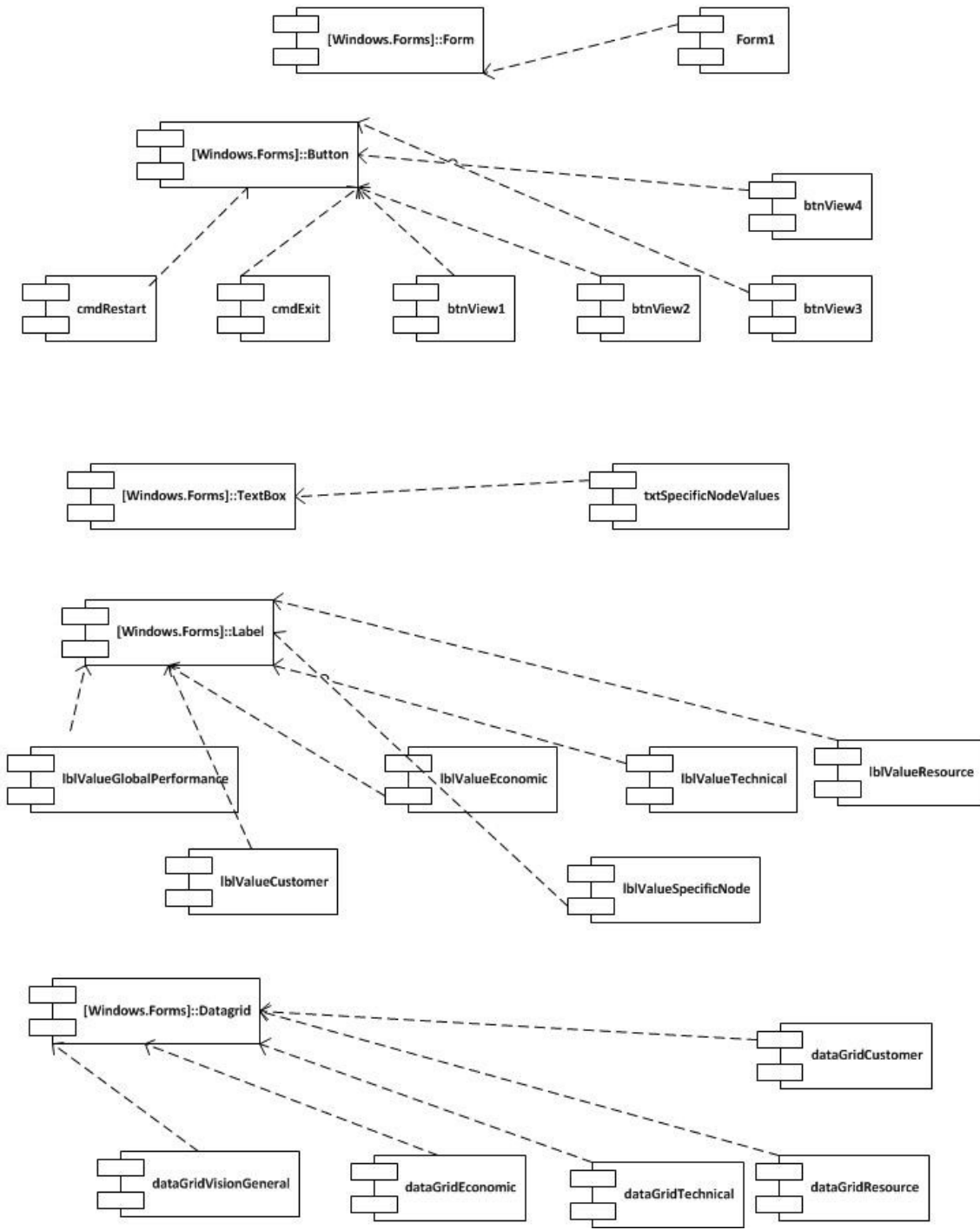
ACTIVITY DIAGRAM



COMPONENT DIAGRAM
FORMINTRO



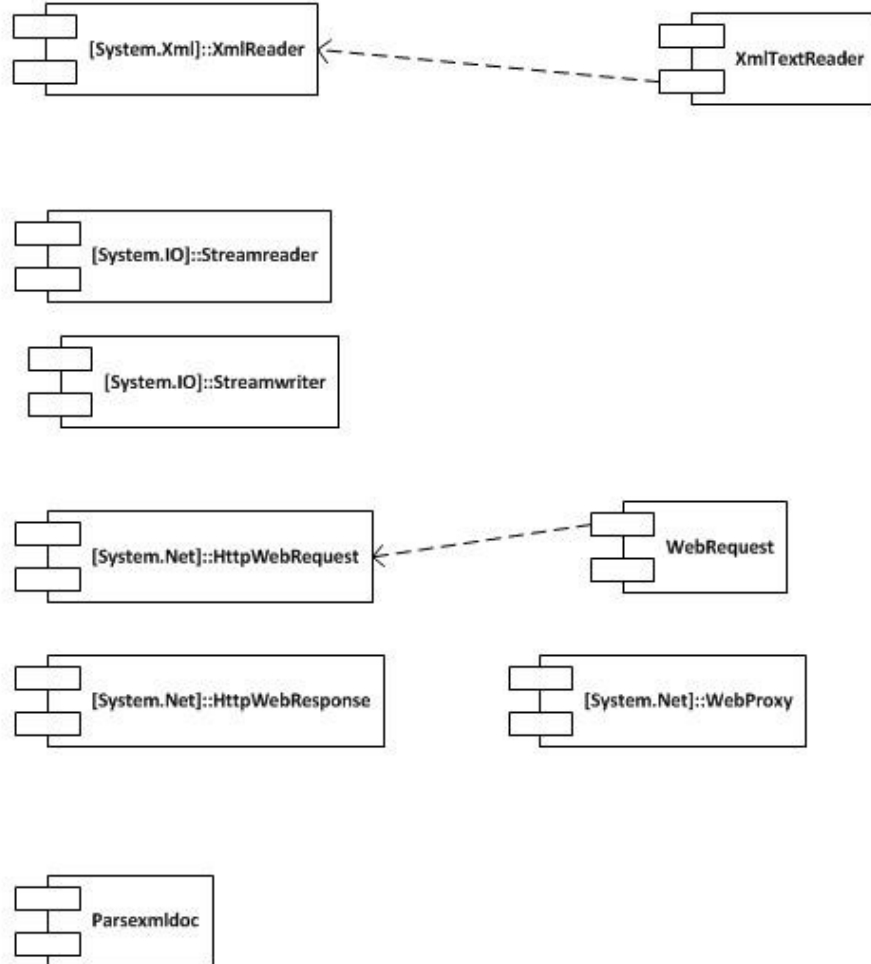
COMPONENT DIAGRAM
FORM1



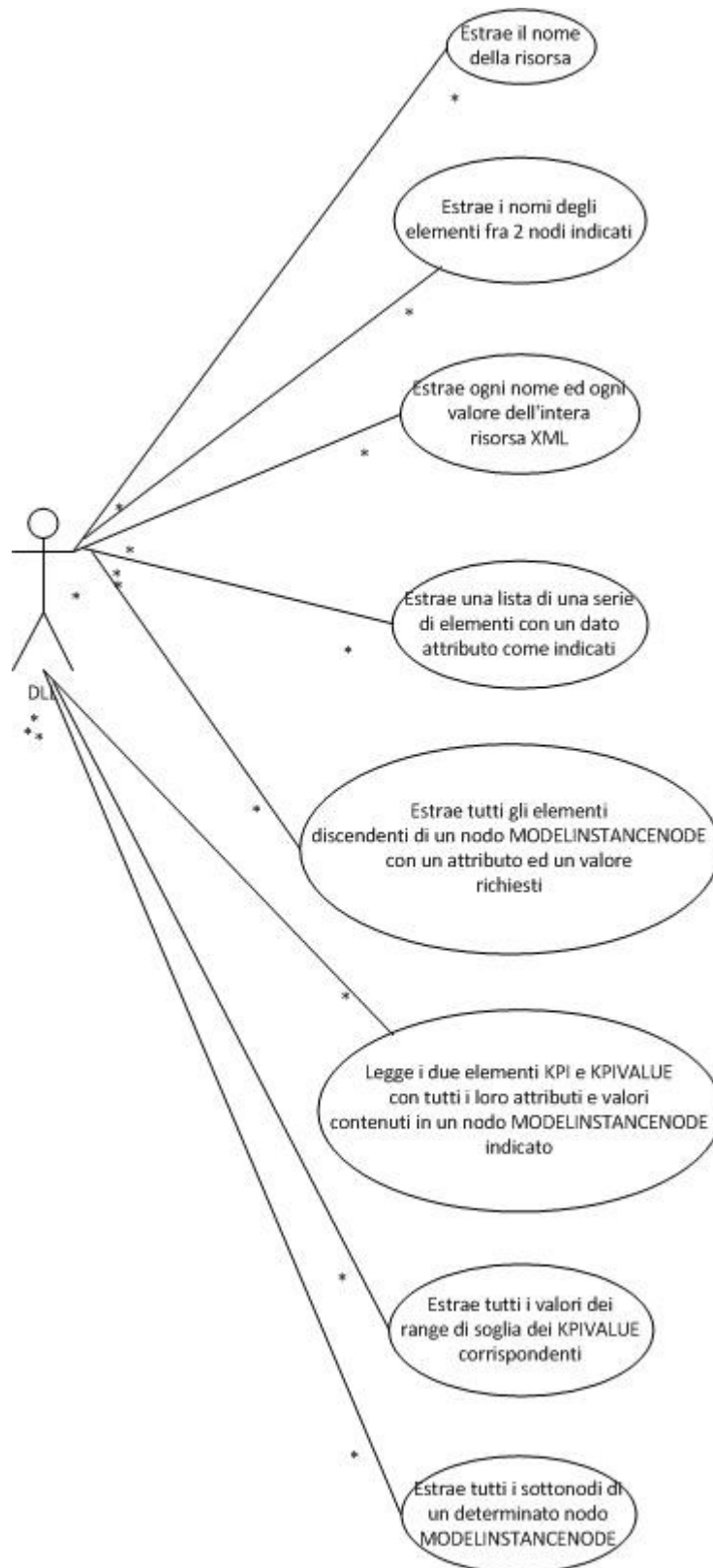
La classi *FormIntro* e *Form1* che rappresentano rispettivamente la 1° e la 2° schermata SW incapsulano alcuni oggetti derivati dai più comuni componenti grafici disponibili con VS così come indicato nel precedente schema *component* UML: **Form**, **Button**, **Label**, **Combobox**, **Textbox** e **Datagrid** personalizzati attraverso l'uso delle proprietà d'aspetto e di dimensioni.

6.2 GRAFICI UML RELATIVI ALLA DLL

COMPONENT DIAGRAM
DLL Parser Document XLM



La **DLL** disponibile è composta dalle classi che consentono sia la lettura del formato **XML** che gli *stream* da passare ai file, oltre che quelli preposti al collegamento ai *web service* con l'uso del protocollo **HTTP**. Nel *component UML* vengono quindi elencati le classi *XmlTextReader* derivata dalla classe *XmlReader* per l'analisi di un file **XML**, le classi di lettura e scrittura flussi utili nel trattamento file come le *StreamReader* e *StreamWriter*, le classi di utilizzo risorse *web* come la *WebRequest* derivata dalla classe *HttpWebRequest* ed infine la classe *Parsexmldoc* per l'estrazione dai dal documento risorsa.



Lo *use case* relativo alla **DLL** elenca quali operazioni sono possibili con l'inserimento di tale componente in un qualunque progetto **.Net**.

6.3 CODICE SORGENTE DLL E SUO FUNZIONAMENTO

La **DLL** compilata in **Visual Studio .Net** con linguaggio **C#** deriva dalla classe **Parsexmldoc** che implementa l'interfaccia **Iparsexml**.

Le funzioni da essa offerte sono quelle elencate dagli *use case* **UML** precedentemente mostrati e fanno uso costante della classe **XmlTextReader** che deriva dalla classe **XmlReader** appartenenti al namespace **System.Xml**.

Nel seguito verrà descritto il funzionamento di ogni metodo illustrandone il codice sorgente spiegato nei dettagli.

METODO **ExtractBetweenElement()**

```
/// <summary>
/// this method can extract all the subnodes between two specified nodes
/// </summary>
/// <param name="ElemStart">Name of the start element</param>
/// <param name="ElemEnd">Name of the end element</param>
/// <returns>All names found</returns>
public string ExtractBetweenElement(string ElemStart, string ElemEnd)
{
    string str = "";
//variable containing the list of names of child nodes
    try
    {
        XmlTextReader dr = new XmlTextReader(pathdocxml);
//Object for reading XML file
        dr.ReadToFollowing(ElemStart);
//Reads until an element with the specified qualified name is found

        do //For each element is assigned its value to a variable
        {
            dr.ReadString();
            str += dr.Name; ;
            dr.ReadString();

        } while (dr.ReadToNextSibling(ElemEnd)); //Advances the XmlReader to
the next sibling element with the specified qualified name

    }
    catch (FileNotFoundException e)
    {
        MessageBox.Show(e.ToString(), "Error Attention!",
        MessageBoxButtons.OK);
        System.Environment.Exit(0);
    }
}
```

```

        return str;
    }

```

Questo metodo consente di selezionare un nome elemento di inizio ed un nome elemento di fine per estrarre tutti gli elementi contenuti fra questi due delimitatori. Creando un oggetto della classe **XmlTextReader** il quale carica in memoria un file **XML** indicato come parametro, mediante il metodo **ReadToFollowing()** il *reader* viene posizionato sul primo elemento voluto e successivamente con un ciclo *do..while* si scandiscono tutti gli elementi catturando ogni elemento fino al nome elemento indicato nel metodo **ReadToNextSibling()**.

Nell'utilizzo pratico con un documento di risorsa *Spago4Q* serve ad esempio per leggere i nomi contenuti fra un nodo **"MODELINSTANCENODE"** ed un nodo **"KPIVALUE"**.

METODO ResourceName()

```

    /// <summary>
    /// This method read the name for the identification resource file
    /// </summary>
    /// <returns>the name of resource</returns>
    public string ResourceName()
    {
        string str = ""; //variable containing the value of the resource name
        try
        {
            XmlTextReader dr = new XmlTextReader(pathdocxml);
            //Object for reading the XML file
            while (dr.Read())
            //until the method is true search RESOURCE element name
            {
                if (dr.Name == "RESOURCE")
                {
                    dr.MoveToElement();
                    //Moves to the element that contains the current attribute node
                    if (dr.MoveToAttribute("name"))
                    //If attribute name is found store value in the variable
                    {
                        str = dr.Value;
                        if (str == "")
                        {

```



```

        listElem.Add(nameelem);
        nameelem = nameelem + "Elemento: " + nameelem + "\r\n";
    }
    if (dr.HasAttributes) //checks if the element has attributes
    {
        while (dr.MoveToNextAttribute())
//until an element has attributes to store their values and adds them to the
ArrayList
        {
            string nameattrib = dr.Name.ToString();
            string valueattrib = dr.Value.ToString();
            listAttri.Add(nameattrib + " " + valueattrib);
            string str = " Attributo: " + nameattrib + " " +
valueattrib + "\r\n";
        }
    }
    dr.Close(); //Close object for reading XML resource
}
catch (FileNotFoundException e)
{
    MessageBox.Show("File not found", "Error Attention!",
MessageBoxButtons.OK);
    System.Environment.Exit(0);
}
}
}

```

Con un'analisi integrale del documento, ancora una volta grazie alla classe **XmlTextReader**, si esegue un ciclo *while* nel quale si continua a leggere la risorsa in sequenza finché il metodo **Read()** restituisce *true*, quindi se un nodo viene definito di tipo *element* come diagnosticato dall'oggetto **XmlNodeType**, vengono catturati ed assegnati a variabili stringa tutti i nomi degli elementi, i propri nomi attributi ed i valori di essi, aggiungendo i dati estratti in due collezioni **ArrayList** di tipo stringa (uno per gli elementi ed uno per gli attributi), in quanto queste comode collezioni disponibili nel *namespace System.Collections* consentono di espandersi dinamicamente e possono essere facilmente adoperate dai costrutti *Foreach* del C#.

Questo metodo è utile per avere una visione completa dell'intero contenuto di un file **XML** che può essere facilmente perfezionato con l'impiego di altri strumenti d'assegnazione e di collezione come: *array*, **ArrayList** o semplici variabili, al fine di

costruire con essi dei *report* o altre forme di visualizzazione, considerando che non è solo applicabile ad una risorsa *Spago4Q* bensì ad ogni file XML.

METODO ListNode()

```
    /// <summary>
    /// Create a list of all values of the elements and of the attribute
selected
    /// </summary>
    /// <param name="name_element">name element node xml to search</param>
    /// <param name="nameattr">value attribute xml to search</param>
    public void ListNode(string name_element, string nameattr)
    {
        try
        {
            XmlTextReader dr = new XmlTextReader(pathdocxml);
//Creation object for reading XML file resource
            listname.Clear(); //Empty ArrayList before filling
            while (dr.Read()) //Read each node until the Read method is true
            {
                if (dr.Name == name_element && dr.HasAttributes)
//When the node is an element and has attributes store all in the arraylist
                {
                    dr.MoveToElement();
//Moves to the element that contains the current attribute node
                    while (dr.MoveToNextAttribute())
//Read until the node has attributes
                    {
                        if (dr.Name == nameattr && dr.Value != "")
                        {
                            string attrname = dr.Value.Trim();
                            listname.Add(attrname);
                        }
                    }
                }
            }
        }
        catch (FileNotFoundException e)
        {
            MessageBox.Show("File not found", "Error Attention!",
MessageBoxButtons.OK);
            System.Environment.Exit(0);
        }
    }
}
```

Il metodo descritto è in grado di estrarre una lista di nodi corrispondenti ad un nome elemento con una stringa prescelta, e che hanno un attributo comune dato da un particolare nome (sempre di tipo stringa). Ognuno dei nodi estrapolati viene quindi inserito in un oggetto di collezione dinamica **ArrayList**.

Il tutto avviene generando un ciclo *while* nel quale finché il metodo **Read()** restituisce *true*, se l'elemento ha un determinato nome e possiede degli attributi, confronta ogni nome attributo con il nome passato come parametro al metodo, dunque se i due sono eguali aggiunge il valore al **ArrayList** preposto.

Questa funzione occorre nella ricerca di elementi con lo stesso nome e con un eguale attributo per la memorizzazione di tutti i valori da collezionare e mostrare in seguito.

METODO GetFromStartEndModelinstancenode()

```

    /// <summary>
    /// Shows the subnodes contained within a specific node MODELINSTANCENODE
    /// </summary>
    /// <param name="paramname">name of the element to search</param>
    /// <param name="paramvalue">value of the name element to search</param>
    /// <returns>values of the all attributes name and value</returns>
    public string GetFromStartEndModelinstancenode(string paramname, string
paramvalue)
    {
        string result = "";
        try
        {
            XmlReader rdDoc = XmlReader.Create(pathdocxml);
//Creation object for reading xml resource

            while (rdDoc.Read())
//Loop for reading each element until the Read method return true
            {
                if (rdDoc.Name == "MODELINSTANCENODE")
//check the name element
                {
                    rdDoc.MoveToAttribute(paramname);
//Moves to the attribute you want
                    if (rdDoc.Value == paramvalue)
//Compare the value of the attribute value
                    {
                        rdDoc.MoveToElement();
//Moves to the element that contains the current attribute node
                        XmlReader rdElem1 = rdDoc.ReadSubtree();//Returns a
new XmlReader instance that can be used to read the current node, and all its
descendants

                        while (rdElem1.Read())
//Scansione degli elementi contenuti nell'oggetto rdDoc
                        {
                            string elemname = rdElem1.Name + "\r\n";
                            result += elemname;
//Store the names of elements in the variable
                            if (rdElem1.HasAttributes)
//checks if the element has attributes

```


METODO ReadElemGlobal()

```
/// <summary>

/// This method read a complete element MODELINSTANCENODE with two
subelement kpi and kpivalue. This method also sets all variables to store each
value.
/// </summary>
/// <param name="elemsearch">Name search item</param>
/// <param name="valuesearch">Value name search item</param>
public void ReadElemGlobal(string elemsearch, string valuesearch)
{
    try
    {
        XmlTextReader xrd = new XmlTextReader(pathdocxml);
//Creating an object for reading XML file
        XmlNodeType tn = xrd.NodeType;
//Creating an object to determine XML node type

        while (xrd.Read())
//Loop for reading all elements until the method is true
        {
            if (xrd.Name == "MODELINSTANCENODE" && xrd.IsStartElement())
//Checks if the name element is given and if is an start element
            {
                xrd.MoveToElement();
//Moves to the element that contains the current attribute node
                xrd.MoveToAttribute(elemsearch);
//Moves to the attribute with the specified name
                if (xrd.Value.ToString() == valuesearch)
//Checks if the value of the attribute is given
                {
                    xrd.MoveToElement();
                    string valuecoderoor = xrd.Value;
//Assigned to the variable string attribute value

                    while (xrd.MoveToNextAttribute())
//Read all the attributes of current element
                    {
                        //Assigned name and value attributes to variables
                        string nameattr = xrd.Name;
                        string valueattr = xrd.Value;
                        //analysis and extraction of each attribute
                        switch (nameattr) //check every element name
                        {
                            //for each MODELINSTANCENODE element checks
                            the element name and adds its value in the list
                            case "name":
                                name_node_global = "";
                                name_node_global = valueattr;
                                if (valueattr == "")
                                    name_node_global = "Blank Element";
                                break;
                            case "code":
                                code_node_global = "";
                                code_node_global = valueattr;
                                if (valueattr == "")
```

```

        code_node_global = "Blank Element";
        break;
    case "description":
        description_node_global = "";
        description_node_global = valueattr;
        if (valueattr == "")
            description_node_global = "Blank
Element";

        break;
    default:
        break;
    }
}

xrd.MoveToElement();
//for each KPI child element checks the element name
and adds its value in the list
if (xrd.ReadToFollowing("KPI"))
{
    while (xrd.MoveToNextAttribute() && xrd.Name !=
"MODELINSTANCENODE")
    {
        string nameattr2 = xrd.Name;
        string valueattr2 = xrd.Value;
        switch (nameattr2)
        {
            case "description":
                kpi_description_global = "";
                kpi_description_global = valueattr2;
                if (valueattr2 == "")
                    kpi_description_global = "Blank
Element";

                break;
            case "name":
                kpi_name_global = "";
                kpi_name_global = valueattr2;
                if (valueattr2 == "")
                    kpi_name_global = "Blank Element";
                break;
            case "interpretation":
                kpi_interpretation_global = "";
                kpi_interpretation_global =
valueattr2;

                if (valueattr2 == "")
                    kpi_interpretation_global = "Blank
Element";

                break;
            case "code":
                kpi_code_global = "";
                kpi_code_global = valueattr2;
                if (valueattr2 == "")
                    kpi_code_global = "Blank Element";
                break;
            default:
                break;
        }
    }
}
}

```

```

xrd.MoveToElement();
//for each KPIVALUE child element checks the element
name and adds its value in the list
if (xrd.ReadToFollowing("KPIVALUE"))
{
while (xrd.MoveToNextAttribute() && xrd.Name !=
"MODELINSTANCENODE")
{
string nameattr3 = xrd.Name;
string valueattr3 = xrd.Value;
switch (nameattr3)
{
case "begindate":
kpivalue_bgdate_global = "";
kpivalue_bgdate_global = valueattr3;
if (valueattr3 == "")
kpivalue_bgdate_global = "Blank
Element";

break;
case "enddate":
kpivalue_enddate_global = "";
kpivalue_enddate_global = valueattr3;
if (valueattr3 == "")
kpivalue_enddate_global = "Blank
Element";

break;
case "target":
kpivalue_target_global = "";
kpivalue_target_global = valueattr3;
if (valueattr3 == "")
kpivalue_target_global = "Blank
Element";

break;
case "description":
kpivalue_description_global = "";
kpivalue_description_global =
valueattr3;

if (valueattr3 == "")
kpivalue_description_global =
"Blank Element";

break;
case "weight":
kpivalue_weight_global = "";
kpivalue_weight_global = valueattr3;
if (valueattr3 == "")
kpivalue_weight_global = "Blank
Element";

break;
case "thresholdid":
kpivalue_thresholdid_global = "";
kpivalue_thresholdid_global =
valueattr3;

if (valueattr3 == "")
kpivalue_thresholdid_global =
"Blank Element";

break;
case "weightedvalue":
kpivalue_weightedvalue_global = "";
kpivalue_weightedvalue_global =
valueattr3;

```


METODO ThresholdRead()

```
    /// <summary>

    /// Reads the values of threshold items based on value of thresholdid
modelinstancenode
    /// </summary>
    public void ThresholdRead()
    {
        try
        {
            XmlReader xrd = XmlReader.Create(pathdocxml);
//object to read xml file

            while (xrd.Read())
//scan all the items until the read method is true
            {
                if (xrd.Name == "THRESHOLD" && xrd.IsStartElement())
//check that the item is the start of the section Threshold
                {
                    if (xrd.MoveToAttribute("id") && xrd.Value ==
kpivalue_thresholdid_global) //Compare the id attribute with the value derived
from the list to check the reference threshold
                    {
                        xrd.MoveToElement();
//Moves to the element that contains the current attribute node
                        XmlReader drrs = xrd.ReadSubtree(); //Returns a new
XmlReader instance used to read the current node of the object xrd and all its
descendants

                            while (drrs.Read())
//Read each element of the drrs object
                            {
                                if (drrs.ReadToFollowing("RANGE"))
// if the item is RANGE read its attributes
                                {
                                    drrs.MoveToElement();
//Moves to the element that contains the current attribute node
                                    while (drrs.MoveToNextAttribute())
//Moves to the next attribute
                                    {
                                        string nameattrth = drrs.Name;
//string variable for name value
                                        string valueattrth = drrs.Value;
//string variable for value attribute
                                        switch (nameattrth)
//For each attribute min and max stores their values in the variables
                                        {
                                            case "min":
                                                range1min_bad = "";
                                                range1min_bad = valueattrth;
                                                break;
                                            case "max":
                                                range1max_bad = "";
                                                range1max_bad = valueattrth;
                                                break;
                                            default:
                                                break;
                                        }
                                    }
                                }
                            }
                    }
                }
            }
        }
    }
}
```

```

    }
}

drrs.MoveToElement();
if (drrs.ReadToFollowing("RANGE"))
//if the item is 2° RANGE read its attributes
{
    while (drrs.MoveToNextAttribute())
//Moves to the next attribute
    {
        string nameattrth = drrs.Name;
//string variable for name value
        string valueattrth = drrs.Value;
//string variable for value attribute
        switch (nameattrth) //For each attribute
min and max of the second RANGE element stores their values in the variables
        {
            case "min":
                range2min_warning = "";
                range2min_warning = valueattrth;
                break;
            case "max":
                range2max_warning = "";
                range2max_warning = valueattrth;
                break;
            default:
                break;
        }
    }
}

drrs.MoveToElement();
if (drrs.ReadToFollowing("RANGE"))
//if the item is 3° RANGE read its attributes
{
    while (drrs.MoveToNextAttribute())
//Moves to the next attribute
    {
        string nameattrth = drrs.Name;
//string variable for name value
        string valueattrth = drrs.Value;
//string variable for value attribute
        switch (nameattrth) //For each attribute
min and max of the third RANGE element stores their values in the variables
        {
            case "min":
                range3min_good = "";
                range3min_good = valueattrth;
                break;
            case "max":
                range3max_good = "";
                range3max_good = valueattrth;
                break;
            default:
                break;
        }
    }
}

```

```

        }
        }
        drrs.MoveToElement();
//Moves to the element that contains the current attribute node
    }

}

}
}
catch (FileNotFoundException e) //catch the error for file not found
{
    MessageBox.Show(e.ToString(), "Error Attention!",
    MessageBoxButtons.OK); //Show the message error
    System.Environment.Exit(0); //Close the program
}
}
}

```

Questo metodo serve all'applicazione per poter conoscere quali sono i range attribuiti per ogni elemento **"MODELINSTANCENODE"** relazionato alla sezione **"THRESHOLD"** della risorsa.

In pratica in ogni elemento **"KPIVALUE"** è presente un attributo **"thresholdid"** attraverso il quale si possono identificare quali sono le soglie stabilite comparando tale valore attributo con quello dell'attributo **"id"** appartenente all'elemento **"Threshold"**.

Si esegue quindi un'istruzione condizionale semplice che nel caso restituisca **true** da inizio alla scansione dell'intero nodo compresi gli attributi e i nodi discendenti, per proseguire quindi con l'estrapolazione dei valori che verranno assegnati alle variabili numeriche. I parametri da passare al metodo indicano quale elemento esatto **"MODELINSTANCENODE"** si vuole estrarre, specificando l'attributo ed il suo valore da cercare.

METODO ExtractAllSubNode()

```

    /// <summary>
    /// This method is used to show all the nodes within a node
MODELINSTANCENODE
    /// </summary>

```

```

    /// <param name="paramsearch">The element to search with value of
code</param>
    public void ExtractAllSubNode(string paramsearch, string valuesearch)
    {
        try
        {
            XmlReader rdDoc = XmlReader.Create(pathdocxml);
//Instance of the object for reading file xml
            subnode_code.Clear(); //Clear for security the arraylist
            subnode_name.Clear();
            while (rdDoc.Read())
//As long as there is nothing to read continues to scan the file
            {
                if (rdDoc.Name == "MODELINSTANCENODE") //If the name of the
current element is equal to MODELINSTANCENODE move to attribute passed as a
parameter
                {
                    rdDoc.MoveToAttribute(paramsearch);
                    if (rdDoc.Value == valuesearch) //If the value is equal
to value passed as a parameter to get all values of the code and name element
                    {
                        rdDoc.MoveToElement(); //Moves to the element that
contains the current attribute node
                        XmlReader rdElem1 = rdDoc.ReadSubtree(); //Returns a
new XmlReader instance used to read the current node of the rdDoc object, and all
its descendants
                        while (rdElem1.Read())
//Read all element of the rdElem1 object until the Read method is true
                        {
                            if (rdElem1.Name == "MODELINSTANCENODE" &&
rdElem1.IsStartElement() == true)
//Check if the element has MODELINSTANCENODE name and its an start element
                            {
                                rdElem1.MoveToAttribute("code");
//Moves to code attribute
                                subnode_code.Add(rdElem1.Value.ToString());
//Stores value of code attribute in the ArrayList
                                rdElem1.MoveToAttribute("name");
//Moves to name attribute
                                subnode_name.Add(rdElem1.Value.ToString());
//Stores value of name attribute in the ArrayList
                            }
                        }
                    }
                }
            }
        }
        catch (FileNotFoundException e)
//Catch l'exception for File not found
        {
            MessageBox.Show(e.ToString(), "Error Attention!",
MessageBoxButtons.OK); //Show the error message
        }
    }
}

```

Tale funzione consente l'estrazione di tutti gli attributi *“code”* e *“name”* dei sotto-elementi denominati **"MODELINSTANCENODE"** contenuti in un nodo padre **"MODELINSTANCENODE"** e scelto dall'utente con i parametri richiesti. Si utilizzano ancora una volta gli oggetti **XmlReader** e la chiamata del suo metodo **ReadSubtree()** per ottenere un contenitore degli elementi discendenti del primo oggetto creandone una nuova istanza, pertanto a differenza dei precedenti metodi in questo caso ci si concentra su un nodo esclusivo di un documento generato da *Spago4Q* che possiede nomi ed attributi espliciti come: *“code”*, *“name”*.

6.4 CODICE SORGENTE (parti salienti Windows Forms Application)

Per un'esposizione più eloquente si mostrano con delle brevi spiegazione accompagnate dai listati le parti più importanti dell'applicazione, sottolineando l'utilizzo nell'ambiente **MS Windows** in cui è stata sviluppata l'utility con l'implementazione di una singola interfaccia.

Step by step viene nel seguito descritto l'evoluzione del software in modo pratico sin dall'avvio, .

All'esecuzione del programma viene eseguita e mostrata la *window FormIntro*

Il codice relativo a tale classe è:

```
public partial class FormIntro : Form
{
    Parsexml doc psx = new Parsexml doc();
    //object of class Parsexml doc to call the methods of reading resources XML

    //Check if first view FormIntro
    public static bool FirstShow = true;

    public FormIntro() //initializes the windows form
    {
        InitializeComponent();
    }
}
```

```

private void FormIntro_Load(object sender, EventArgs e)
{
    OpenResource os = new OpenResource();
//object to download and open the XML file
    if (FirstShow == true)
//check if it's the first opening of the window
    {
        os.ReqExecuteDocument();
//If it's the first Form opening sends web request for to download resource file
    }
    os.PathLocalResource();
//sets the local path where it's saved the file
    this.lblResourceName.Text += " " + psx.ResourceName();
//Shows in the label the resource nama
    LoadListNodeStart();
//filling the ComboBox objects with the names of the all nodes MODELINSTANCENODE
    cmbStartNode2.SelectedIndex = 1;
//Setting selected item of the first ComboBox
    cmbStartNode3.SelectedIndex = 11;
//Setting selected item of the second ComboBox
    cmbStartNode4.SelectedIndex = 21;
//Setting selected item of the third ComboBox
    cmbStartNode5.SelectedIndex = 15;
//Setting selected item of the fourth ComboBox
    }

    public void LoadListNodeStart()
//Loading of the all nodes in the ComboBox
    {
        psx.ListNode("MODELINSTANCENODE", "name");
//Extraction of the each node with an name MODELINSTANCENODE and an attribute
named pluto
        foreach (string a in psx.listname)
//loading the combobox of all items found
        {
            if (a != "")
            {
                this.cmbStartNode2.Items.Add(a);
                this.cmbStartNode3.Items.Add(a);
                this.cmbStartNode4.Items.Add(a);
                this.cmbStartNode5.Items.Add(a);
            }
        }
    }

    public void SetStartNode() //Setting of the main nodes to view
    {
        Form1.n2 = this.cmbStartNode2.Text;
//Variable setting of the main nodes selected
        Form1.n3 = this.cmbStartNode3.Text;
        Form1.n4 = this.cmbStartNode4.Text;
        Form1.n5 = this.cmbStartNode5.Text;
    }

    private void btnStartNode1_Click(object sender, EventArgs e)
//click mouse event management
    {
        Form1 frm1 = new Form1(); //Initialization window form
        this.Hide(); //Hide the main form
    }

```

```

        SetStartNode(); //Setting start nodes selected
        frm1.Show(); //Show the window form
        FirstShow = false;
//Setting variable for recognize first opening window form
    }

    private void FormIntro_FormClosed(object sender, FormClosedEventArgs e)
//FormClosed click event management
    {
        Application.Exit(); //Close the program
    }
}

```

Questa classe oltre ad incaricarsi del download in locale della risorsa mediante l'oggetto **OpenResource** con i relativi metodi, fra cui il **ReqExecuteDocument()** richiamato per il collegamento al *web service* di *Spago4Q* ed il reperimento di una risorsa specifica prestabilita, svolge anche l'analisi del file scaricato e consente in fase di *run-time* di scegliere quali nodi primari considerare per la visualizzazione.

Il metodo **LoadListNodeStart()** al suo interno istanzia l'oggetto **psx** della classe **Parsexmldoc** per eseguire il metodo **ListNode()**, questo nel caso specifico estrae tutti i valori dell'attributo *"name"* di tutti i nodi **"MODELINSTANCENODE"** con cui si riempiono le quattro **Combobox**, dalle quali è possibile selezionare i quattro nodi primari da visualizzare in dettaglio. Al valore della proprietà *"current item"* di ogni **Combobox** vanno a corrispondere per assegnamento, attraverso il metodo **SetStartNode()**, le variabili stringa della **Form1** usate come parametri nel metodo **ViewAllInTheForm()** che imposta la schermata del pannello principale nel modo così illustrato.

Codice ViewAllInTheForm():

```

    public void ViewAllInTheForm(string n1, string n2, string n3, string n4,
string n5)
    {

```

```

        Parsexml doc pd = new Parsexml doc(); //Object for analysis XML file
        this.dataGridViewGeneral.Rows.Clear();
//Clear all item of the DataGridView object

        Parsexml doc.codefind = "QEST nD for Business Service";
//sets the variable for the selection of a node
pd.ReadElemGlobal("name", Parsexml doc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueGlobalPerformance.Text = "Name resource: " +
pd.ResourceName() + "\r\n" + Parsexml doc.codefind + "\r\n" + "Value: " +
Parsexml doc.kpivalue_value_global; //shows the values found in the label: name
resource and global value of the main node
this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueGlobalPerformance.Location.X;
//sets the horizontal distance from the label that shows the global value
Form1.yline = this.lblValueGlobalPerformance.Location.Y + 80;
//sets the vertical distance from the label that shows the total value
this.DrawGraph(); //Draw in the window form graphic indicators

        Parsexml doc.codefind = n2;
//sets the variable for the selection of the first parent node node
pd.ReadElemGlobal("name", Parsexml doc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueEconomic.Text = Form1.n2 + " : " +
Parsexml doc.kpivalue_value_global;
//shows the values found in the label: name parent node and global value
this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueEconomic.Location.X;
//sets the new horizontal distance from the label that shows the global value
Form1.yline = this.lblValueEconomic.Location.Y + 45;
//sets the new vertical distance from the label that shows the global value
this.DrawGraph(); //Draw in the window form graphic indicators

        Parsexml doc.codefind = n3;
//sets the variable for the selection of the second parent node node
pd.ReadElemGlobal("name", Parsexml doc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueResource.Text = Form1.n3 + " : " +
Parsexml doc.kpivalue_value_global;
//shows the values found in the label: name parent node and global value
this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueResource.Location.X;
//sets the new horizontal distance from the label that shows the global value
Form1.yline = this.lblValueResource.Location.Y + 45;
//sets the new vertical distance from the label that shows the global value
this.DrawGraph(); //Draw in the window form graphic indicators

        Parsexml doc.codefind = n4;
//sets the variable for the selection of the third parent node node
pd.ReadElemGlobal("name", Parsexml doc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueTechnical.Text = Form1.n4 + " : " +
Parsexml doc.kpivalue_value_global;
//shows the values found in the label: name parent node and global value
this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueTechnical.Location.X;

```

```

//sets the new horizontal distance from the label that shows the global value
Form1.yline = this.lblValueTechnical.Location.Y + 45;
//sets the new vertical distance from the label that shows the global value
this.DrawGraph(); //Draw in the window form graphic indicators

Parsexmldoc.codefind = n5;
//sets the variable for the selection of the fourth parent node node
pd.ReadElemGlobal("name", Parsexmldoc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueCustomer.Text = Form1.n5 + " : " +
Parsexmldoc.kpivalue_value_global;
//shows the values found in the label: name parent node and global value
this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueCustomer.Location.X;
//sets the new horizontal distance from the label that shows the global value
Form1.yline = this.lblValueCustomer.Location.Y + 45;
//sets the new vertical distance from the label that shows the global value
this.DrawGraph(); //Draw in the window form graphic indicators

Parsexmldoc.subcodefind = n5; //sets the variable for the selection
of the fourth child nodes of the fourth parent node
pd.ExtractAllSubNode("name", Parsexmldoc.subcodefind);
//Selection and extraction of the all subnodes
this.dataGridCustomer.Rows.Clear();
//Clear all item of the DataGridView
for (int n = 0; n < Parsexmldoc.subnode_name.Count; n++)
//fills the DataGridView with all elements taken from the ArrayList
{
    this.dataGridCustomer.Rows.Add(Parsexmldoc.subnode_name[n],
Parsexmldoc.subnode_code[n]); //Add each name element in the control DataGridView
}

Parsexmldoc.subcodefind = n2; //sets the variable for the selection
of the first child nodes of the fourth parent node
pd.ExtractAllSubNode("name", Parsexmldoc.subcodefind);
//Selection and extraction of the all subnodes
this.dataGridEconomic.Rows.Clear();
//Clear all item of the DataGridView
for (int n = 0; n < Parsexmldoc.subnode_name.Count; n++)
//fills the DataGridView with all elements taken from the ArrayList
{
    this.dataGridEconomic.Rows.Add(Parsexmldoc.subnode_name[n],
Parsexmldoc.subnode_code[n]); //Add each name element in the control DataGridView
}

Parsexmldoc.subcodefind = n3; //sets the variable for the selection
of the second child nodes of the fourth parent node
pd.ExtractAllSubNode("name", Parsexmldoc.subcodefind);
//Selection and extraction of the all subnodes
this.dataGridResource.Rows.Clear();
//Clear all item of the DataGridView
for (int n = 0; n < Parsexmldoc.subnode_name.Count; n++)
//fills the DataGridView with all elements taken from the ArrayList
{
    this.dataGridResource.Rows.Add(Parsexmldoc.subnode_name[n],
Parsexmldoc.subnode_code[n]); //Add each name element in the control DataGridView
}

```

```

        Parsexmldoc.subcodefind = n4; //sets the variable for the selection
of the third child nodes of the fourth parent node
        pd.ExtractAllSubNode("name", Parsexmldoc.subcodefind);
//Selection and extraction of the all subnodes
        this.dataGridTechnical.Rows.Clear();
//Clear all item of the DataGridView
        for (int n = 0; n < Parsexmldoc.subnode_name.Count; n++)
//fills the DataGridView with all elements taken from the ArrayList
        {
            this.dataGridTechnical.Rows.Add(Parsexmldoc.subnode_name[n],
Parsexmldoc.subnode_code[n]); //Add each name element in the control DataGridView
        }

    }
}

```

Come è possibile osservare l'uso delle variabili statiche di tipo stringa della classe **Parsexmldoc**, impiegate anche come parametri per i metodi **ReadElemGlobal()** ed **ExtractAllSubNode()**, è adatto per esprimere i criteri di selezione e di ricerca elementi se impostate in un progetto indipendente da quello di tesi, mentre in questo caso sono usate per interfacciarsi e scambiare i dati fra le *Forms Windows*: questo fa sì appunto che ci sia un primo grado di fruizione della **DLL** integrabile in altri progetti.

Dopo aver scelto i nodi ed aver generato l'evento *onclick* del *button* “SetStartNode” viene mostrata attraverso la classe *Form1* la medesima *window* composta dai controlli grafici fra i quali i più importanti sono i **DataGridView**.

Ognuno di questi oggetti viene identificato con un valore contraddistinto della loro proprietà '*name*' ed ognuno viene inizializzato con il metodo **DataGridVG()** della classe *Form1* che viene richiamato all'interno del metodo **ViewAllInTheForm()**.

Codice DataGridVG():

```

public void DataGridVG()
{
    this.dataGridVissionGeneral.ColumnCount = 2; //number columns
    this.dataGridVissionGeneral.Columns[0].Name = "Name";
//Header name column 1
}

```

```

        this.dataGridVisionGeneral.Columns[1].Name = "Value"; //Header name
column 2
        this.dataGridVisionGeneral.Rows.Add(Parsexml.doc.name_node_global,
Parsexml.doc.kpivalue_value_global);

        this.dataGridEconomic.ColumnCount = 1; //number columns
        this.dataGridEconomic.Columns[0].Name = "Name"; //Header name column
1

        this.dataGridCustomer.ColumnCount = 1; //number columns
        this.dataGridCustomer.Columns[0].Name = "Name"; //Header name column
1

        this.dataGridResource.ColumnCount = 1; //number columns
        this.dataGridResource.Columns[0].Name = "Name"; //Header name column
1

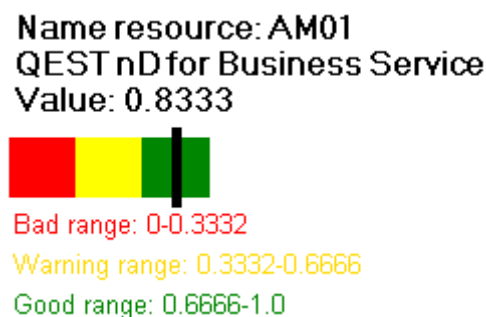
        this.dataGridTechnical.ColumnCount = 1; //number columns
        this.dataGridTechnical.Columns[0].Name = "Name"; //Header name column
1

    }

```

All'interno del pannello sono presenti anche degli indicatori grafici, ispirati a una forma di *level bar graph* in orizzontale, inserita per avere un riferimento rapido del valore globale del nodo **“MODELINSTANCENODE”**. L'immagine è composta da una barra verticale che si posiziona in uno spazio rettangolare diviso in tre segmenti di tre colori diversi, ognuno dei quali ha un'ampiezza pari ai *range* delle *threshold* stabilite nel file XML di risorsa.

Figura 6 – Indicatore grafico di valore e soglie



Il grafico viene generato attraverso la funzione **DrawGraph()** contenuta nella classe **Form1** e viene anch'esso richiamato per ogni **DataGridView** nel metodo **ViewAllInTheForm()**.

Le librerie grafiche utilizzate sono quelle incorporate nel *framework .Net di Microsoft* e viene istanziata la classe **Graphics** del namespace **System.Drawing**, la quale mette a disposizione ottimi metodi di facile comprensione per la generazione di grafici non troppo complessi, ma di natura statistica e/o matematica.

Codice DrawGraph():

```
public void DrawGraph() //Method for draw graphic indicators
{
    Graphics gs = this.CreateGraphics();
    //instance of the use of graphic design methods

    //Casting of the variables related to the values KPI
    x1min = Convert.ToDecimal(Parsexmldoc.range3min_good) * 100;
    x1max = Convert.ToDecimal(Parsexmldoc.range3max_good) * 100;
    x2min = Convert.ToDecimal(Parsexmldoc.range2min_warning) * 100;
    x2max = Convert.ToDecimal(Parsexmldoc.range2max_warning) * 100;
    x3min = Convert.ToDecimal(Parsexmldoc.range1min_bad) * 100;
    x3max = Convert.ToDecimal(Parsexmldoc.range1max_bad) * 100;
    valuepoint = Convert.ToDecimal(Parsexmldoc.kpivalue_value_global) *
100;

    //
    //Set variables for the coordinate of the graphic indicator
    int x1s = (int)x1min + Xoffset;
    int x1e = (int)x1max + Xoffset;
    int x2s = (int)x2min + Xoffset;
    int x2e = (int)x2max + Xoffset;
    int x3s = (int)x3min + Xoffset;
    int x3e = (int)x3max + Xoffset;

    int y = yline;
    int x1Vline = (int)(valuepoint) + Xoffset;
    //
    //Set variables for min and max range threshold
    goodrangestr = Parsexmldoc.range3max_good;
    warningrangestr = Parsexmldoc.range2max_warning;
    badrangestr = Parsexmldoc.range1max_bad;
    //
    //Draw three big lines of bad, warning and good range value
    Pen myPenGood = new Pen(Color.Green, 30);
    Pen myPenWarning = new Pen(Color.Yellow, 30);
    Pen myPenBad = new Pen(Color.Red, 30);
    gs.DrawLine(myPenGood, x1s, y, x1e, y);
```

```

gs.DrawLine(myPenWarning, x2s, y, x2e, y);
gs.DrawLine(myPenBad, x3s, y, x3e, y);

//Draw value vertical line
Pen valueline = new Pen(Color.Black, 5);
gs.DrawLine(valueline, x1Vline, yline - 30, x1Vline, yline + 20);

//Draw label colored string
Brush bred = new SolidBrush(Color.Red);
Brush byellow = new SolidBrush(Color.Gold);
Brush bgreen = new SolidBrush(Color.Green);
Font fo = new Font("Arial", 10);
string range1 = "Bad range: 0-" + badrangestr;
string range2 = "Warning range: " + badrangestr + "-" +
warningrangestr;
string range3 = "Good range: " + warningrangestr + "-" +
goodrangestr;
float x1 = x3s;
float y1 = yline + 20;
float x2 = x3s;
float y2 = yline + 40;
float x3 = x3s;
float y3 = yline + 60;
gs.DrawString(range1, fo, bred, x1, y1);
gs.DrawString(range2, fo, byellow, x2, y2);
gs.DrawString(range3, fo, bgreen, x3, y3);

gs.Dispose(); //Releases all resources used by this Graphics
}

```

Il funzionamento del precedente metodo è così riassumibile. Viene eseguito un *casting* da tipo stringa a tipo decimale delle variabili statiche della classe **ParsexmlDoc** contenenti i valori minimi e massimi dei range di soglia, quindi vengono riassegnati a nuove variabili. I valori inoltre vengono moltiplicati di un fattore 100 in modo da renderli compatibili con la dimensione del grafico espressa con un intero limitato.

I tre segmenti vengono disegnati secondo il metodo **DrawLine()** che riceve come parametri un oggetto di tipo **Pen** oltre che delle 4 coordinate cartesiane per indicare l'inizio e la fine del segmento in orizzontale e verticale.

L'oggetto **Pen** stabilisce colore e dimensione i pixel della linea, in questo caso con uno spessore impostato a **30px**, così è stato possibile affiancare i tre segmenti creando la figura di un rettangolo.

Il valore misurato estratto, ovvero quello contenuto nella variabile **'kpivalue_value_global'** relativa al valore dell'attributo **'value'** dell'elemento **"KPIVALUE"** del file di risorsa *Spago4Q*, viene disegnato nuovamente con il metodo **DrawLine()**, stavolta mantenendo ferma la coordinata **X** corrispondente al **'value'** e modificando solo la coordinata **Y** per attraversare così il rettangolo dei range nella sua dimensione verticale.

Gli oggetti **Label** contenenti le cifre numeriche estratte ricevono tali valori attraverso il metodo **DrawString()** che richiede come parametri l'oggetto di tipo **Brush** per definire il colore della stringa, il tipo di font da impiegare, la stringa da scrivere e infine due valore di tipo *float* per indicare la posizione in cui inserire la stringa mediante le coordinate **X,Y** relative al grafico soprastante.

Analizzando ulteriori dettagli inerenti il metodo **ViewAllInTheForm()** della classe **Form1**, si può riscontrare l'istanza ripetuta della classe **Parsexmldoc** e di alcuni suoi metodi.

In pratica per ogni nodo, ossia per ogni elemento scelto, si ripete una sequenza di codice per l'estrapolazione dei valori cardine.

Ad esempio il codice seguente si occupa della ricerca delle due informazioni di nome attributo e di valore globale raggiunto dalla misurazione per un determinato elemento:

```
Parsexmldoc.codefind = n2;

//sets the variable for the selection of the first parent node node

pd.ReadElemGlobal("name", Parsexmldoc.codefind);
//Selection and extraction of a none
pd.ThresholdRead(); //Search of the range threshold
this.lblValueEconomic.Text = Form1.n2 + " : " +
Parsexmldoc.kpivalue_value_global;
//shows the values found in the label: name parent node and global value

this.DataGridVG(); //Initialize all DataGridView object
Form1.Xoffset = this.lblValueEconomic.Location.X;
//sets the new horizontal distance from the label that shows the global value
Form1.yline = this.lblValueEconomic.Location.Y + 45;
```

```
//sets the new vertical distance from the label that shows the global value
    this.DrawGraph(); //Draw in the window form graphic indicators
```

Successivamente, sempre nello stesso metodo, si ripete la seguente sequenza per estrarre la lista dei sotto-nodi di ogni nodo primario:

```
Parsexmldoc.subcodefind = n2; //sets the variable for the selection
of the first child nodes of the forth parent node

pd.ExtractAllSubNode("name", Parsexmldoc.subcodefind); //Selection
and extraction of the all subnodes
this.dataGridEconomic.Rows.Clear(); //Clear all item of the
DataGridView
for (int n = 0; n < Parsexmldoc.subnode_name.Count; n++) //fills the
DataGridView with all elements taken from the ArrayList
{
    this.dataGridEconomic.Rows.Add(Parsexmldoc.subnode_name[n],
Parsexmldoc.subnode_code[n]); //Add each name element in the control DataGridView
}
```

Riassumendo, relativamente alla classe *Form1* vengono svolti i seguenti passi sequenziali come pseudo-codice:

- Assegnamento alla variabile il valore dell'attributo da ricercare
- Esecuzione del metodo *ReadElemGlobal(string elemsearch, string valuesearch)* per la ricerca del nome attributo e del valore dell'attributo univoci per un nodo "MODELINSTANCENODE"
- Esecuzione del metodo *Threshold()* per la ricerca e la definizione delle soglie dei range per il nodo "MODELINSTANCENODE" precedentemente trovato
- Visualizzazione testuale con una *Label* delle due informazioni trovate
- Inizializzazione del *DataGridView* che servirà anche a contenere ed a mostrare la lista dei sotto-nodi del nodo trovato
- Impostazione di un offset per distanziare e posizionare il grafico in relazione alle *Label* già presenti nella *Form*

- Generazione del grafico attraverso il metodo *DrawGraph()* per la visualizzazione del valore globale dell'elemento
- Assegnamento alla variabile stringa del valore dell'attributo di un nodo per cui trovare i sotto-nodi
- Estrazione di tutti i sotto-nodi mediante il metodo *ExtractAllSub(string paramsearch, valuesearch)* dove *paramsearch* è un nome attributo e *valuesearch* è il valore della variabile assegnata al passaggio precedente
- Svuotamento del *DataGridView* per consentire il successivo riempimento mediante un ciclo *For* che preleva i nomi dei sotto-nodi contenuti in un *ArrayList*

Per effettuare il *download* di una risorsa, ovvero di un file **XML** dal *server Spago4Q*, è indispensabile fare una precisa richiesta **web**. Nell'espletamento di tale funzione viene adoperata la classe *OpenResource* al momento di questa scrittura in piena fase di miglioramento, ma che tuttavia funziona egregiamente tranne che per la limitazione temporanea di dover stabilire a priori quale documento prelevare con esattezza e quali percorsi assoluti raggiungere, sia per *l'upload* che il *download*.

Il listato del metodo è il seguente:

```
public void ReqExecuteDocument()
{
    //Set url web service
    string url = @"http://spago4q.org/Spago4Q/sdk/DocumentsService?WSDL";

    //Upload file xml request
    System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
    doc.Load(pathrequestxml);

    //assignment a string the entire contents of the files required for
the body
    string content = doc.InnerXml;

    //Creation of the web request
    HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
```

```

//Set header of the request
req.Method = "POST";
req.ContentType = "text/xml; charset=UTF-8";
req.Accept = "application/soap+xml, application/dime,
multipart/related, text/*";
req.UserAgent = "Axis/1.4";
HttpRequestCachePolicy nocache = new
HttpRequestCachePolicy(HttpRequestCacheLevel.NoCacheNoStore);
req.CachePolicy = nocache;
req.Proxy = new WebProxy("http://spago4q.org", false);
req.Headers.Add("SOAPAction", "\"\"");
req.Headers.Add("Pragma: no-cache");
byte[] buffer = new System.Text.ASCIIEncoding().GetBytes(content);
req.ContentLength = content.Length;

//Creation of the stream to send xml request in the body
Stream requestStream = req.GetRequestStream();
requestStream.Write(buffer, 0, buffer.Length);
requestStream.Close();

//Creation of the object response web request
WebResponse res = (HttpWebResponse)req.GetResponse();

//creation of the stream containing the response of the request xml
to be saved to a local file
StreamReader sr = new StreamReader(res.GetResponseStream());
StreamWriter oWriter = new StreamWriter(Path.Combine(pathresponsexml,
"resExecuteDoc.xml"));
string strfortxt = sr.ReadToEnd();

//Delete unnecessary rows to XML
int lenghtstr1 = strfortxt.Length;
int startxml1 = strfortxt.IndexOf("<?xml version=\"1.0\"
encoding=\"ISO-8859-1\"?>");
strfortxt = strfortxt.Substring(startxml1 - 1, lenghtstr1 -
startxml1);
int lenghtstr2 = strfortxt.Length;
int startxml2 = strfortxt.IndexOf("</DOCKPI>");
strfortxt = strfortxt.Substring(1, startxml2 + 8);
oWriter.Write(strfortxt);

//Close all streams and the response
sr.Close();
oWriter.Close();
res.Close();
}

```

Le variabili di percorso impostate in questo caso sono due:

```

public string pathrequestxml = @"../../requestXML\4_REQ_executeDocument.xml";
public string pathresponsexml = @"../../resourceXML";

```

Il funzionamento in pseudo-codice è il seguente:

- Impostazione della **URL** del server che ospita la risorsa
- Incapsulamento in un oggetto di tipo *XmlDocument* del corpo della richiesta da inviare nella *web request* intera. Questo passaggio è davvero molto importante, poiché il tipo di file da inviare come **BODY** è di tipo **XML** ed è costruito a partire dal risultato di altri servizi precedentemente richiesti per ottenere una serie di informazioni d'assemblare seguendo un preciso modello
- Assegnamento dell'intero contenuto del file **XML** precedentemente caricato nell'oggetto *XmlDocument* ad una variabile stringa
- Creazione dell'oggetto di tipo *HttpWebRequest* e relativa inizializzazione con parametro **URL** assegnata alla variabile *'url'*
- Impostazione di tutti i parametri della *Header* per la richiesta web i quali richiedono specifici valori
- Creazione di un oggetto di tipo *Stream* per l'invio dell'intero contenuto della *request* mediante uno *stream* sequenziale di scrittura dotato di buffer
- Istanziamento della classe *WebResponse* per l'oggetto che conterrà la risposta della *web request*
- Creazione di un nuovo oggetto di tipo *StreamReader* che preleverà tutta la *web response* dell'oggetto dello step precedente
- Creazione di un ulteriore oggetto di tipo *StreamWriter* che andrà a scrivere il risultato della richiesta integralmente in un file locale di tipo **XML**
- Dato che il file **XML** salvato non è del tutto corretto per delle linee di testo superflue presenti all'inizio ed alla fine, e che quindi non è conforme allo standard **XML**, procedere ad una sorta di normalizzazione eliminando le righe ed i caratteri inutili così da ottenere un documento **XML** idoneo all'analisi
- Chiudere la *web response* e gli *stream* precedentemente aperti

NB. Per una comprensione maggiore si rimanda alla consultazione del capitolo 4 paragrafo 4 “**CREAZIONE WEB REQUEST HTTP**”

Capitolo 7

CONCLUSIONI

Il software realizzato corrisponde adeguatamente alle specifiche volute in conformità alle idee di sviluppo. La facilità d'utilizzo da parte di chiunque, e non solo per i programmatori esperti, è stata ottenuta mediante una veste grafica semplice e intuitiva.

La sua complessità, intesa anche come quantità di risorse necessarie al sistema su cui eseguirlo, è abbastanza esigua e solo nella fase di analisi e lettura del file **XML** si nota qualche rallentamento con **PC** dotati di scarsa **RAM** o comunque con un **HW** più datato.

L'espandibilità e le migliorie apportabili offrono molte idee, in particolare per le funzioni della **DLL** che potranno e dovranno essere senza dubbio perfezionate data la stabilità richiesta, perciò sarà oggetto nel prossimo futuro di uno studio approfondito per la derivazione di tutte le classi al fine di arricchire l'applicazione.

L'obiettivo primario per il progetto è quello di poter adattare l'intera soluzione sui sistemi hardware con differenti sistemi operativi, aspetto realisticamente raggiungibile poiché basato sulla piattaforma **.Net**, ma attualmente poco sperimentabile dato che per necessità personali non è stato possibile disporre degli strumenti adatti a programmare in ambienti diversi da **MS Windows**.

7.1 SVILUPPI FUTURI

Tra le prossime funzionalità che saranno auspicabilmente aggiunte è possibile elencarne attualmente alcune:

- Creazione di *report* di stampa completi di tutti i valori misurati e dei grafici di riferimento con impaginazione di stampa personalizzabile
- Implementazione funzione di scambio delle informazioni acquisite mediante *e-mail*, oppure alternativamente con un sistema di *chat* in tempo reale, al fine di consentire la condivisione dei dati fra collaboratori situati in luoghi remoti
- Maggiore integrazione di questa soluzione con la piattaforma *SpagoWorld* e i **tool** che essa offre
- Adattamento dell'intera soluzione all'uso dell'ambiente *SharePoint* per condivisione dati progetto con siti web generati con tale **SW** in ambito aziendale

7.2 LIMITAZIONI DA RISOLVERE

I due limiti principali riscontrabili nella versione attuale del programma e che avranno priorità di risoluzione sono:

- Utilizzo pieno dei *web service* attraverso il richiamo diretto dei metodi con parametri messi a disposizione dal descrittore **WSDL** il quale è stato scritto per una maggiore compatibilità con un ambiente di sviluppo *Java*, dunque causa di difficoltà nell'uso con **VS 2010 PRO**
- Modifica d'impostazione dell'aspetto dei grafici di visualizzazione dei dati, poiché mediante le librerie grafiche incluse in **.Net** non è facilmente possibile generare figure diverse e ridimensionarle a *run-time*

Dopo ogni lavoro svolto e meccanismo acquisito è comunque auspicabile di aver ottenuto quanto meno i minimi requisiti che **Engineering Group** necessariamente dovrà valutare dato il suo prestigio professionale.

Capitolo 8

RIFERIMENTI

Bibliografia

- 1: Luca Cabibbo, Introduzione al Data Warehousing ed alla Progettazione di Data Warehouse Dimensionali, , <http://www.dia.uniroma3.it/~cabibbo/dw/>
- 2: Davide Taibi, Principi di UML e WebML, <http://www.taibi.it/public/2006/09/UML&WebML.pdf>
- 3: Wikipedia, Norme della serie ISO 9000, , http://it.wikipedia.org/wiki/Norme_della_serie_ISO_9000
- 4: NATO Stanags, Standardization Agreements, , <http://www.nato.int/cps/en/natolive/stanag.htm>
- 5: Wikipedia, ISO 14000, , http://it.wikipedia.org/wiki/ISO_14000
- 6: Wikipedia, Service-oriented architecture, , http://it.wikipedia.org/wiki/Service-oriented_architecture
- 7: Engineering, SpagoWorld, , <http://www.spagoworld.org/xwiki/bin/view/Spago4Q/>
- 8: Qualipso.org, Qualipso Trust and Quality in Open Source System, , <http://www.qualipso.org/documents>
- 9: CodeSwat, Swat4J, , www.codeswat.com
- 10: SQUALE, Squale Software Quality Enhancement, , <http://www.squale.org/>
- 11: Wikipedia, ITIL, , <http://it.wikipedia.org/wiki/ITIL>
- 12: Wikipedia, CMMI, , <http://it.wikipedia.org/wiki/CMMI>
- 13: Wikipedia, GQM, , <http://it.wikipedia.org/wiki/GQM>
- 14: José Cordeiro, Enterprise Information Systems: 11th International Conference, ICEIS 2009, 2009
- 15: OMG's, , , <http://www.omg.org/mof/>
- 16: W3School, Web Services Tutorial, , <http://www.w3schools.com/webservices/default.asp>
- 17: UDDI-XML.org, About UDDI, , <http://uddi.xml.org/uddi-101>
- 18: Mayo Joseph, C# Tutto&Oltre, 2002
- 19: Mahesh Chand, Creating C# Class Library (DLL) Using Visual Studio .NET, , <http://www.c-sharpcorner.com/uploadfile/mahesh/dll12222005064058am/dll.aspx>