**CAPITOLUL 3. Specificațiile aplicației**

3.1 Generalitați

În capitolul anterior au fost prezentate doar cateva dintre cele mai utilizate tipuri de diagrame folosite in domeniul IT, marea majoritate a diagramelor folosite de catre companiile ce activeaza in domeniu fiind concepute dupa un standard intern, adaptat domeniului în care activează acea companie. În plus acest tip de prezentare a informației (schematic prin folosirea diagramelor) a devenit extrem de popular și in discipline ce nu au legatură cu disciplinele ingineresti cum ar fi in psihologie, actividați didactice sau chiar în economie.

Putem identifica nevoia clară pe piata a unei aplicații ce poate facilita managementul, crearea si editarea de diagrame. Fara dubii exista o serie de aplicații ce indeplinesc intr-o oarecare masură nevoile unui anume segment de piața, acesta fiind și punctul slab al aplicatiilor de acest tip: incapacitatea de adaptare la grupuri diferite de utilizatori .

Deseori chiar și aplicatiile destinate desenarii de diagrame pentru un domeniu specific, cum ar fi IT-ul, sunt foarte rigide și neadaptate la nevoile utilizatorului. Cel mai bun exemplu fi aplicatia StarUML. Aceasta este o aplicate open source destinată desenării de diagrame de tip UML (Unified Modeling Language) gratuită și relativ simplu de folosit. Este foarte populară în mediul academic, mai specific folosită de catre elevi și studenți. Cu toate că această aplicație este destul de reusită ca și concept și implementare are o problema majora: neadaptarea la limbajul de programare ce urmeaza a fi folosit pentru implementarea sistemului. Prin urmare unele tipuri, constructii sau legaturi intre obiecte (definite in limbajul de programare lipsesc din diagrama de clase si vice-versa).

Cu toate ca limbajele orientate pe obiect au o structura relativ asemanatoare in materie de tipuri si relatii, exista diferențe care la prima vedere ar parea neimportante și nesemnificative insa care pot crea ambiguitați in etapa de implementare. Cu toate că standardul UML este global recunoscut, se permit mici modificări a unei entitați create pentru a se adapta mai bine la domeniul aplicației, ceea ce nu este suportat de marea majoritate a aplicațiilor de desenare a diagramelor de tip UML. Aceste mici modificări, necesare pentru a pastra consistenta intre diagramele de clase (de exemplu) si codul efectiv sunt ignorate, ceea ce va produce erori în pașii urmatori design-ului.

Pentru a demonstra ideea voi reveni la exemplul anterior. Creatorii StarUML afirmă că aplicatia lor suportă designul diagramelor de clase pentru limbajul Java. Cu toate că afirmația este partial adevarată, StarUML nu are specificat un tip de clasă cum ar fi “Clasele abstracte”. Astfel o clasa abstractă de obiecei prin convenție va fi prefixata cu <Abstract> in numele acesteia, convenție care de altfel nu exista in generatorul de STUB-uri prezent în aplicatie si prin urmare toate functionalitațile prezente la nivel de generare de cod sau verificare de model a diagramei, acestea fiind inoperabile.

O altă problemă prezentă la aplicatiile disponibile ar fi multitudinea de functionalitați oferite. Afirmația de mai sus poate fi controversata și poate contraintuitiva. Pentru a fi cat mai explicit voi da un alt exemplu de aplicație foarte des folosită de catre inginerii software: VisualParadigm.

Visual Paradigm este un tool folosit pentru desenarea diagramelor de tip UML având ca și client ținta ingineri ce activează la nivel profesional în domeniul proiectării software. Faptul că această aplicație a primit 5 premii majore in domeniul IT demonstrează faptul ca este o aplicație reusită, însa datorită faptului că publicul ținta al acestei aplicații sunt proiectanții de sistem seniori se oferă un numar atât de mare de funcionalitați încat un utilizator cu un nivel de experiență redus în domeniul proiectării sistemelor s-ar confrunta cu dificultați majore în identificare funcționalităților necesare acestuia. Interfata grafica extrem de încarcata fac identificarea functionalitaților cheie a aplicației extrem de dificilă, astfel un utilizator neinițiat, care ar vrea o solutie rapidă, nu își va putea crea acea soluție fară o pregatire riguroasă in folosirea aplicației. Figura de mai jos prezintă o captură de ecran ce conține interfața grafică oferită de către aplicația VisualParadigm.

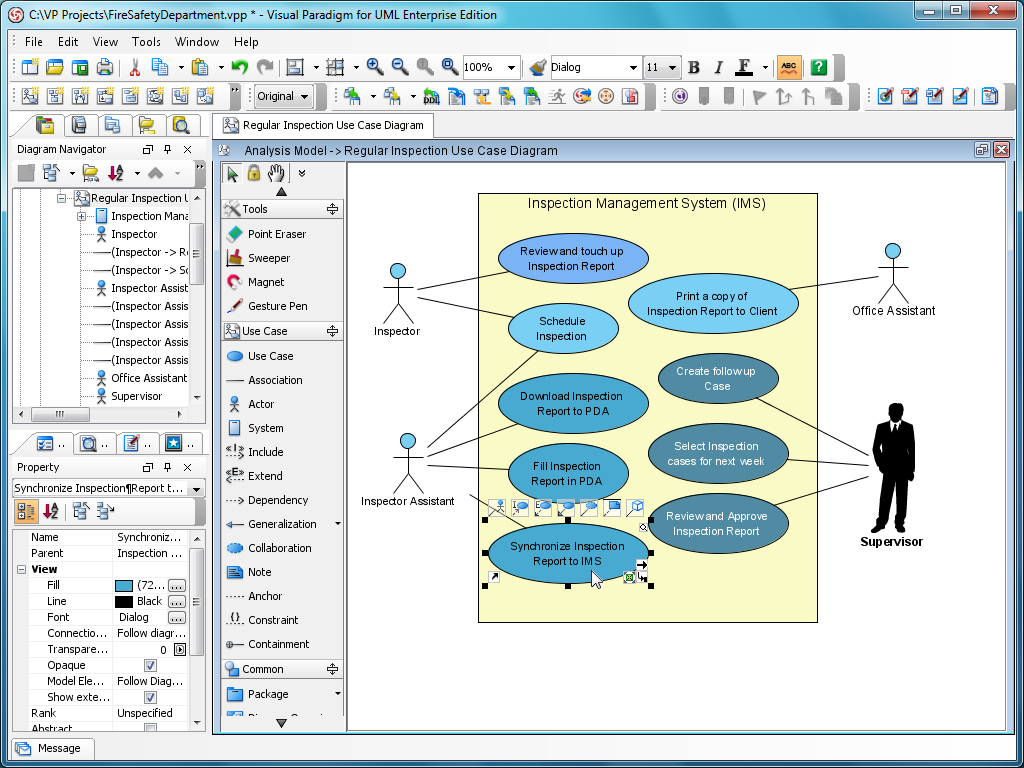


Figura 3.1. Interfață grafică Visual Paradigm

Există exemple numeroase care prezintă avantajele și dezavantajele folosirii aplicațiilor deja existente. În timp ce unele oferă functionalitați avansate dar sunt greu de utilizat, altele oferă functionalitați usor de utilizat dar mult prea simpliste pentru a crea o diagramă bine definită.

Prin aplicatia prezentată în această lucrare s-a încercat implementarea unui instrument ce va adresa aceste probleme. Printr-o separare clară între interfața grafică, controlul aplicației și functionalitațile oferite de catre aceasta a rezultat o aplicație scalabilă la cerințele utilizatorului, independent de domeniul în care activează sau de gradul de experiență al acestuia.

3.2 Specificatii si constrangeri

În aplicația prezentată s-a incercat păstrarea avantajelor altor aplicații de pe piață și eliminarea dezavantajelor acestora, fapt posibil datorită specificațiilor bine definite. În acest capitol se vor descrie specificațiile de nivel înalt ale aplicatiei și motivația implementării acesteia.

Un prim considerent major în construirea aplicației a fost **scalabilitatea** acesteia. Scalabilitatea se referă la comportamentul consistent al aplicatiei în diverse condiții de lucru. Aplicatia livrata trebuie sa fie suficient de scalabilă pentru a rula in parametri normali fie in cazul utilizării de către un utilizator pe un calculator personal, fie in cazul unei intreprinderi in care aplicatia este folosită concomitent de către câteva sute de utilizatori concomitent, folosind un server de baze de date pentru date. Scalabilitatea unui sistem software are la baza gama de tehnologii și biblioteci folosite pentru dezvoltarea acestuia.

Adaptabilitatea redusa a unei aplicatii ii reduce drastic durata de viata, in special in cazul unei aplicatii orientata pe utilizator privat (și nu pe mediu enterprise) cum este aplicația prezentată. Astfel putem deduce un alt set de constrangeri de care va trebui tinut cont: **adaptabilitatea la limbaj și standarde**. Cu fiecare schimbare adusă unui limbaj deja existent, apariție de standard nou în modul de desenare a diagramelor, sau apariția unui limbaj total nou, aplicația trebuie actualizată pentru a permite acomodarea acestor schimbari.

O aplicație care are un proces de instalare și configurare complicat este în fond o aplicatie cu o intretinere dificilă. În cazul aplicațiilor foarte dependente de platformă și cu multe fisiere de proprietați, variabile de sistem sau căi de acces, se poate ajunge la situatia în care marea majoritate a clientilor fie nu vor instala aplicatia și vor căuta o alternativă mai simplu de configurat, fie vor apela la producatorii aplicației pentru asistența, ceea ce va creste costurile de întreținere. În concluzie, este nevoie ca aplicația dezvoltată sa fie **ușor de configurat.**

Cu toate ca liderul pe piața în materie de sisteme de operare ramâne Microsoft cu produse din seria Windows, există un segment de piața destul de extins ce folosește sisteme de operare alternative pentru stațiile personale de lucru, de exemplu Ubuntu sau MacOS. Pentru a nu pierde acel segment de utilizatori se propune ca aplicația sa fie **independenta de platforma si de arhitectura interna a sistemului.**

Studiile arata ca aplicațiile cu interfața grafică prea încarcată sau care nu evidențiază funcționalitațile cele mai relevante pentru utilizatori iși pierd popularitatea. Pentru a nu intra în acea categorie de aplicații se propune crearea unei **interfețe grafice prietenoase și configurabile.**