**Analiza sistemului software ”Toolset”**

Istoric document

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Description** | **Date** |
| 1.1 | Costean Adrian | Crearea unei prime analize a sistemului ”Toolset” si definirea unor cerinte cheie in dezvoltarea acestui sistem | 12.10.2017 |
|  |  |  |  |
|  |  |  |  |

Cuprins

[1 Introducere 4](#_Toc495444049)

[1.1 Obiectiv 4](#_Toc495444050)

[1.2 Scopul 4](#_Toc495444051)

[1.3 Limitari 4](#_Toc495444052)

[2 Cerinte 4](#_Toc495444053)

[2.1 Cerinte functionale 4](#_Toc495444054)

[2.1.1 Sisteme de operare 4](#_Toc495444055)

[2.1.2 Configurare 5](#_Toc495444056)

[2.1.3 Interfata grafica 5](#_Toc495444057)

[2.1.4 Suport command-line 5](#_Toc495444058)

[2.1.5 Sisteme de versionare 5](#_Toc495444059)

[2.1.6 Suport compilatoare 6](#_Toc495444060)

[2.1.7 Conventie structurare proiect 6](#_Toc495444061)

[2.1.8 Sistem de dependinte 6](#_Toc495444062)

[2.2 Cerinte non-functionale 7](#_Toc495444063)

[2.2.1 Cerinte hardware 7](#_Toc495444064)

[2.2.2 Cerinte software 7](#_Toc495444065)

[2.2.3 Disponibilitate 7](#_Toc495444066)

[2.2.4 Interfata grafica 7](#_Toc495444067)

[2.2.5 Exactitate 7](#_Toc495444068)

[2.2.6 Viteza 7](#_Toc495444069)

[2.2.7 Instructiunile de utilizare 7](#_Toc495444070)

[2.2.8 Portabilitate 7](#_Toc495444071)

[2.2.9 8](#_Toc495444072)

[2.3 Cerinte care influenteaza arhitectura 8](#_Toc495444073)

[3 Componente 8](#_Toc495444074)

[3.1 Descopmunere in componente 8](#_Toc495444075)

[3.2 Analiza functionala a componentelor 8](#_Toc495444076)

[3.2.1 CM 8](#_Toc495444077)

[3.2.2 Conector CM 9](#_Toc495444078)

[3.2.3 Interfata grafica 9](#_Toc495444079)

[3.2.4 Interfata command-line 9](#_Toc495444080)

[3.2.5 String parser 9](#_Toc495444081)

[3.2.6 Executor 9](#_Toc495444082)

[3.2.7 Livrator 9](#_Toc495444083)

[3.3 Argumentare 9](#_Toc495444084)

[4 Prezentare sistem 9](#_Toc495444085)

[4.1 Perspectiva 1 (cu diagrame) 9](#_Toc495444086)

[4.2 Perspectiva 2 (cu diagrame) 9](#_Toc495444087)

[5 Indicatori de calitate 10](#_Toc495444088)

[5.1 Scalabilitatea 10](#_Toc495444089)

[5.2 Timpul de raspuns al interfetei 10](#_Toc495444090)

[5.3 Numarul de cereri simultane 10](#_Toc495444091)

[6 Tehnologii middleware 10](#_Toc495444092)

[7 Modele si stiluri arhitecturale 10](#_Toc495444093)

[7.1 Stil arhitectural 10](#_Toc495444094)

[8 Scenarii de validare 11](#_Toc495444095)

# Introducere

## Obiectiv

In cadrul companiei Continental, departamentul IBS ( Interior, Body & Security) ocupa un rol important in dezvolatarea software si hardware a elementelor ce tin de siguranta pasagerilor ( airbag-uri, sisteme de avertizare a soferului, frane etc). In ultimii 20 de ani asistam la o dezvoltare tehnologica exponentiala, prin urmare si metodele, programele, utilitarele trebuie sa evolueze in acest ritm pentru a asigura o aliniere cu stadardele si concurenta.

## Scopul

Comparand activitatea unui programator de acum 20 ani, cu activitatea unui programator de acum, putem observa foarte bine ca procesele au devenit din ce in ce mai complexe, cerintele care tin de calitatea codului scris, de validarea acestuia, de eficienta s.a.m.d sunt din ce in ce mai mari, de asemenea raportul de timp in care un produs trebuie livrrat e mult mai mic. Tinand cont de aceste constrangeri, e aproape imposibil ca un programator sa combine toate aceste elemente si sa tina evidenta lor, de aceea e nevoie de un utilitar care sa automatizeze pe cat posibil operatiile repetitive, sa adauge functionalitati de verificare a codului, de validare a acestuia si de intocmire a unor analize de performanta si calitate si multe alte elemente care sa asigure o eficientizare si armonizare a procesului de dezvoltare software.

## Limitari

Pentru a evidentia necesitatea implementrii sistemului descirs in acest document, putem porni spre exemplu de la un simplu proiect C/C++. Un dezvoltator trebuie sa ruleze intai manual generatoare de cod, sa compileze manual sursele C asigurandu-se ca adauga toate optiunile de compilator, sa ruleze teste de calitate, sa ruleze un linkeditor care sa asambleze intreg proiectul intr-o librarie finala care va fi incarcata pe componenta hardware, sa isi faca copie de siguranta a proiectului intr-o baza de date etc. Munca in echipa devine destul de dificila, impartirea task-urilor e greu de realizat, nu exista o siguranta si un proces bine definit prin care sa poata fi verificata calitatea produsului final si multe alte limitari care apar in timpul procesului de dezvoltare.

# Cerinte

## Cerinte functionale

### Sisteme de operare

Sistemul de operare suportat de produs trebuie sa fie Windows cu posibilitate de extindere la Linux.

### Configurare

Fiecare proiect va trebui sa poata avea un fisier de configurare global unde va putea modifica urmatorii parametrii

* versiuni de tool-uri (interne/externe)
* tipul de compilator
* parametrii pentru compilator sau linker
* fisierele sursa care vor fi incluse/excluse din proiect
* Activarea/dezactivare conexiunii la sistemul de versionare

### Interfata grafica

Interfata grafica trebuie sa contina urmatoarele elemente:

* structura proiectului unde userul trebuie sa aiba posibilitatea de a naviga printre folderele si fisierele proiectului.
* un editor de cod C/C++ unde userul va scrie codul. Aditional, editorul de text va avea functionalitate automata de marcare a erorilor de calitate si de sintaxa.
* un navigator de cod care sa faciliteze navigarea in cod (navigatorul va contine functiile principale si alte elemente principale).
* butoane care sa asigure urmatorele functii:
* Compilare proiect
* Recompilare proiect
* Compilare fisier
* Verificare de calitate pe proiect
* Verificare de calitate pe fisier
* Generare de raport pe proiect
* Generare librarie finala
* Reimprospatarea proiectului ( in caz ca exista modificari facute local, dar nu din interfata grafica)
* o consola care sa reflecta rezultatul fiecarei actiuni rulate din meniu

### Suport command-line

Orice actiune efectuata din interfata grafica trebuie sa fie suportatat si din command-line.

### Sisteme de versionare

Produsul “Toolset” trebuie sa asigure suport pentru conexiunea cu sistemele de versionare Sinergy, IMS, Perforce sau GitHub unde este stocat proiectul C/C++ pentru a asigura o mai buna disciplina.

Conectarea se face cu credentialele userului care ruleaza produsul, iar setarile de conectare trebuie sa poata fii configurabile dintr-un fisierul de configurare (2.1.2) de catre user. O data conectat la sistemul de versionare, produsul trebuie sa ruleze querry-uri pentru a colecta informatiile legate de proiectul care se afla in mediul de lucru.

Aceste informatii trebuie sa cuprinda :

* Versiunea fiecarui fisier prezent in proiect
* Versiunea curenta a proiectului
* Numele real al proiectului
* Rolul userului conectat

### Suport compilatoare

Proiectele sunt concepute in functie de hardware pentru anumite tipuri de microprocesoare, din cauza aceasta, produsul trebuie conceput in asa masura incat userul sa poata configura compilatorul din fisierul de configurare globala (2.1.2) . Suportul trebuie asigurat pentru compilatoarele din familia “**gh”**, “**arm**” si “**iar**” dar sa aiba posibilitatea de a fi extins daca apare nevoia aceasta.

### Conventie structurare proiect

Trebuie definita o conventie astfel incat fiecare proiect sa respecte aceeasi structura, astfel:

* Source – va contine fisierul de configurare global
* La nivelul source ( dar nu mai sus) vor exista folderele cu fisierele \*.c/cpp si \*.h

iar pentru programul “Toolset”, acesta va genera un set de foldere generice care vor facilita accesul la informatiile necesare, astfel la acelasi nivel cu folderul “source” vor fi generate urmatoarele foldere:

* Obj – aici se vor genera fisiere si rapoarte temporare
* Src – aici se va face a copie a fisierlor \*.c/\*.cpp si \*.h care vor fi utilizate in configuratia curenta
* Out – aici se vor livra rezultate actiunilor executate ( ex. : rapoarte de calitate, libraria finala, rapoarte individuale de analiza de cod)

### Sistem de dependinte

In cazul unei modificari in proiect, timpul necesar recompilarii intregului proiect trebuie eliminat prin crearea unui sistem de depndinte care sa detecteze care fisiere din proiect sunt afectate de schimbare, astfel in momentul in care butonul “Compilare proiect” sau “Verificare de calitate pe proiect” e apasat, doar asupra fisierelor modificate sa fie executata actiunea.

## Cerinte non-functionale

### Cerinte hardware

Programul va rula fara probleme pe PC-uri care au ca si configuratie hardware cel putin 4gb ram si un procesor cu un singur nucle la o frecventa minima de 2.0 Ghz

### Cerinte software

Aditional pe langa programul “Toolset”, nu va fi nevoie de instalarea manuala de catrea utilizator a altor programe

### Disponibilitate

Programul va trebui sa fie disponibil pe o retea globala de unde fiecare user va putea sa il copieze pe local

### Interfata grafica

“A user interface is like a joke, if you have to explain it, it’s not that good” – autor necunoscut

Interfata cu userul trebuia sa fie una cat mai simpla si mai sugestiva.

Ca si asezare vizuala, interfata trebuie impartita astfel:

* In partea stanga trebuie sa apara navigatorul de proiect
* In partea centrala a interfetei trebui sa fie pozitionat editorul de cod C/C++
* In partea dreapta sa fie pozitionat navigatorul de cod
* In partea superioara sa fie pozitionate butoanele
* In partea inferioara trbuie sa apara consola

### Exactitate

Rezultatele rapoartelor trebuie sa fie concludente si trebuie sa raporteze starea proiectului si a fisierelor continute de acesta intr-un procent de 100%. La fel si libraria finala va trebui sa reflecte exact codul care a fost compilat.

### Viteza

Generarea raportului de calitate per proiect nu trebuie sa depaseasca 8 ore.

### Instructiunile de utilizare

Instructiunile de utilizare ale produsului trebuie sa fie coerente si clare pentru a nu se realiza comenzi gresite si trebuie sa fie traduse in limbile franceza, germana si engleza.

### Portabilitate

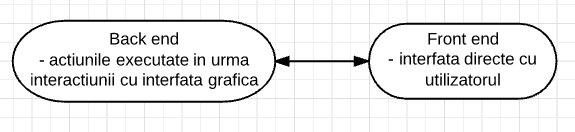
Un mediu de lucru creat pe o statie trebuie sa fie utilizabil si functional daca e transferat pe o alta statie.

### 

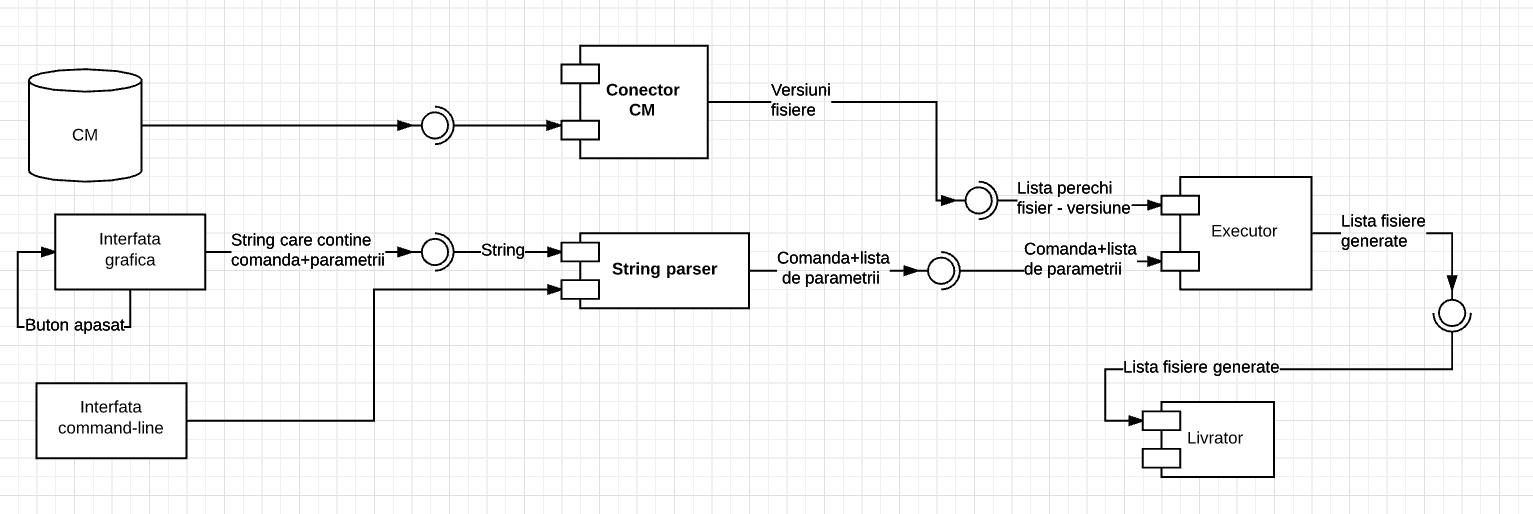
## Cerinte care influenteaza arhitectura

* 2.1.2 Configurare
* 2.1.3 Interfata grafica
* 2.2.1 Cerinte hardware
* 2.2.8 Portabilitate

# Componente



## Descopmunere in componente



## Analiza functionala a componentelor

### CM

Reprezinta baza de date/configuration management unde e stocat proiectul si de unde se vor colecta informatiile privitoare la versiunile fisierelor si al proiectului.

### Conector CM

Aceasta componenta va face legatura intre baza de date/configuration management si program. Aceasta componenta este responsabila de colectarea informatiilor necesare in rapoarta, precum versiunile de fisiere.

### Interfata grafica

Aceasta componenta reprezinta interfata cu utilizatorul din care user-ul poate lansa actiunile pe care doreste sa le execute programul. Apasarea oricarui buton din interfata va genera un sir de caractere care va contine de fapt programul necesar executarii cu o lista de parametrii.

### Interfata command-line

Aditional, utilizatorul poate executa si din command-line orice actiune ruleaza din interfata grafica, cu conditia sa adauge toti parametrii corespunzatori.

### String parser

Aceasta componenta primeste ca si input sirul de caractere din command-line sau din interfata grafica si il parseaza extragand intr-un format asteptat de catre Executor aplicatia care se doreste a fi rulata si parametrii cu care se va executa aceasta.

### Executor

Aceasta componenta reprezinta unitatea de baza a programului. Aici se iau deciziile in functie de aplicatia care se doreste a fi executata si parametrii care s-au setat. Dupa executia fiecarei aplicatii, o lista de fisiere generate e pasata mai departe componentei responsabile de livrarea fisierelor in diferite foldere.

### Livrator

Aceasta componenta primeste lista de fisiere generate si le copiaza din locatia curenta in locatia pre-stabilita prin conventie la punctul 2.1.7

## Argumentare

# Prezentare sistem

## Perspectiva 1 (cu diagrame)

## Perspectiva 2 (cu diagrame)

# Indicatori de calitate

## Scalabilitatea

Sistemul trebuie sa poata fi scalat astfel incat sa permita distribuirea la 10000 de utilizatori aflati in locatii diferite fara a creste costul de instalare si configurare.

## Timpul de raspuns al interfetei

Actionarea asupra oricarui buton din interfata trebuie sa lanseze insantaneu procesul reprezentat de acesta. De asemenea, output-ul generat de catre procesul lansat in executie trebuie sa fie colectat si raportat in timp real in consola.

## Numarul de cereri simultane

Numarul de cereri de actiuni simultane/proiect trebuie sa fie unic, iar numarul de cereri de actiuni simultane/proiecte diferite nu trebuie sa fie o constrangere a programului “Toolset” ci a resurselor hardware ale statiei.

# Tehnologii middleware

(Am cautat pe internet si am gasit multe definitii pentru “tehnologii middleware”. Nu am reusit insa sa fac o mapare exacta intre ce folosesc in program si definitie, din acest motiv am lasat aceasta sectiune necompletata)

# Modele si stiluri arhitecturale

## Stil arhitectural

Aplicatia din punct de vedere structural se incadreaza cel mai bine in stilul arhiectural “monolithic application”(aplicatie monolit). Practic toate legaturile dintre componente, toate apelarile de tool-uri externe ( incluzand adaptarile si operatiile necesare procurarii inormatiilor necesare), toate functionalitatile sunt in strasna legatura una cu alta si componentele chiar daca pot fi separate la nivel de analiza functionala, acestea se afla intr-o stransa dependenta unele de altele si nu se poate face o inlocuire a oricarei componenta cu o alta.

Din punct de vedere al memoriei partajare, aplciatia se incadreaza cel mai bine in pattern-ul arhitectural “rule-based system”(sistem bazat pe reguli). Practic din configuratia globala combinand cu cunostintele aplicatiei, sistemul va sti sa aleaga ce e necesar a fi rulat la momentul in care a venit o cerere din partea userului.

# Scenarii de validare