# Cuprins

# 1 Introducere

## 1.1 Scopul documentui

Acest document are rolul de a descriere funcționalitatea unei produs software ce reprezintă un sistem de binarizare a imaginilor.

Astfel, se dorește ca sistemul prezentat mai departe să ofere o variantă cât mai clară a unui document vechi ce a fost convertit în format electronic.

Acest document poate fi folosit și înteles atât de utilizatorii aplicației și membrii echipei dezvoltatoare, cât și de posibili viitori dezvolatori, codul sursă fiind open source.

## 1.2 Definire termeni tehnici

**digitalizare -** acțiunea de *a digitaliza;* în sensul prezent in acest document - trecerea unui document fizic in mod electronic, digital

**binarizare -** transformarea unei imagini într-un format ce conține numai pixeli albi și negri

**IBS -** Image Binarization System

**BAM -** Binarization Algorithm Module

**VBAM -** Voting Binarization Algorithm Module

**pixel -** element component, de obicei foarte mic, al imaginilor grafice, ce posedă trei atribute care se pot exprima digital (numeric): culoare, opacitate (transparență) și poziție în matricea în care se divide imaginea

**BMP** - *fișier imagine Bitmap* sau formatul de fișier DIB (Device Independent Bitmap - „bitmap" independent de dispozitiv) sau pur și simplu Bitmap, este un format de fișier imagine de tip rastru folosit pentru a stoca imagini digitale independent de dispozitivul de afișare (cum ar fi un adaptor grafic)

**TIFF -** Tagged Image File Format

## 1.3 Rezumat document

Capitolul 1 conține o descriere a documentului, conținutul, și scopul creării acestuia.

Capitolul 2 cuprinde descrierea, nu foarte amplă, a produsului prezentat in acest document: necesitatea aplicației, funcțiile sale, precum și beneficiile aduse celor ce utilizeaza aplicatia.

Capitolul 3 prezintă structurile de date importante și formatul fișierelor folosite în cadrul soluției.

Capitolul 4 prezintă arhitectura sistemului – atât descriptiv, cât și sub forma unei diagrame de arhitectură.

Capitolul 5 prezintă ferestrele aplicației și succesiunea lor în cadrul sistemului.

Capitolul 6 identifică componentele critice ale aplicației (componente a căror performanță influențează decisiv performanța globală a aplicației) si propune alternative de proiectare a componentelor critice (pentru a fi folosite în cazul insuccesului soluției propuse inițial).

# 2 Descriere generală aplicație

## 2.1 Situația actuală

În decursul ultimelor decenii stocarea și crearea de documente a cunoscut o tranziție de la formatul fizic, hărtie, la formatul electronic.

Deși majoritatea documentelor noi sunt stocate în ambele formate, cele vechi se găsesc doar în format fizic. Unele dintre acestea sunt documente foarte vechi și rare multe dintre ele fiind într-o stare de degradare, digitalizarea lor fiind singura modalitate de păstrare și distribuire.

## 2.2 Scopul aplicației

Digitalizarea documentelor presupune procesarea lor printr-un proces complex, datorită complexității structurii acestora cât și a diferențelor de structură de la document la document sau chiar între paginile aceluiași document. Documentele sunt în schimb

simplificate, catalogăndu-se pixelii în pixeli de fundal sau conținut prin conversia bitonală, numită și binarizare.

Pentru binarizarea documentelor s-au creat diverși algoritmi, performanțele acestora depinzând de documentele procesate.

## 2.3 Rezumat funcționalități

Scopul acestui proiect este de a se dezvolta un sistem de binarizare a imaginilor (IBS) care include două module:

- Modulul de binarizare: transformă o imagine în una binară
- Modulul de votare: folosind mai multe rezultate ale mai multor module de binarizare se compune o singură imagine binară

## 2.4 Context produs

Analiza documentelor este un proces complex care presupune mai multi pași de procesare, dar datorită probabilității mari de apariție a erorilor cele mai multe nu se aplică. În schimb se folosește o variantă alb-negru a imaginii care oferă o separare mai clară dintre fundal și conținut, catalogand pixelii în alb și negru, fundal sau conținut, metodă numită binarizare.

Din păcate obținerea unui algoritm optimal de separare a conținutului de fundal este problematică, niciun algortim cunoscut nu oferă o soluție adecvată pentru orice document.

## 2.5. Beneficiu/Valoare

Pachetul format din cele două module BAM și VBAM împreună cu algoritmii utilizați și detaliile tehnice ale acestora vor fi prezentate în concursul internațional “Document Image Binarization Contest” din cadrul conferinței ICDAR 2015.

# 3 Modelul datelor

În cadrul acestui document, prin date se înţelege fişier imagine aflat în oricare din cele două instanţe posibile: fişier de intrare, fişier de ieşire (rezultat) sau informaţie în etapa de prelucrare propriu-zisă.

## 3.1 Structuri de date globale

În proiect va exista o clasă/structură de date globală în care o sa încărcăm fişierul de intrare curent asupra căruia se vor aplica ulterior cei trei algoritmi de binarizare. Efectul benefic al existenţei acestei structuri ar fi că încărcarea datelor din fişier s-ar realiza o singură dată, în loc ca această citire să fie realizată la rularea fiecăruia dintre algoritmi.

## 3.2 Structuri de date de legătură

Stocarea în memorie a imaginilor, în timpul prelucrării, se va face folosind structuri de date bidimensionale alocate dinamic. Tipul variabilei care va stoca o imagine va fi de tip "**int**", un pixel luând valori în intervalul [0, 255]. Pentru corectitudinea datelor se doreşte ca la iniţializare valoarea pixelilor să fie 0; pentru iniţializarea implicită cu 0 se va folosi funcţia calloc.

Pentru stocarea rezultatului obţinut de *Modulul de Citire*, în urma citirii şi parsării datelor de intrare, se va folosi o structură typedef struct / o clasă.

## 3.3 Structuri de date temporare

Datele de legătură asigură fluxul de informaţie între principalele module ale aplicaţiei: Modulul de Citire si Parsare a Datelor de Intrare, Modulul de Prelucrare, Modulul de Scriere a Datelor de Ieşire.

Modulul de Citire pentru VBAM poate primi oricâte argumente de tipul "**string**" reprezentând numele unor fişiere rezultat în urma prelucrării BAM.

Ambele module de prelucrare, BAM şi VBAM, vor comunica cu modulele de citire şi scriere prin date de legătură reprezentare prin matrici; se va trimite pentru prelucrare doar matricea de pixeli şi tipul de format imagine. Tipul de format este necesar pentru a fi folosit la scrierea rezultatelor în fişierele imagine finale; această informaţie va fi un câmp în structura / clasa imagine iniţializat în procesul de parsare a datelor de intrare.

Modulul de Scriere a Datelor de Ieşire pentru BAM va citi informaţiile din structura temporară, le va formata corespunzător şi le va scrie în fişierele de ieşire. Rezultatele BAM vor fi două fişiere imagine: [output_imagine] şi [output_imagine]_c, [output_imagine] este valoarea argumentului al doilea primit la execuţie.

## 3.4 Formatul fişierelor utilizate

Aplicaţia va utiliza fişiere de tip BMP şi TIFF.

Formatul **BMP** (Bitmap):

- Extensia fişierelor: .bmp
- Compatibilitate: acest format este independent de platformă.
- Codificare:
- Structura fişierului BMP:

- antetul fişierului conţine: tipul fişierului (“BM”), mărimea fişierului în octeţi, offsetul de la care încep datele imaginii.
- structura BITMAPINFOHEADER conţine toate informaţiile despre imagine: dimensiunea structurii, lăţime/înălţime imagine, numărul de plane, numărul de biţi per pixel, modul de compresie folosit, rezoluţia pe verticală/orizontală, numărul de octeţi din imagine, numărul de culori folosite, numărul de culori importante;
- tabloul pixelilor din imagine; aceştia pot fi memoraţi necomprimat sau folosind o metodă de compresie pe 4 sau 8 biţi. Metoda în care au fost stocaţi pixelii este specificată de valoarea câmpului biCompression din antetul fişierului.

Formatul **TIFF** (Tagged Image File Format):

Acest format este comun pentru diverse aplicaţii pentru imagini, fiind formatul preferat pentru scanare şi fax.

Extensia fişierelor poate fi: .tiff, .tif

Compatibilitate: acest format este independent de platformă.

Dezavantaj: dimensiune mare. Fişierele TIFF pot fi comprimate fără pierderi (exemplu algoritm de comprimare: LZW).

Avantaje:

- calitate foarte bună; suport diferite tipuri de compresie;

- permite salvări repetate fără alterarea calităţii; este recomandat pentru aplicaţii de prelucrare a imaginilor.

Codificare: pe 24 biţi (8 biţi/canal), dar TIFF permite stocare pana la 64 biţi.

Structura fişierului TIFF:

- antetul ocupă 8 octeţi; primii doi octeţi au valori ASCII şi indică modul în care trebuie interpretate datele: “II” - datele sunt scrise după convenţia little-endian byte order, “MM” datele sunt scrise după convenţia big-endian byte order.

- IDF (Image File Directory); prima intrare în IDF are o valoare de 2 octeţi şi indică numărul total de intrări. Restul intrărilor în IDF au o lungime de 12 octeţi şi conţine informaţii despre un câmp etichetat (prin etichete se face referirea la un dublet valoare-proprietate).

- datele sunt stocate pe rânduri de dimensiuni variabile.

# 4 Modelul arhitectural și modelul componentelor

## 4.1 Arhitectura sistemului

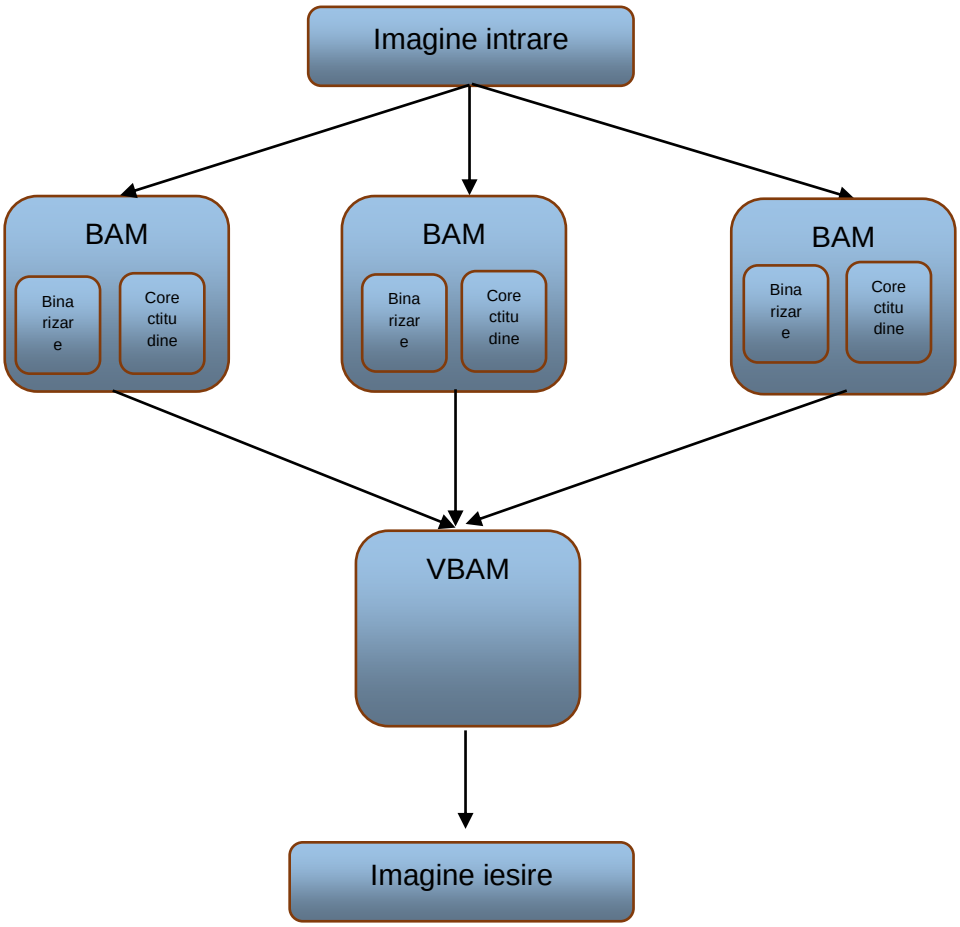
### 4.1.1 Șabloane arhitecturale folosite

Soluția software actuală a fost proiectată după modelul client. Aplicația conține două tipuri de module distincte în componența sa: VBAM și BAM. Există un modul VBAM și 3 module BAM. Funcția de bază este următoarea: BAM-urile preiau ca input o imagine, aplică un algoritm de binarizare și întorc ca output 2 imagini. O imagine reprezintă soluția oferită de algoritmul de binarizare pentru

imaginea input, iar a doua imagine reprezintă factorul de încredere pentru corectitudinea imaginii output. Aceste imagini rezultat sunt preluate ulterior de către VBAM, care aplică un algoritm de votare, pentru a stabili care dintre imagini este varianta optimă pentru binarizare.

### 4.1.2 Diagrama de arhitectură

Diagrama de arhitectură de mai jos descrie componentele arhitecturii aplicației și relațiile de interacțiune dintre acestea;



## 4.2 Descrierea componentelor

Aplicația constă din urmatoarele module interconectate:

- **BAM**: aplică un algoritm de binarizare asupra imaginii primite ca input și realizează imaginea binarizată și imaginea conținând factorul de corectitudine pentru fiecare pixel.
- **VBAM**: preia output-ul modulelor BAM (imaginile binarizate și imaginile factor de corectitudine) și le supune unui algoritm de votare în urma căruia va desemna o singură imagine, presupusă corectă (sau cea mai apropiată de corectitudine**)**

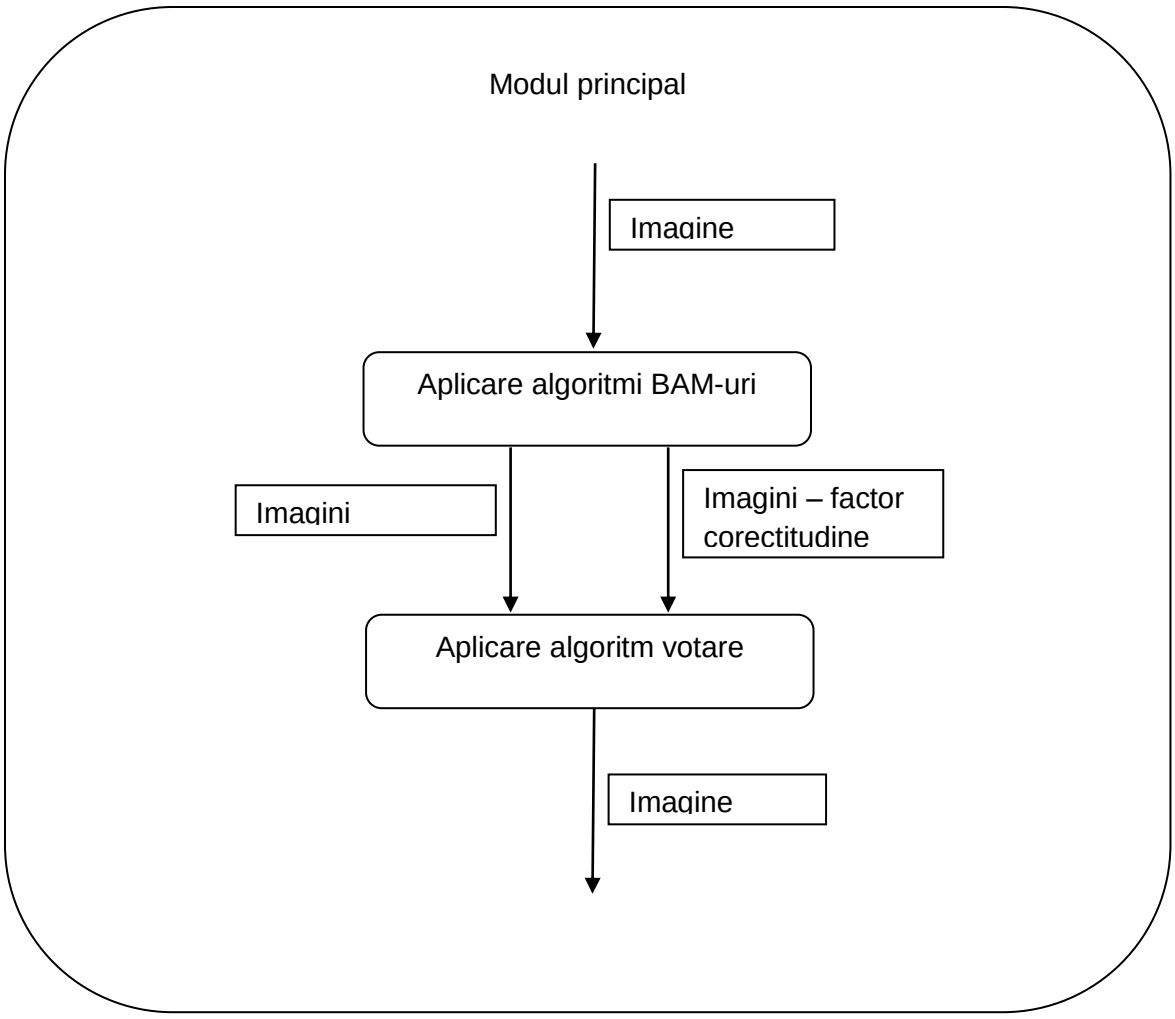
## 4.3 Restricțiile de implementare

Modulele aplicației trebuie să acopere următoarele restricții de implementare:

- Modulele BAM și VBAM vor fi dezvoltate utilizând limbajul de programare C/C++
- Formatul imaginilor de intrare va fi .PGM

## 4.4 Interacțiunea dintre componente

Atunci când utilizatorul pornește programul, se va lansa în execuție proiectul principal, care va rula toate BAM-urile pe imaginea primită ca input, va returna imaginile de output, care la rândul lor vor fi preluate ca input de VBAM. VBAM-ul va aplica un algoritm de votare asupra imaginilor și va desemna care imagine este aleasă drept imagine corectă (sau cea mai apropiată de realitate).




# 5 Modelul interfeței cu utilizatorul

## 5.1 Succesiunea interfețelor

Aplicația de BAM nu este prevăzută cu o interfață grafică controlul efectuându-se din consolă. Pentru pornirea aplicației este necesar să intrați din consolă în directorul cu executabilul. La rularea executabilului trebuiesc pasate două argumente. Primul argument este numele fişierului pentru imaginea ce se doreste a fi binarizată (formatul imaginii de intrare îl găsiti la pag. xy) şi cel de-al doilea argument este numele fişierului în care va fi salvată imaginea rezultat. In plus aplicația va crea un fişier "confidence" în care găsesti valori de sigurantă pe baza căror s-a ales culoarea pentru fiecare pixel din imaginea rezultat.

Pentru rularea aplicației de VBAM, similar aplicației de BAM este nevoie să intrăm din consolă în directorul cu executabilul. Executabilul trebuie să primească oricâte argumente de tipul “string” reprezentând numele unor fişiere rezultate în urma prelucrării modulelor BAM altfel acesta va returna o eroare.

# 6 Elemente de testare

## 6.1 Componente critice

Componentele ce influențează decisiv performanța globală a aplicației sunt BAM-urile care se ocupă de binarizarea imaginii din necesitatea de a traversa imaginea repetat pentru a obține o binarizare cat mai reuşită.

Pentru a evita pierderile majore de performanță aplicația poate fi implementată în asa fel încât BAM-urile să ruleze în paralel si VBAM-ul să pornească procesarea atunci cand a primit rezultatele de la oricare 2 din BAM-uri.

Pentru testarea de stabilitate BAM-ul va fi testat cu imagini corupte, cu un alt format decât cel suportat de aplicație, iar VBAM va primi un număr greşit de BAM-uri.

Fiecare BAM si VBAM va fi testat individual pentru a se asigura funcţionarea corectă a fiecărei componente în parte. Apoi se vor testa împreună pentru a se verifica funcţionarea completă a programului.

## 6.2 Tipuri de testare

**White-box testing**: constă în testarea şi verificarea codului propiu-zis; în timpul implementării fiecărei clase developer-ul care se ocupă de clasa respectivă va face teste pentru a se asigura că fiecare metodă returnează rezultatul corect.

**Black-box testing**: reprezintă majoritatea procesului de testare; aceasta se efectuează după ce toate componentele proiectului sunt realizate şi constă în introducerea input-ului şi verificarea output-ului, daca este cel corespunzator.

**Feature testing**: următoarele paragrafe oferă detalii despre procesul de testare pentru fiecare componentă în parte:

- **Testare BAM**: fiecare BAM va fi testat pe rând pentru următoarele situații: fişier de input corupt, fişier de input cu un format diferit față de cel suportat de aplicație.
- **Testare VBAM**: modulul de VBAM va fi testat cu o serie de imagini.

**Testare non-functională**: se verifică performanța programului, din punct de vedere al timpului de rulare.