MEDUSPILE
Programmatūras prasību specifikācija

Datums: 18.06.2015
Projekts: MEDUS (2013/0031/2DP/2.1.1.1.0/13/APIA/VIAA/010)
Dokumenta tips: Programmatūras prasību specifikācija
Statuss: Galīgā versija
Autors: Ints Mednieks (IME) / EDI
Pārbaudīja:
Pieņēma:
Fails: MEDUS paraugprojekta specifikācija 1_01.doc
Versiju vēsture:

Versija	Autors	Datums	Apraksts
1.01	IME	26.06.2015	Labojumi
1.0	IME	18.06.2015	Galīgā versija
0.9	IME	16.06.2015	Versija ar korekcijām saskaņošanai
0.8	IME	26.05.2015	Sagatavots 3 dažādo .NET moduļu funkciju izsaukšanas piemērs 2.pielikumā; korekcijas 1.pielikumā
0.7	IME	11.05.2015	Pievienots importa funkcijas ‘importPIX’ izsaukšanas piemērs
0.6	IME	07.05.2015	Korekcijas arhitektūras aprakstā
0.5	IME	06.05.2015	Izlabots datu modelis. Pabeigti funkciju apraksti
0.4	IME	28.04.2015	Pievienots pielikums ar informāciju par no MATLAB kompilēto .NET moduļu integrāciju
0.3	IME	8.04.2015	Pievienoti funkciju apraksti
0.2	IME	2.04.2015	Pievienots datu modelis un apraksts
0.1	IME	30.03.2015	Pirmais melnraksts.

Rīga, 2015

Saturs

1. Ievads

1.1. MEDUS projekts un MEDUSPILE paraugprojekts

Projekta MEDUS mērķis ir izveidot .NET programmu izstrādes tehnoloģiju, kas ļauj ievērojami paaugstināt lietojumprogrammatūras izstrādes pielāgojamību un produktivitāti.

Projekta ievaros ir radīta specializēta projekta modelēšanas vide un programmatūras izstrādes rīki .NET platformai. Izmantojot šos rīkus, var izstrādāt pielāgotas programmas dažādiem pielietojumiem.

Projekta ietvaros paredzēts izstrādāto tehnoloģiju pārbaudīt, realizējot paraugprojektu tālizpētes datu (TD) apstrādes jomā. Paraugprojekta ietvaros tiks izstrādāta lietojumprogramma MEDUSPILE, kuras prasību specifikācija definēta šajā dokumentā.

Lietojumprogramma MEDUSPILE būs paredzēta lietošanai klienta datorā Windows vidē. Datu apstrādes funkcijas tiks realizētas MATLAB vidē un pieslēgtas .NET moduļu ("assembly") veidā. Lietotāji varēs izmantot lokālu vai centralizētu datubāzi.

MEDUSPILE būs lietojumprogramma, kas domāta TD importam un vizualizācijai, priekšapstrādei un klasifikācijai, izmantojot dažādu sensoru datu sapludināšanas pieejas, kuras izstrādātas MEDUS projekta ietvaros. Programma izmantojama pētnieciskiem mērķiem, bet var kalpot arī kā kodols specializētu TD apstrādes programmu izstrādei dažādās pielietojumu jomās (mežsaimniecībā, lauksaimniecībā, vides aizsardzībā, pilsētvides monitoringam u.c.), kurās mērķtiecīgi izmantot tālizpētes iespējas.

1.2. Dokumenta mērķis

Šis dokuments domāts projekta MEDUS ietvaros izstrādājamās programmas MEDUSPILE prasību definēšanai. Mērķa auditorija ir projekta dalībnieki SWH SETS un EDI.

1.3. Definīcijas, Saīsinājumi

DAM	Datu Apstrādes Modulis (.NET assembly)
DLL	Dynamic Link Library (dinamiski piesaistāma bibliotēka)
HTML	HyperText Markup Language - hiperteksta iezīmēšanas valoda
.NET	Firmas Microsoft programmatūras ietvars
IIS	Internet Information Services (Windows konteiners, kurā darbojas tīkla lietojumprogrammas)
MS	Multispektrālais (attēls)
HS	Hiperspektrālais (attēls)
LIDAR	Light Detection And Ranging –detektēšana un attāluma mērīšana ar gaismu
SAR	Synthetic Aperture Radar – sintētiskās apertūras radars
SDK	Software Development Kit
SHP	Shapefile (formas faila) paplašinājums

1.4. Literatūra

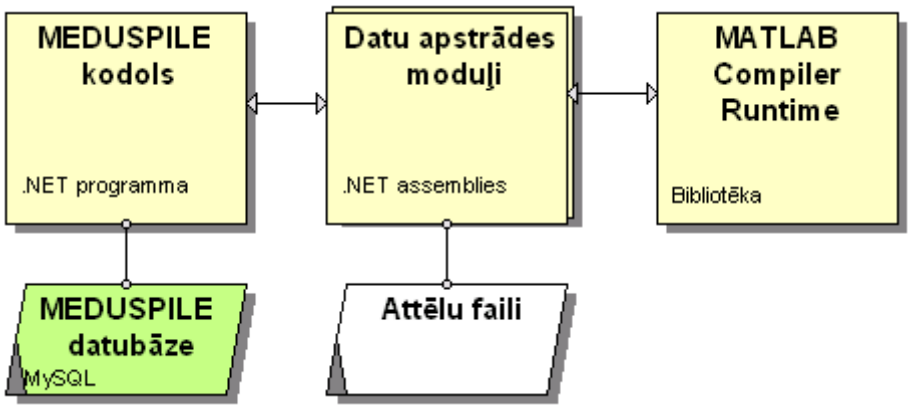
[1] MATLAB – The language of Technical Computing. Tiešsaistē: 'http://se.mathworks.com/products/matlab/ '
[2] Projekts MEDUS. Starppārskats "Tālizpētes datu veidu un formātu izpēte". EDI, 2014.
[3] R. Dinuls, A. Lorencs, I. Mednieks, “Using Consolidated Covariance Image for Discrimination of Habitats”, in Proceedings of the 13th Biennial Baltic Electronics Conference, Tallinn, Estonia, pp.299-302., 2012.
[4] R. D. Jackson, A. R. Huete, “Interpreting vegetation indices”, in Preventive Veterinary Medicine, Vol. 11, Issues 3–4, pp. 185–200, 1991.
[5] MATLAB 2015a Compiler SDK. Help faili: '../help\compiler_sdk\index.html'.
[6] MATLAB 2015a DotNetBuilder Help fails: MWArrayAPI.chm

2. Vispārējs apraksts

MEDUSPILE jārealizē lietojumprogrammas veidā, kura balstās uz .NET v.4.0 vai vēlākas versijas ietvaru, un kuru jāinstalē klienta datorā Windows 7 vai vēlākas versijas vidē. Lai nodrošinātu pēc iespējas lielāku ātrdarbību, apstrādājamie TD jāizvieto failu sistēmā, ieteicams uz SSD tipa diskiem, kā arī jāizmanto lokālais tīkls ar ātrumu 1Gb/s, ja tiek izmantota centralizēta datu glabāšana. Lai minimizētu izmaksas, jāizmanto brīvpieejas datu bāzu vadības sistēma, piemēram MySQL. Datu apstrādes algoritmi jārealizē MATLAB v.2015a [1] vidē un jāpiesaista programmai .NET moduļu ("assembly") veidā, kas sagatavoti, izmantojot MATLAB Compiler SDK komponenti, kura ir EDI rīcībā. Programma jāveido tā, lai to būtu iespējams vienkārši un ātri paplašināt, pievienojot jaunas TD apstrādes funkcijas.

2.1. Arhitektūra

MEDUSPILE komponentes un apkārtējā izpildes vide ilustrēti 1.att.



Att. 1. Programmas MEDUSPILE komponentes un izpildes vide.

MEDUSPILE jāinstalē darbam Windows vidē, tā izmantos savu datubāzi, kuru var izvietot uz lokālā datora vai servera. Programmas kodolam, kas veidots, izmantojot projekta ietvaros izstrādāto tehnoloģiju, jānodrošina lietotāja saskarni un operācijas ar datubāzi. Specializētai TD apstrādei tiks izmantoti MATLAB vidē programmēti un kompilēti datu apstrādes moduļi (DAM), kas tiks

integrēti MEDUSPILE programmā kā .NET moduļi ("assemblies"). Šie moduļi, kurus sagatavos EDI projekta dalībnieki, tiks izsaukti no programmas kodola un nodrošinās gan TD apstrādi, gan arī rezultātu vizualizāciju. MEDUSPILE datubāzē tiks glabāti tikai metadati, liela apjoma multispektrālie attēli un citi TD tiks glabāti failu veidā, lai nodrošinātu iespējas ar tiem manipulēt maksimāli efektīvi. Tā kā datu apstrādes moduļi būs realizēti MATLAB programmēšanas vidē, to izpildei programmai tiks pieslēgta MATLAB kompilatora izpildes laika bibliotēka.

2.2. Funkcijas

MEDUSPILE programmai jāpilda šādas funkcijas:

- Tālizpētes datu imports. Tālizpētes dati tiek iegūti ar dažādu sensoru (multispektrālo, hiperspektrālo, LiDAR, SAR) palīdzību un tiek piegādāti dažādos formātos. Šādu datu veidi un formāti aprakstīti [2]. Programmai jāsatur funkcijas šādu datu importam datubāzē un failu sistēmā, lai tiem būtu iespējams ērti piekļūt no MEDUSPILE programmas.
- TD bibliotēkas caurskatīšana. Lietotājam jāpiedāvā iespējas atlasīt sarakstā datus, izvēloties datu veidu un projektu, kuram tie piesaistīti.
- Datu apstrāde (radiometriskā korekcija, attēlu veidošana no LiDAR datiem, klasifikācija…). Apstrādes funkciju izpildei lietotājam jāpiedāvā izvēlēties datu avotu, apstrādes veidu un funkcijas jāizpilda, izsaucot attiecīgu datu apstrādes moduli. Pēc funkcijas izpildes jānodrošina rezultātu vizualizācija un saglabāšana MEDUSPILE datubāzē un failos.
- Multispektrālo un hiperspektrālo datu vizualizācija. Tā kā spektrālie dati satur informāciju no daudzām spektra joslām, nepieciešams izvēlēties, kuru joslu datus attēlot un ar kādu relatīvo gaišumu. Vizualizāciju nodrošina ar MATLAB programmēta moduļa palīdzību.
- Veģetācijas indeksu attēlu veidošana, sliekšņu pielietošana. Veģetācijas indeksu attēlu pikseļu vērtības aprēķina ar MATLAB programmēta moduļa palīdzību, balstoties uz aritmētiskām operācijām ar attiecīgām multispektrālā attēla pikseļu vērtībām dažādās multispektrālā attēla joslās [4].
- Zemes izmantošanas kategoriju definēšana. Lai realizētu TD klasifikācijas uzdevumus, no kuriem izplatītākais ir zemes izmantošanas klasifikācija, MEDUSPILE programmai ar MATLAB programmēta moduļa palīdzību jānodrošina klasifikācijas kategoriju definēšanas un apmācības apgabalu (lauka datu) atzīmēšanas iespējas.
- Lauka datu imports un vizualizācija. Gadījumiem, kuros lauka datu pieejami ārēji sagatavotu failu veidā, jānodrošina šo datu imports un attēlošana uz spektrālā vai cita attēla fona. MEDUSPILE programmā jānodrošina šādas funkcijas realizējošo moduļu izsaukšana.

2.3. Lietotāji

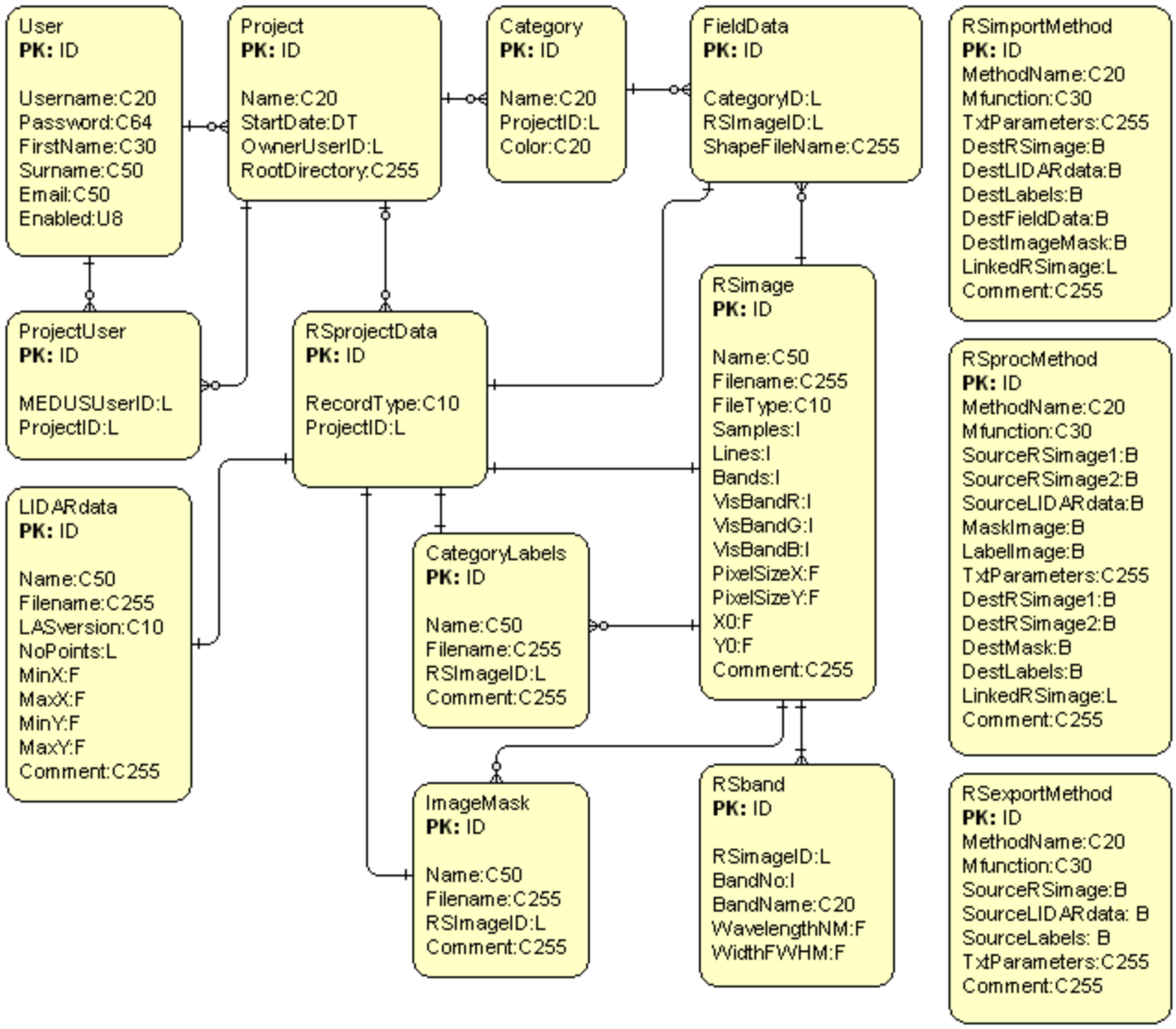
Programmas lietotāji definēto funkciju apjomā būs pētnieki, kuri nodarbojas ar metožu izstrādi TD apstrādei.

Programma jābūvē tā, lai to varētu ērti paplašināt, tādejādi tā varētu kalpot par sagatavi programmu izstrādei TD izmantošanai dažādās pielietojumu jomās.

3. Datu objekti

Datubāzē glabājamo objektu datu modelis parādīts 2.attēlā un galvenās entītijas aprakstītas 1.tabulā. Pamatprincipi, uz kuriem tas balstīts, ir šādi:

- darbs ar programmu notiks projektu veidā. Katram projektam ir īpašnieks un lietotāji, kurus piesaista īpašnieks;
- datubāzē glabājas tikai TD un cita veida datu apraksti, kuros ir norādes uz pašu datu vietu failu sistēmā. Katram projektam ir sava vieta failu sistēmā (saknes mape);
- projektam var piesaistīt dažāda veida TD, palīgdatus (masku un kategoriju iezīmju attēlus). Projektam var piesaistīt datubāzē jau esošus datus (kopējot), kuri importēti citu projektu ietvaros vai arī importēt projektam specifiskus datus. Ja projekts saistīts ar attēlu klasifikāciju, tam jādefinē kategorijas un, iespējams, jāimportē vai interaktīvi jādefinē lauka dati (apriorā informācija par attēlu pikseļiem, kas atbilst kādai no kategorijām);
- datubāzē tiek sagatavota informācija par izmantojamajām importa, datu apstrādes un eksporta metodēm, kuras sagatavotas MATLAB vidē un programmai padarītas pieejamas .NET moduļu veidā. Strādājot ar projektu, tiek izsauktas pieejamās metodes un kā parametri tām nodoti projektā pieejamie datu objekti. Iegūtie rezultāti tiek saglabāti failu veidā, datubāzē tiek pierakstīta informācija par tiem un tie tiek arī piesaistīti projektam;
- visas darbības projektu ietvaros, kuru rezultātā mainījušies dati datubāzē, kā arī .NET moduļu izsaukumi, tiek piefiksētas projektu žurnālā.



Att. 2. MEDUSPILEs datu modelis. Lauku datu tipi:

- Cn- teksta lauks ar maksimālo garumu n;
- I – integer
- U8 – unsigned 8 biti
- L – long
- B – boolean
- D – double (64 biti)
- F - float (32 biti)
- DT - datetime

Visi ID lauki ir L tipa.

1.tabula. Datubāzes entītijas.

Entītija	**Apraksts**
Project	Datu, metožu un rezultātu apkopojums, kas raksturo vienu ar MEDUSPILE palīdzību risināmo uzdevumu. Projektam ir viens īpašnieks- MEDUSPILE lietotājs, kas organizē darbus. Projekta aktivitātes tiek piefiksētas žurnāla ierakstos šim projektam. Projektam tiek izveidota datu mape failu sistēmā, kurā tiek glabāti ar projektu saistītie TD faili, lauka dati, starprezultātu faili (tai skaitā masku un iezīmju faili) un galējie rezultāti.
ProjectUser	Ieraksti, kas sasaista projektus ar lietotājiem. Katram lietotājam, kurš nav administrators, būs redzami tikai projekti, kuru īpašnieki viņu ir iesaistījuši.
User	Programmas lietotājs. Parole tiek glabāta šifrētā veidā. Administrators lietotāju var deaktivēt.
RSimage	Informācija par TD attēlu, kas var saturēt MS, HS un SAR datus, uz ko norāda lauks FileType ('MS' | 'HS' | 'SAR'). Paši dati binārā formā glabājas failu sistēmas failā ar nosaukumu Filename. TD attēla faili tiek izveidoti ar MATLAB importa metožu palīdzību, kuru rezultātā dati tiek pārveidoti iekšējā formātā (3D vai 2D matrica). MS un HS attēli tiek glabāti kā 3D matricas ar izmēru Lines x Samples x Bands. SAR attēls ir 2D attēls (Bands=1).
RSband	TD attēla spektrālā josla. SAR attēla gadījumā, kā arī MS un HS attēlu palīgjoslām tai nav definēts viļņa garums (WavelengthNM) un joslas platums (WidthFWHM). Joslai ir: numurs, tas vienāds ar attēla matricas trešās dimensijas indeksu; vārds, ko attēlo GUI; viļņa garums nm.
LIDARdata	Informācija par LIDAR datiem. Paši dati iekšējā formātā glabājas failā ar nosaukumu Filename un tiek izveidoti ar importa metožu palīdzību.
RSprojectData	Ieraksti, kas saista TD datus (RSimage | LIDARdata) vai citus attēlus (FieldData | ImageMask | CategoryLabels) ar projektu. RecordType norāda datu tipu: 'RSimage' | 'LIDAR' | 'Field' | 'Mask' | 'Labels'.
Category	TD klasifikācijas (piem. zemes izmantošanas) kategorija. Projektos, kas saistīti ar TD klasifikāciju, jādefinē kategorijas, uz kurām klasificēt datus.
FieldData	Katrai klasificējamajai kategorijai var būt definēti apmācības datu rajoni, t.i. rajoni, kuri ir zināmi kā šai kategorijai piederoši. Katrs FieldData ieraksts satur informāciju par šiem rajoniem, kas definēta SHP faila veidā, kurš saistīts ar kādu no TD attēliem.
ImageMask	Binārs 2D attēls pikseļu kopas izdalīšanai no pamatattēla, uz kuru norāda RSImageID. Attēla izmēri vienādi ar pamatattēla Lines x Samples.
CategoryLabels	Kategoriju numuru attēls 2D, saistīts ar pamatattēlu, uz kuru norāda RSImageID. Attēla izmēri vienādi ar pamatattēla Lines x Samples. Pikseļu vērtības *k*, kuras ir >0, norāda uz pikseļa piederību kategorijai ar numuru *k*. Šāds attēls var kalpot gan kā lauka dati, gan klasifikācijas rezultāta

	fiksēšanai.
RSimportMethod	Informācija par datu importa metodi. Lai metodi izmantotu, jābūt sagatavotam .NET modulim, no kura šo metodi var izsaukt. Ar DestRSimage, DestLIDARdata un DestLabels lauku palīdzību tiek norādīts importa mērķa tips (tieši vienam no šiem laukiem jāpiešķir vērtība "true", pārējiem- "false"). Definētie parametri TxtParameters obligāti jāievada, lai metodi izsauktu. Importa avots jāizvēlas pirms metodes izsaukšanas un jānodod kā viens no teksta parametru virknes *TxtParametri* locekļiem. *TxtParametri* ir formātā "param1=<value>; param2=<value>;…", tos izsauktā metode pati izanalizē. Ja metodes izsaukums ir veiksmīgs, no tās atgrieztās struktūras jāizveido attiecīgā tipa (RSimage, LIDARdata, CategoryLabels, FieldData vai ImageMask) mainīgo, kurš jāieraksta datubāzē ar izvēlēto vārdu un par kuru jāizveido arī *RSprojectData* ieraksts. Paši importētie dati paliks failu sistēmā, uz tiem norādīs attiecīgais lauks Filename, kurš norādīts izejas parametrā. Ja tiek importēti CategoryLabels, FieldData vai ImageMask tipa dati, tie jāsasaista ar izvēlēto RSimage objektu.
RSprocMethod	Informācija par datu apstrādes metodi. Lai metodi izmantotu, jābūt sagatavotam .NET modulim, no kura šo metodi var izsaukt. Ar SourceRSdata1, SourceRSdata2, SourceLIDARdata, MaskImage, LabelImage lauku palīdzību tiek norādīts, kuri no ieejas un izejas parametriem tiek izmantoti metodes izsaukumā. Vismaz vienam no izejas parametriem DestRSimage1, DestRSimage2, DestMask vai DestLabels jāpiešķir vērtība "true". Definētie parametri TxtParameters obligāti jāievada, lai metodi izsauktu. TxtParametri ir formātā "param1=<value>;param2=<value>;…", tos izsauktā metode pati izanalizē. Ja metodes izsaukums ir veiksmīgs, no tās atgrieztajām struktūrām jāizveido mainīgo(s), kurš(-i) jāieraksta datubāzē ar izvēlēto(-ajiem) vārdu(-iem) un par kuru(-iem) jāizveido arī *RSprojectData* ieraksts(-i). Izejas parametriem atbilstošie dati paliks failu sistēmā, uz tiem norādīs attiecīgais lauks Filename, kurš norādīts izejas parametrā. Ja metodes izpildes rezultātā tiek izveidoti CategoryLabels, FieldData vai ImageMask tipa dati, tie jāsasaista ar izvēlēto RSimage objektu.
RSexportMethod	Informācija par datu eksporta metodi. Lai metodi izmantotu, jābūt sagatavotam .NET modulim, no kura šo metodi var izsaukt. Ar SourceRSimage, SourceLIDARdata un SourceLabels lauku palīdzību tiek norādīts eksporta avota tips (tieši vienam no šiem laukiem jāpiešķir vērtība "true", pārējiem- "false"). Definētie parametri TxtParameters obligāti jāievada, lai metodi izsauktu. TxtParametri ir formātā "param1=<value>;param2=<value>;…", tos izsauktā metode pati izanalizē. Eksporta mērķis (faila vārds) jāizvēlas pirms metodes izsaukšanas un jānodod kā viens no teksta parametru virknes *TxtParametri* locekļiem.

4. Vispārējās prasības

Realizācijā jābalstās uz sekojošiem principiem:

- Jānodrošina iespēja ērti paplašināt programmas funkcijas, definēt dažādus objektu aprēķina algoritmus un citus datu veidus, piesaistīt papildus datu apstrādes .NET moduļus (ar programmas pārkompilēšanu)
- Jānodrošina vairāku lietotāju paralēls darbs ar centralizētu datubāzi uz servera vai individuāls darbs uz viena datora ar lokālu datubāzi
- Jānodrošina iespēju strādāt ar vairākiem funkciju ekrāniem (piem. ciļņu veidā)
- Viegla interfeisa valodas maiņa darba sesijā (interfeisa teksti konfigurācijas failos)
- Jāparedz ar kontekstu saistīti paskaidrojumi
- Jāizmanto .NET platforma

5. Funkcionālās prasības

5.1. Lietotāju vadības funkcijas

Kopīgi izmantotas datubāzes lietotāju vadības funkcijas var izpildīt tikai lietotājs ar Administratora lomu. Lietotājs ar Administratora lomu izveido citu lietotāju kontus.

Lietotājs ar Administratora lomu var cita lietotāja kontu deaktivēt, kā arī atiestatīt paroli uz sākotnējo.

Par katru lietotāju jābūt aizpildītiem šādiem laukiem:

- Lietotāja vārds
- Vārds
- Uzvārds
- E-pasta adrese

5.2. Lietotāju lomas

Ir šādas lomas:

- Administrators: tikai lietotājam ar šādu lomu ir tiesības pievienot/rediģēt/deaktivēt/dzēst citus lietotājus User tabulā. Administrators var izveidot jaunu projektu, kļūt par tā īpašnieku un piesaistīt šādam projektam citus lietotājus. Administrators var arī atiestatīt lietotāju paroles, kā arī pārlūkot audita žurnālu. Ja programmu individuāli lieto viens lietotājs uz lokāla datora, tam jāpiešķir administratora tiesības. Ja programmu lieto vairāki lietotāji uz viena datora, vismaz vienam no tiem būs jābūt administratora tiesībām;
- Parasts lietotājs: tādam ir visas citas tiesības, izņemot tās, kuras ekskluzīvi piešķirtas tikai administratoriem. Parasts lietotājs var mainīt savu paroli.

5.3. Datu objektu caurskatīšana, pievienošana, rediģēšana un dzēšana

Jāparedz iespēja caurskatīt visus sistēmā paredzētos datu objektus saraksta veidā ar filtrēšanas un sakārtošanas iespējām. Sarakstā izvēlēto ierakstam jāparedz detaļu skats, kurā tiek attēloti visi datu objekta lauki. Daļai no datu objektiem (skat. sīkāku aprakstu zemāk) jāparedz iespēja pievienot jaunus ierakstus un rediģēt datu laukus.

Iespējams pievienot / rediģēt /dzēst šādus datu objektus:

- User (nepieciešamas administratora tiesības)
- ProjectUser (to var darīt administrators - projekta īpašnieks – OwnerUser)
- Project (jaunu projektu var izveidot administrators)
- Category
- RSimportMethod (nepieciešamas administratora tiesības)
- RSprocMethod (nepieciešamas administratora tiesības)
- RSexportMethod (nepieciešamas administratora tiesības)

Iespējams izveidot tikai importējot vai apstrādes funkcijas izpildes rezultātā (kā arī dzēst):

- RSimage (var rediģēt Comment lauku)
- RSband (tiek importēti kopā ar saistīto RSimage)
- LIDARdata (var rediģēt Comment lauku)
- CategoryLabels (var rediģēt Comment lauku)
- ImageMask (var rediģēt Comment lauku)
- FieldData
- RSprojectData (var arī piesaistīt datus, kopējot no cita projekta)

5.4. Datu importa funkciju definēšana

Jānodrošina iespējas definēt datu importa funkcijas, ko var izmantot projektos. Tās jāsaskaņo ar izstrādātajiem attiecīgajiem DAM.

Lai to veiktu, lietotājam jāparāda sistēmā jau definēto importa funkciju saraksts, sakārtots vārdu (MethodName) alfabētiskā secībā (no RSimportMethod tabulas) ar filtrēšanas iespējām (skatīt visas funkcijas | skatīt funkcijas, kas domātas RSimage objektu importam- tām DestRSimage=true | skatīt funkcijas, kas domātas LIDARdata objektu importam- tām DestLIDARdata=true | skatīt funkcijas, kas domātas CategoryLabels objektu importam- tām DestLabels=true). Šim sarakstam jāparedz iespēja pievienot jaunu ierakstu- ja tā tiek izsaukta, jāatver dialoga logs, kurā var ievadīt RSimportMethod lauku vērtības. Importa funkcijai jāizvēlas viens no laukiem DestRSimage, DestLIDARdata vai DestLabels kā importa mērķis, kā arī tekstveida parametrus formātā "param1=<value>; param2=<value>;…"..

5.5. Datu apstrādes funkciju definēšana

Jānodrošina iespējas definēt datu apstrādes funkcijas, ko var izmantot projektos. Tās jāsaskaņo ar izstrādātajiem attiecīgajiem DAM.

Lai to veiktu, lietotājam jāparāda sistēmā jau definēto apstrādes funkciju saraksts (no RSprocMethod tabulas), sakārtots vārdu (MethodName) alfabētiskā secībā. Šim sarakstam jāparedz iespēja pievienot jaunu ierakstu- ja tā tiek izsaukta, jāatver dialoga logs, kurā var ievadīt RSprocMethod lauku vērtības. Apstrādes funkcijai var atzīmēt vienu vai divus apstrādājamos TD attēlus (SourceRSimage1, SourceRSimage2) un/vai apstrādājamo SourceLIDARdata objektu, kā arī maskas un kategoriju iezīmju attēlus un tekstveida parametrus formātā "param1=<value>; param2=<value>;…". Apstrādes funkcijas rezultāts var būt viens vai divi TD attēli (DestRSimage1, DestRSimage2), maskas attēls (DestMask) vai kategoriju iezīmju attēls (DestLabels).

5.6. Datu eksporta funkciju definēšana

Jānodrošina iespējas definēt datu eksporta funkcijas, ko var izmantot projektos. Tās jāsaskaņo ar izstrādātajiem attiecīgajiem DAM.

Lai to veiktu, lietotājam jāparāda sistēmā jau definēto eksporta funkciju saraksts (no RSexportMethod tabulas), sakārtots vārdu (MethodName) alfabētiskā secībā. Šim sarakstam jāparedz iespēja pievienot jaunu ierakstu- ja tā tiek izsaukta, jāatver dialoga logs, kurā var ievadīt RSexportMethod lauku vērtības. Funkcijai jānorāda eksportējamais TD objekts (SourceRSimage, SourceLIDAR data vai SourceLabels) un izmantojamie tekstveida parametri formātā "param1=<value>; param2=<value>;…". Apstrādes funkcijas rezultāts ir failu sistēmā eksportēts fails, to lietotājs izvēlēsies eksporta DAM izpildes laikā.

5.7. Projektu vadība (izveidošana, papildināšana, dzēšana)

Projektus var izveidot un dzēst tikai lietotājs ar administratora tiesībām. Administrators, kurš izveidojis projektu, kļūst par tā īpašnieku un var pievienot/atvienot projektam citus lietotājus. Parasts lietotājs var atvērt un strādāt ar esošu projektu, kuram viņš ir piesaistīts.

Projektu ietvaros notiek šādas darbības:

- TD vai CategoryLabels imports
- TD vai palīgattēlu (ImageMask, CategoryLabels) piesaistīšana no cita projekta vai pie projekta nepiesaistīta datu objekta
- Kategoriju definēšana
- Lauka datu (FieldData) imports vai interaktīva definēšana kādai no kategorijām
- Apstrādes metožu izsaukšana, rezultāti tiek piesaistīti atvērtajam projektam. Apstrādes funkcijas, ko izsauc projektā, var darboties tikai ar projektam piesaistītiem datu objektiem
- Eksporta metožu izsaukšana

5.8. Datu imports uz projektu

Izpildot šo funkciju, lietotājam jāpiedāvā izvēlēties izsaucamo importa funkciju. Pēc šīs izvēles dialogā jāattēlo funkcijas izsaukšanas parametru ievades lauki; saskaņā ar funkcijas definīciju RSimportMethod ierakstā (skat. 5.4) lietotājam jāpiedāvā ievadīt nepieciešamos parametrus, kuri ir obligāti jāievada, lai funkciju izsauktu. Pēc parametru ievades jāatļauj funkcijas izsaukšana, kuras izpildei tiek izsaukts attiecīgs DAM ar ievadītajiem parametriem. Ja tā ir veiksmīga, tā atgriež attiecīgā izejas parametra mainīgo, kurš jāieraksta datubāzē ar izvēlēto vārdu un par kuru jāizveido arī RSprojectData ieraksts, tā sasaistot importa rezultāta objektu ar pašlaik atvērto projektu. Importētajiem datiem jāpaliek failu sistēmā, uz tiem norādīs attiecīgais lauks Filename, kurš norādīts izejas parametrā.

5.9. Datu kopēšana no cita projekta

Ja datu objekts jau eksistē, jānodrošina iespēja to nokopēt lietošanai atvērtajā projektā, izveidojot attiecīgo RSprojectData objektu. Lai to paveiktu, lietotājam jāparāda datu objektu saraksts ar filtrēšanas/meklēšanas iespējām un jāpiedāvā izvēlēties no šī saraksta piesaistāmo datu objektu. Pēc lietotāja izvēles un piesaistes iniciēšanas ar pogas / saites palīdzību, jāizveido RSprojectData ieraksts, kas sasaista TD objektu ar projektu.

5.10. Kategoriju definēšana

Projektiem, kas saistīti ar attēlu klasifikāciju, jānodrošina iespēja definēt objektu kategorijas un to vizualizēšanas parametrus (piem. krāsu).

5.11. Lauka datu imports un vizualizācija

Ja projektā definētas kategorijas un tas paredz klasifikāciju ar apmācību, kategorijām nepieciešams sagatavot informāciju apmācībai (t.s. „lauka datus"). Viens no izplatītiem veidiem, kuros tiek definēti lauka dati, ir formas faili ar paplašinājumu SHP (shapefile), kuros nodefinētas ģeogrāfisko apgabalu robežas, par kuriem ir zināms, ka tie pieder noteiktai (piem. zemes izmantošanas) kategorijai. Programmā nepieciešams paredzēt importa DAM, kas sasaista pieejamo SHP failu ar projektā definētu kategoriju.

5.12. Kategoriju apgabalu atzīmēšana attēlā

Alternatīvs veids, kā sagatavot apmācības datus zemes izmantošanas klasifikatora apmācībai, ir TD attēlā interaktīvi atzīmēt apgabalu, ko lietotājs atzīst par kādai kategorijai piederīgu. Lai to veiktu, jāizvēlas viena no kategorijām, TD attēls un jāizsauc šīs funkcijas realizācijai sagatavots DAM (izmantojot speciālu RSprocMethod). Rezultātā jāizveido SHP fails un attiecīgie ieraksti tabulās FieldData un RSprojectData.

5.13. Datu apstrāde projekta ietvaros

Lai veiktu datu apstrādi, jāsagatavo tam domātie DAM un attiecīgie RSprocMethod ieraksti datubāzē.

Izpildot šo funkciju, lietotājam jāpiedāvā izvēlēties izsaucamo apstrādes funkciju. Pēc šīs izvēles dialogā jāattēlo funkcijas izsaukšanas parametru ievades lauki; saskaņā ar funkcijas definīciju RSprocMethod ierakstā lietotājam jāpiedāvā ievadīt nepieciešamos ieejas un izejas parametrus, kuri ir obligāti jāievada, lai funkciju izsauktu. Pēc parametru ievades jāatļauj funkcijas izsaukšana, kuras izpildei tiek izsaukts attiecīgs DAM ar ievadītajiem parametriem.

Ja funkcijas izpilde ir veiksmīga, tā atgriež funkcijas definīcijā paredzētos izejas parametra(-u) mainīgo(-s), kurš(-i) jāieraksta datubāzē ar izvēlēto(-ajiem) vārdu(-iem) un par kuru(-iem) jāizveido arī RSprojectData ieraksts(-i), tā sasaistot rezultāta(-u) objektu(-s) ar pašlaik atvērto projektu. Rezultātiem (attēliem vai citiem datu objektiem) jāpaliek failu sistēmā, uz tiem norādīs lauks Filename, kurš norādīts attiecīgajā izejas parametrā.

5.14. Datu eksports

Lai veiktu datu eksportu no MEDUSpile-s, jāsagatavo tam domātie DAM un attiecīgie RSexportMethod ieraksti datubāzē.

Izpildot šo funkciju, lietotājam jāpiedāvā izvēlēties izsaucamo eksporta funkciju. Pēc šīs izvēles dialogā jāattēlo funkcijas izsaukšanas parametru ievades lauki; saskaņā ar funkcijas definīciju RSexportMethod ierakstā (skat. 5.4) lietotājam jāpiedāvā ievadīt nepieciešamos parametrus, kuri ir obligāti jāievada, lai funkciju izsauktu. Pēc parametru ievades jāatļauj funkcijas izsaukšana, kuras izpildei tiek izsaukts attiecīgs DAM ar ievadītajiem parametriem.

5.15. Citas funkcionālās prasības

5.15.1. Drošība

Parole jāglabā šifrētā veidā.

5.15.1. Kļūdu žurnāls

Jāveido speciāls MEDUSPILE programmas kļūdu žurnāla fails; ik pēc nedēļas jāveido jauns.

5.15.2. Audita pieraksti

Projekta dalībnieku aktivitātes, kuras izraisa izmaiņas datubāzē, kā arī DAM izsaukšana, jāfiksē datubāzes tabulā ProjectLog. Lietotājam ar Admistratora lomu jāparedz iespēja pārlūkot šīs tabulas datus, tai skaitā to filtrēt un sakārtot.

5.15.3. Interfeisa valoda

Visi lietotāja saskarnes teksti tiek glabāti sistēmas atbalstītajām valodām atbilstošos failos. Sistēmu instalējot, tiek uzstādīti Piegādātāja izvēlētās lietotāja saskarnes valodas faili. Noklusētā valoda-latviešu.

6. Nefunkcionālās prasības

6.1. Datora resursi

Rekomendētie datora resursi, uz kura izmantos programmu:
- CPU 2.5GHz 4 cores
- RAM 8GB
- Videokarte ar GPU paātrinātāju
- Windows 7 vai jaunāka operētājsistēma
- Pārlūki: Firefox, Chrome

Displeja izšķirtspēja: 1920x1080 pikseļi,

6.2. Komentāri

Programmas izejas teksti jākomentē latviešu valodā tādā formātā, lai komentārus varētu izmantot Doxygen dokumentācijas ģenerators (www.doxygen.org).

6.3. Dokumentācija

Programmas lietošanas instrukcija latviešu valodā jāpiegādā kopā ar programmu no tās atverama palīga formā.

Programmai nepieciešama administratora rokasgrāmata latviešu valodā, kurā aprakstīti tās instalēšanas un uzturēšanas procedūras.

6.4. Programmatūras piegādes pakete

Programmā jāpiegādā izpildāma instalācijas faila veidā.

1.pielikums. Interfeiss ar MATLAB kompilētu .NET moduli

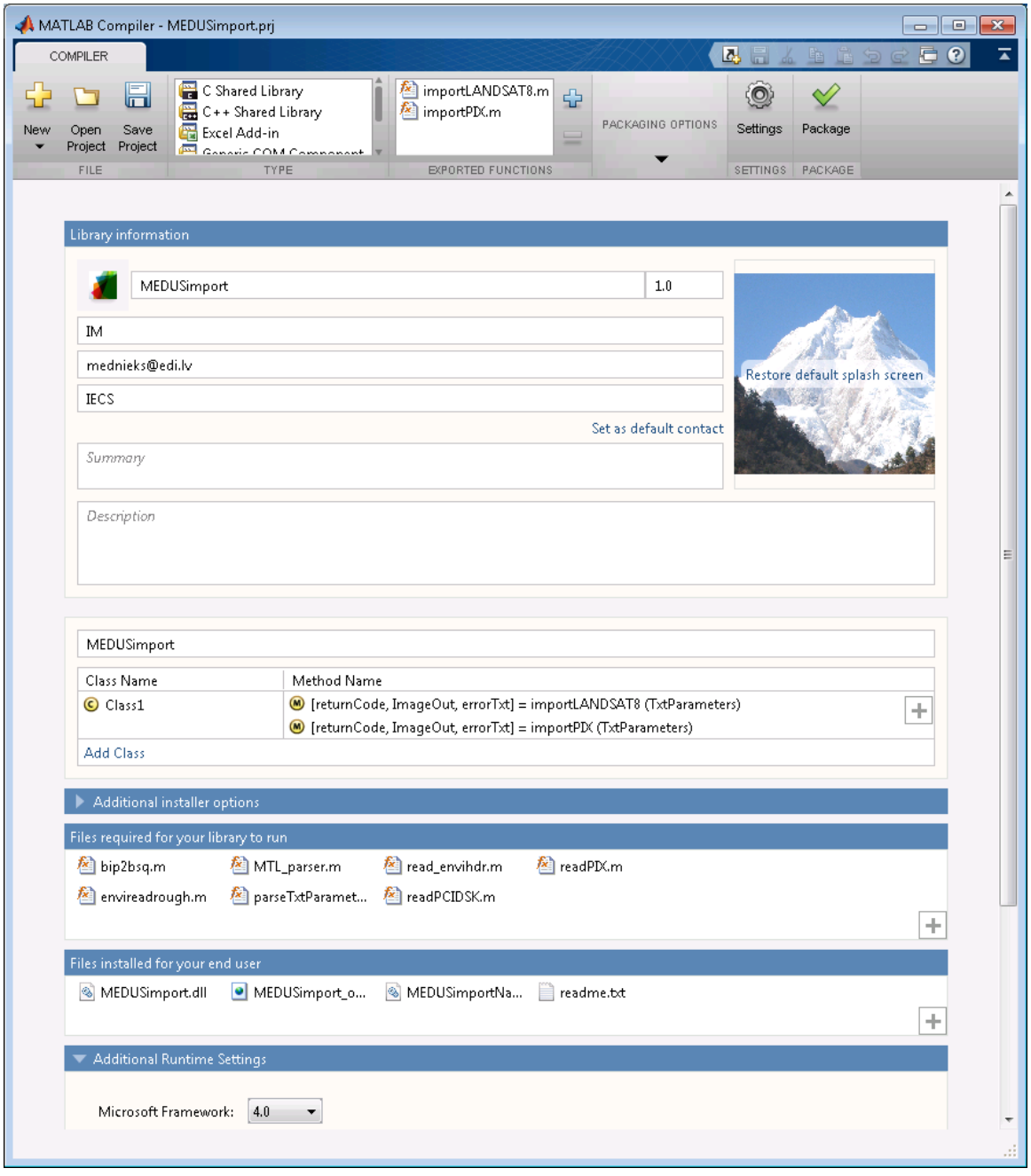
MATLAB 2015a kompilatora SDK ģenerē .NET moduļus, kurus var integrēt .NET lietojumprogrammās, izmantojot zemāk aprakstītos principus. Sīkāks apraksts dots MATLAB 2015a dokumentācijā [5, 6].

- Pirms .NET moduļa kompilācijas jāsagatavo izsaucamo MATLAB funkciju M-faili. .NET moduļi (assemblies) tiek kompilēti no MATLAB funkcijām, izmantojot MATLAB Library Compiler rīku (sk. attēlu zemāk). Šeit jāizvēlas sekojoši parametri:

- rīku TYPE sadaļā jāizvēlas '.NET Assembly';
- rīku EXPORTED FUNCTIONS sadaļā jāpievieno .NET modulī redzamās funkcijas;
- rīku 'Settings' sadaļa jāatstāj bez izmaiņām;
- jāizvēlas, vai MATLAB Runtime (MCR) tiks iekļauta instalācijas paketē vai to instalēšanas laikā lejuplādēs no tīmekļa (ieteicams pirmajā instalācijas paketē to iekļaut, lai nebūtu versiju nesakritības; pārējās paketēs tā vairs nebūs vajadzīga, jo jāinstalē vienreiz);
- bibliotēkas vārds (piemērā *MEDUSimport*), uz kuru .NET programmā vajadzēs atsaukties 'References' sadaļā; tur vajadzēs ievietot atsauci uz attiecīgo DLL failu (*MEDUSimport.DLL*) un uz MATLAB interfeisa bibliotēku *MWArray.DLL,* kas tiks ieinstalēta kopā ar MATLAB Runtime (atrodama mapē …/MATLAB/MATLAB Runtime/<versija>/toolbox/dotnetbuilder/ bin/win64/v.2.0);
- autors, e-pasts, firma- izstrādātāja informācija;
- namespace (piemērā *MEDUSimport*), kuru .NET programmā vajadzēs pievienot ar *'using'* rindas palīdzību;
- .NET modulī pieejamās klases un tajās izmantojamās metodes. Modulī var iekļaut vairākas klases un katrā no tām vairākas metodes;
- .NET ietvara versija: 4.0;
- pārējiem parametriem ieteicams saglabāt automātiski ģenerētās vērtības;
- jāsaglabā projekts (failā <bibliotēkas vārds>.PRJ).

- Lai sagatavotu .NET moduļa instalācijas paketi, no rīku joslas jāizsauc funkcija 'Package'. Rezultātā zem MATLAB darba mapes tiks izveidota mape ar vārdu, kas vienāds ar bibliotēkas vārdu, un zem tās- 3 apakšmapes: 'for_redistribution'; 'for_redistribution_files_only'; 'for_testing', kā arī fails 'PackagingLog.txt', kas ir pakošanas procesa žurnāla fails. Mapē 'for_redistribution' tiks iekopēts instalācijas fails 'MyAppInstaller_web.exe', kurš jānogādā .NET lietojumprogrammas izstrādātājiem. Ja būs izvēlēts instalācijas paketes variants ar iekļautu MATLAB Runtime, tiks izveidots arī instalācijas fails 'MyAppInstaller_mcr.exe'. To jālieto datorā, kurā tiek pirmo reizi instalēts sagatavotais .NET modulis ar MATLAB funkciju(-ām).

- Datorā, kurā tiek gatavota MEDUSPILE programma, sagatavotais .NET modulis jāinstalē, startējot 'MyAppInstaller_web.exe' (vai pirmo reizi instalējot pirmo .NET moduli- 'MyAppInstaller_mcr.exe'). Jāizmanto noklusētās parametru vērtības.



- .NET programmas projekta sadaļā 'References' jāpievieno:
  - atsauce uz moduļa DLL bibliotēkas failu, kas atrodams instalētajā mapē, piem.'C:\IECS\MEDUSimport\MEDUSimport.DLL';
  - pirmo reizi instalējot pirmo .NET moduli, no MATLAB Runtime instalācijas mapes jāpievieno arī 'MWArray.DLL' bibliotēka, kas nodrošina .NET programmas saiti ar MATLAB Runtime bibliotēku;

- .NET programmas projekta konfigurācijās jāizvēlas "target=x64";

- MATLAB funkcijas izsaukšanai jāizveido jauns tās klases (piem. ‘MEDUSimport.Class1’) objekts, kurā iekļauta izsaucamā funkcija (piem. 'ImportPIX') un tās izsaukšanai jāizmanto 3 parametri: atgriezto rezultātu mainīgo skaits (vesels skaitlis); izejas parametru masīvs ar tipu 'ref MWArray[]', ieejas parametru masīvs ar tipu 'MWArray[]'. Datu tipu konvertācijai var izmantot MWArray.DLL iebūvētās iespējas [6]. Ieejas parametru masīvs satur vismaz 1 parametru- teksta rindu TxtParameters, kurā tiek nodoti tekstveida parametri formātā "param1=<value>; param2=<value>;…" (pēc pēdējā parametra atdalītājs ‘;’ nav obligāts). Piemērā zemāk šajā rindā tiek nodots importējamā faila pilns vārds kā parametrs `fileName=<faila vārds>`, projekta un attiecīgās failu mapes nosaukums kā parametrs `projectDirectory=<projekta vārds>`, kā arī parametrs `showImage`, kas norāda, ka importētais attēls funkcijai jāilustrē speciāli tam atvērtā logā.

  Piemērs (konsoles lietojumprogramma):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using MEDUSimport;
using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;

namespace MEDUSimport
{
  class Program {
     static void Main(string[] args) {
        MEDUSimport.Class1 obj = null;

        MWArray input = null;
        MWArray[] result = null;
        String err = "";

        try
        {
            // sagatavo ieejas parametru
            input =
(MWArray)"fileName=D:/Ints/MEDUS/Specifikācija/MEDUSpile/MEDUSimportInstall/Dati/pl130.pix;showImage
;projectDirectory=kārļi";

            obj = new MEDUSimport.Class1();      // izveido klases instanci
            obj.importPIX(3, ref result, new MWArray[] { input });
            int returnCode = (int)(MWNumericArray)result[0];

            if (returnCode == 0)
            {
                MWStructArray imageOut = (MWStructArray)result[1];
                String name = imageOut.GetField("Name").ToString();
                String filename = imageOut.GetField("Filename").ToString();
                String fileType = imageOut.GetField("FileType").ToString();
                int samples = (int)(MWNumericArray)imageOut.GetField("samples");
                int lines = (int)(MWNumericArray)imageOut.GetField("lines");
                int bands = (int)(MWNumericArray)imageOut.GetField("bands");
                int visBandR = (int)(MWNumericArray)imageOut.GetField("VisBandR");
                int visBandG = (int)(MWNumericArray)imageOut.GetField("VisBandG");
                int visBandB = (int)(MWNumericArray)imageOut.GetField("VisBandB");
                double pixelSizeX = (double)(MWNumericArray)imageOut.GetField("PixelSizeX");
                double pixelSizeY = (double)(MWNumericArray)imageOut.GetField("PixelSizeY");
                double x0 = (double)(MWNumericArray)imageOut.GetField("X0");
                double y0 = (double)(MWNumericArray)imageOut.GetField("Y0");
                // imageOut satur importētā attēla RSimage objekta info
            }
```

```
            } catch {
                throw;
            }
        }
    }
}
```

Piemērā izmantotās MATLAB funkcijas interfeiss ir šāds:

```
[returnCode, ImageOut, errorTxt] = importPIX(TxtParameters);
```

No C# programmas šī funkcija tiek izsaukta šādi:

```
String TxtParameters = "<tekstveida parametru virkne>";
MWArray[] result = null;
obj.importPIX(3, ref result, new MWArray[] { TxtParameters });
```

, t.i. funkcijai tiek nodots viens ieejas parametrs- tekstveida parametru virkne TxtParameters, bet 'result' masīvā tiks atgriezti 3 izejas parametri:

- returnCode = 0, ja izsaukums veiksmīgs, vai -1, ja nē;
- ImageOut – MWStructArray klases masīvs, kurā ir informācija RSimage klases objekta izveidošanai;
- errorTxt – kļūdas paziņojuma rinda, kura ir piepildīta, ja returnCode= -1;

- Lai pārbaudītu, .NET programmas izstrādes vide gatava MATLAB kompilēto moduļu izmantošanai, jāizveido konsoles lietojumprogrammas projekts ar piemērā parādīto galvenās programmas C# saturu un saskaņā ar aprakstītajiem soļiem; šī programma jāstartē skaņošanas režīmā, jāieliek apstāšanās punkts un jāizpilda pa soļiem. Funkcijas `'importPIX'` izpildes laikā tiks nolasīts PIX formāta attēls no faila, kura pilns vārds tiks nodots ar tekstveida parametru virknes locekļa `fileName=<faila vārds>` palīdzību. Nolasītā attēla saturs tiks parādīts atsevišķi atvērtā logā, kura nosaukums satur importētā faila vārdu.

- Ja .NET programmā ir nepieciešams darboties ar MATLAB izveidotiem datu masīviem, jāņem vērā, ka MATLAB datu masīvus atmiņā glabā pa kolonām (pirmais indekss mainās ātrāk), bet C,C++,C#- pa rindām.

2.pielikums. Importa, apstrādes un eksporta funkciju .NET moduļu izmantošanas testa piemērs

Testa piemērs satur vienas importa, vienas apstrādes un vienas eksporta funkcijas izsaukumus, lai uz tā bāzes varētu sagatavot visu šo dažādo funkciju izsaukšanas piemēru testēšanai. Piemēra kods:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using MEDUSimport;
using MEDUSexport;
using MEDUSprocess;
using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;


namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // izsauc no GUI
        private void testPIX(object sender, EventArgs e)
        {
            MEDUSimport.Class1 obj = null;
            MEDUSprocess.Class1 obj2 = null;
            MEDUSexport.Class1 obj3 = null;

            MWArray input = null;
            MWArray[] result = null;
            String err = "";

            try
            {
                // sagatavo ieejas parametru
                input = (MWArray)"fileName=D:/Ints/MEDUS/Specifikācija/MEDUSpile/MEDUSimportInstall/Dati/pl130.pix;showImage;projectDirectory=kārļi";

                obj = new MEDUSimport.Class1();      // izveido klases instanci
                obj.importPIX(3, ref result, new MWArray[] { input });
                int returnCode = (int)(MWNumericArray)result[0];

                if (returnCode == 0)
                {
                    MWStructArray imageOut = (MWStructArray)result[1];
```

```
String name = imageOut.GetField("Name").ToString();
String filename = imageOut.GetField("Filename").ToString();
String fileType = imageOut.GetField("FileType").ToString();
int samples = (int)(MWNumericArray)imageOut.GetField("samples");
int lines = (int)(MWNumericArray)imageOut.GetField("lines");
int bands = (int)(MWNumericArray)imageOut.GetField("bands");
int visBandR = (int)(MWNumericArray)imageOut.GetField("VisBandR");
int visBandG = (int)(MWNumericArray)imageOut.GetField("VisBandG");
int visBandB = (int)(MWNumericArray)imageOut.GetField("VisBandB");
double pixelSizeX = (double)(MWNumericArray)imageOut.GetField("PixelSizeX");
double pixelSizeY = (double)(MWNumericArray)imageOut.GetField("PixelSizeY");
double x0 = (double)(MWNumericArray)imageOut.GetField("X0");
double y0 = (double)(MWNumericArray)imageOut.GetField("Y0");
// imageOut satur importētā attēla RSimage objekta info

obj2 = new MEDUSprocess.Class1();       // izveido klases instanci
input = (MWArray)"bandNIR=12;bandVIS=7;showImage;projectDirectory=kārļi"; // sagatavo ieejas parametru

obj2.processPIXcalcNDVI(3, ref result, new MWArray[] {imageOut, input });
int returnCode2 = (int)(MWNumericArray)result[0];
if (returnCode2 == 0)
{
    MWStructArray imageOut2 = (MWStructArray)result[1];
    String name2 = imageOut2.GetField("Name").ToString();
    String filename2 = imageOut2.GetField("Filename").ToString();
    String fileType2 = imageOut2.GetField("FileType").ToString();
    int samples2 = (int)(MWNumericArray)imageOut2.GetField("samples");
    int lines2 = (int)(MWNumericArray)imageOut2.GetField("lines");
    int bands2 = (int)(MWNumericArray)imageOut2.GetField("bands");
    int visBandR2 = (int)(MWNumericArray)imageOut2.GetField("VisBandR");
    int visBandG2 = (int)(MWNumericArray)imageOut2.GetField("VisBandG");
    int visBandB2 = (int)(MWNumericArray)imageOut2.GetField("VisBandB");
    double pixelSizeX2 = (double)(MWNumericArray)imageOut2.GetField("PixelSizeX");
    double pixelSizeY2 = (double)(MWNumericArray)imageOut2.GetField("PixelSizeY");
    double x02 = (double)(MWNumericArray)imageOut2.GetField("X0");
    double y02 = (double)(MWNumericArray)imageOut2.GetField("Y0");
    // imageOut2 satur izejas attēla RSimage objekta info

    obj3 = new MEDUSexport.Class1();    // izveido klases instanci
    input = (MWArray)"fileName=L:/testPIX.tif;showImage;projectDirectory=kārļi"; // sagatavo ieejas parametru
    obj3.exportPIXgeotiff(3, ref result, new MWArray[] { imageOut, input });
    int returnCode3 = (int)(MWNumericArray)result[0];
    if (returnCode3 == 0)
    {
        int ok = 1;
        // pārbaudi, vai ierakstīts fails L:/testPIX.tif
    }
    else
    {
        err = ((MWCharArray)result[2]).ToString();
```

```
                        }

                    }
                    else
                    {
                        err = ((MWCharArray)result[2]).ToString();
                    }
                }
                else
                {
                    err = ((MWCharArray)result[2]).ToString();
                }
            }
            catch
            {
                throw;
            }
        }

    }
}
```

Piemērā izmantoto MATLAB funkciju interfeisi ir šādi:

```
[returnCode, ImageOut, errorTxt] = importPIX(TxtParameters);
// TxtParameters virknes saturs: fileName=<faila vārds>; projectDirectory=<projekta vārds>;
showImage""

[returnCode, ImageOut2, errorTxt] = processPIXcalcNDVI(ImageOut,TxtParameters);
// TxtParameters virknes saturs "bandNIR=<joslas numurs>; bandVIS=<joslas numurs>; showImage;
projectDirectory=<projekta vārds>")

[returnCode, StructOut, errorTxt] = exportPIXgeotiff(ImageOut, TxtParameters);
// TxtParameters virknes saturs "fileName=<eksportējamā faila vārds>;projectDirectory=<projekta
vārds>")
```

Visi izsaukumi atgriež vienādus izejas parametrus:

- returnCode = 0, ja izsaukums veiksmīgs, vai -1, ja nē;
- ImageOut – MWStructArray klases masīvs, kurā ir informācija RSimage klases objekta izveidošanai;
- errorTxt – kļūdas paziņojuma rinda, kura ir piepildīta, ja returnCode= -1;

Projektā tiek iekļautas atsauces uz 3 moduļiem: *MEDUSimport, MEDUSexport* un *MEDUSprocess*, katrs no kuriem domāts savai funkciju grupai.

Kā pirmā tiek izsaukta modulim *MEDUSimport* pievienota MATLAB funkcija *importPIX* ar 1 ieejas un 3 izejas parametriem.
Ieejas parametrā *TxtParameters* jānodod teksta parametru virkne ar šādiem locekļiem:

- „fileName=<PIX formāta faila vārds ar pilnu ceļu un paplašinājumu 'PIX'>;", to izvēlas lietotājs dialogā, piem. „pl130.pix". Blakus jābūt arī info failam ar to pašu vārdu, bet pievienotu paplašinājumu 'HDR', piem. „pl130.pix.hdr" ;
- „showImage;" , ja tas ir, funkcija attēlos importēto attēlu, citādi nē;
- „projectDirectory=<projekta nosaukums>;", piem. „projectDirectory=kārļi;" , norāda uz eksistējošu projektu, kam atbilst projekta mape failu sistēmā, to funkcija meklēs saknes mapē „L:/MEDUSPILEprojects" (šādai mapei jāeksistē, startējot programmu). Izpildes rezultātā importētais attēls tiks ielasīts mapē „L:/MEDUSPILEprojects/<projekta saknes mape>/RSimport", piem. failā „L:/MEDUSPILEprojects/kārļi/RSimage/ pl130.mat". Pirms funkcijas „importPIX" izsaukšanas šai mapei jau jāeksistē.

Funkcijas izsaukšanas rezultātā tiek izveidota struktūra *imageOut*, kurā ir informācija par importēto attēlu, ko vajag izmantot *RSimage* objekta izveidošanai.

Kā nākošā no moduļa *MEDUSprocess* tiek izsaukta apstrādes funkcija *processPIXcalcNDVI,* kura domāta NDVI attēla veidošanai no PIX attēla *bandNIR* un *bandVIS* spektra joslās, un kurai ir 2 ieejas parametri: struktūra *imageOut,* kas izveidota *importPIX* izsaukuma laikā un satur informāciju par *RSimage* tipa attēlu; kā arī parametrs „TxtParameters" ar šādiem locekļiem:

- „bandNIR=<NIR spektra joslas numurs>;"
- „bandVIS=<VIS spektra joslas numurs>;"
- „showImage;"
- projectDirectory=<projekta nosaukums> (sk. aprakstu augstāk).

Projektējamajā programmatūrā nododamo struktūru *imageOut* būtu jāizveido no izvēlētā datubāzes tabulas *RSimage* ieraksta. Funkcija *processPIXcalcNDVI* izveido un saglabā citu *RSimage* tipa attēlu un atgriež informāciju par to struktūrā *imageOut2*, kas projektējamajā programmatūrā jāizmanto, lai izveidotu jaunu *RSimage* ierakstu par šo attēlu.

Pēdējais no fukciju izsaukumiem (*exportPIXgeotiff*) ilustrē eksporta funkcijas izmantošanu no moduļa *MEDUSexport*, kas importēto PIX formāta attēlu saglabā GeoTIFF formāta failā. Šīs funkcijas izsaukumam ir 2 ieejas parametri: struktūra *imageOut*, kas izveidota *importPIX* izsaukuma laikā un satur informāciju par *RSimage* tipa attēlu; kā arī parametrs „TxtParameters" ar šādiem locekļiem:

- „fileName=<eksportējamā faila vārds ar pilnu ceļu un paplašinājumu „TIF";
- „showImage;"
- projectDirectory=<projekta nosaukums> (sk. aprakstu augstāk).

Ja nav kļūdu, funkcijas rezultāts ir ierakstītais TIFF formāta fails.