INTELLIGENCE ISSUES AND NOVEL APPROACH TO BUILD SOFTWARE FOR IMAGE PROCESSING

Alexei BELOTSERKOVSKY, Alexandr NEDZVED, Pavel LUKASHEVICH

United Institute of Informatics Problems, National Academy of Sciences,
Department of Intelligent Information Systems,
Minsk, Republic of Belarus

e-mail: abelotser@newman.bas-net.by

Abstract

In this paper an elegant concept is proposed, which is for process management in applied analytical systems. Described schemes are currently used for image processing, and allow designing adaptive graphic user interface based on intelligent social (recommending) component so to build-up a functionality with flexible scripts and use objective and reliable algorithms to solve applied tasks

Keywords**:** *scripting language, GUI, automated image analysis, process management*

1 INTRODUCTION

Nowadays, there are plenty of software libraries, commercial and developed by consortiums under Open Source conception for any applied field, and image processing is not an exception. Usually, to support such software tools is not trivial for users thus there are specific requirements to software organization. It depends on solving tasks, the software user skills, and the environment of user's workplace. Therefore, modern requirements to image analysis software is dynamic organization of functionality, user interface, data management for image processing methods, quality and distance control, etc.

It is quite complicated to replicate in computerized applications tools for automated image analysis if we need a full universal support of workplace in several ways [1]. Thus, we come to the necessity of using the already popular notion of Software Flexibility. It mustn't be confused with a flexible development methodology – "Agile software development", – the concept, which is quite popular as a new feature of software applications. The generally accepted definition of software flexibility has been given by IEEE [2] (Flexibility: "The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed"). It can be challenged on a number of positions. In addition there is already a number of metrics at the moment, characterizing the degree of flexibility of the developed environment [3] but sometimes it contradict with each other. This issue is actually not the

subject of this paper, since authors are pursuing the global goal to offer the architecture of applied task-driven system that would later allow:

- aiding image analysis, by making the results quantitative, reproducible, and objective;
- selecting relevant subsets of features in diagnostic tasks, by tracing the set of features most commonly used by experts;
- discovering new knowledge, by allowing the extraction of novel features and the execution of statistical tests for evaluating their significance;
- supporting decision making since it can be considered as a first step in the development of a knowledge base for automated analysis and diagnosis.

At this stage, we propose conceptual schemes for process management of image processing system (as an application domain), and generating a user interface, with the intellectual, social (collaborative, recommending) component according to specific needs of certain users and developers. It allows modifications and building-up the functionality due to the wide use of workflows (processing scripts), which obviously will contribute to solve a problems described above.

2 SCRIPTING AND SOFTWARE FLEXIBILITY

In software application, the script – is a program that automates certain task that user would perform manually using interface without script [4, 5]. Script languages allow developers to mix and "interlock" with different functions or packages, as well as to coordinate the results obtained by the system as a whole. To create a flexible environment, or a software system with an "open architecture", scripts are efficient in terms of their integrating level as a tool which modifies and extends functionality of developed software for image analysis. A material published by Guillaume Marceau [6] has been used to choose the best script language, which is planned to be a base core of software system with an "open architecture". Parameters of 72 implementation of programming language are compared in his study and applied to 19 special tests, which had been prepared in the frame of the project "The Computer Language Benchmarks Game". Different parameters were evaluated such as speed of code execution, code size, memory needed to implement certain functions.

The flexibility and particular functionality of a language Lua [7] has allowed to define it as the most suitable one for the task to be solved. Through the use of a core interpreter Lua, we have finally gained following advantages:

- Flexible integration of individual tasks in a single operating space. Implementation of the various components is invariant to assembling and adapting process for particular tasks.
- Problem decomposition into sub-tasks, taking into account system properties, that is more efficient allocation of jobs between team members and program developers, simplifying the process of debugging.
- Simple portability (without changings and recompiling) of already implemented and compiled components in new projects and their integration into the new

connections system, circuit of functioning and information exchange between other components.

- The ability to use well-functioning compiled scenarios of build as a basis for new project with new or updated functionality.
- Designing a system based on the principle "from simple to complex", increasing gradually the functionality of system addressing the priorities of the project.

Depending on peculiarities of a problem the structure of software being developed is organized so that C + + as the base language of the majority functions can simultaneously extend Lua. C-functions call Lua-function and vice versa, for instance. The basis of the symbiotic relationship between Lua and its host language is a virtual stack, a LIFO-data structure ("last in-first out") which stores function arguments and results temporarily. To call base language from Lua (and vice versa), the caller side pushes values into the stack and calls target function. The recipient pops arguments from the stack, processes data and pushes result into the stack back. When control returns to the caller side, it retrieves values from the stack. Using C API, an application can also receive information from Lua-structure by calling any Lua-function of C. The reverse is similar. Lua can load and call functions on demand.

As a result of Lua integration into the program it has been discovered the opportunity to write and export C-functions into Lua programs, and use it directly from the script of the program. In addition, such functions can be created and used as separate specialized DLL-modules (with Lua supporting interface). In view of high prevalence of language there are a large number of already developed modules. There is also an opportunity to use common dll-libraries with a specially designed interface functions.

Using described above methodology for flexible software environment we gain a simple management with a look of application and rapid interface adaptation to manifold tasks of image analysis. As a result, interactive part of application can be reduced to its minimum, which greatly simplifies the work with program and extend its usability.

3 PRINCIPLES OF FLEXIBLE GRAPHIC USER INTERFACE

Graphic User Interface (GUI) of an application consists of two sets of functions: basic static and extensible set of functions, which is controlled by Lua-scripts (Fig.1). Complex GUI-elements are programmed in C++ as a Lua-function as a single component. It covers such forms as a widget to display and edit images with the appropriate tools, interactive dialogues (for instance, interactive threshold segmentation or image histogram displaying), resulting tables and text editor for Lua-script. Simple elements are binding directly in Lua-functions so to cover: menu control, panels with control elements and universal dialogue that includes necessary elements to call Lua-functions.

The basic principles of working with such an application are as follows:

1. GUI - DLL is managed by Lua being a root element of the interface (all the elements sit here in the container).

2. Positioning of elements in the container is performed by using GUI. It locates and draws the element according to rules. Main containers are frame, vertical sizer, and horizontal sizer.
3. Events handlers are defined as functions that are attached to widgets.
4. Event loop is starting after creating a dialogue.
5. Universal dialogue covers all necessary elements to call Lua-function.

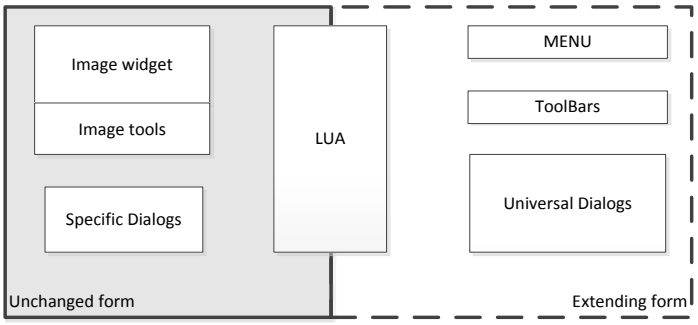

Figure 1 GUI of flexible application

The technology aimed at issues how to arrange process management so to:

- adapt interface to needs of certain specialist (user) and use available algorithmic experience of users;
- modify algorithms according to this or that task and then again provide appropriate interface to interact with user;

Adaptation of GUI is performed by using Recommending System or on the base of Intelligent Agent which can analyze features of processing images.

4 USING INTELLIGENT AGENT TO MANAGE SCRIPTS

When architecting a task-driven system using proposed conceptions we suppose that there are at least two different schemes for generating GUI with intelligent support. Systems which are built by these schemes are described below. We also propose Intelligent Agent (IA) which is the Agent mentioned above [8]. It is embedded into next schemes.

Inherently, the engine of IA is the same for all schemes. The only difference is in fact that for one case the GUI script generation executes according to a pre-existing set of estimates of processing scripts based on extracted features of certain image. For another case estimates are formed adaptively with the number of users' interest with a different application (for us it is image processing domain). Thus quality of estimation increases with increasing of number of users or intensity of using the program (if system runs locally for one user).

4.1 Intelligent Agent Without Collaborative Filtering

GUI for this case is generated by Lua-scripts based on calculating characteristics of processed image (for example: display panel for texture characteristics if texture is under investigation).

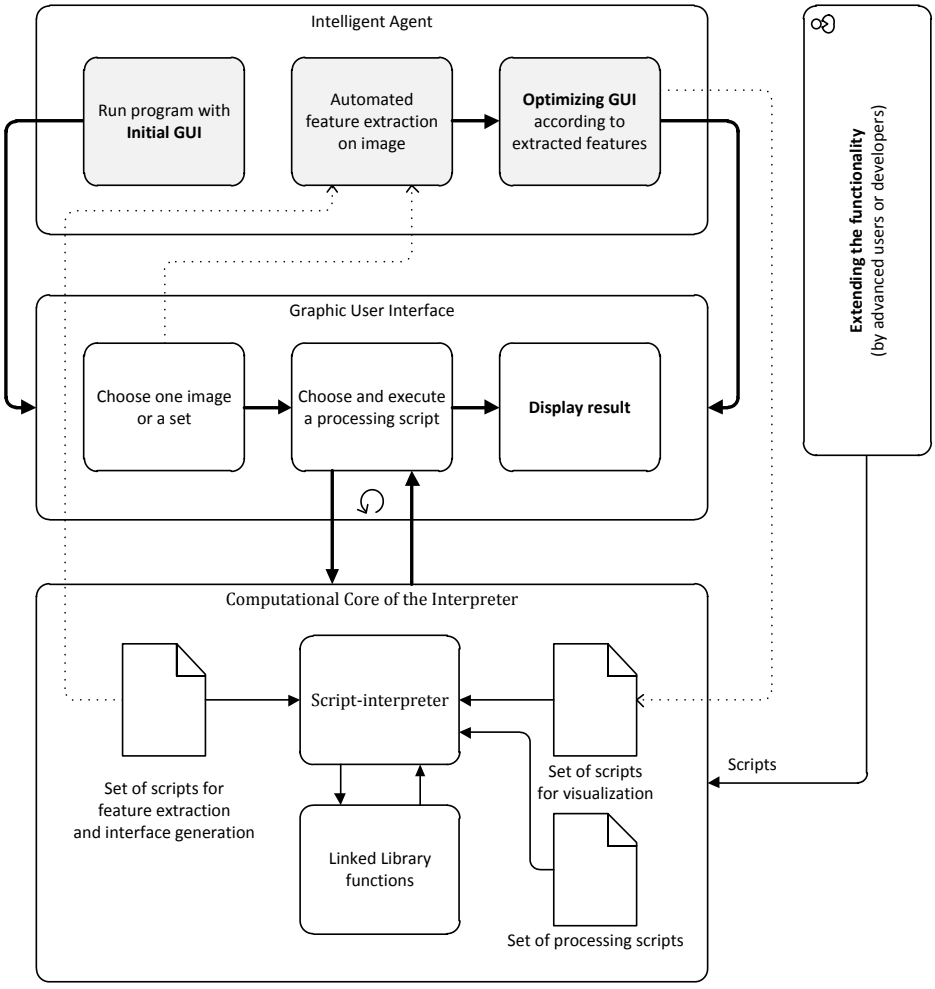

Figure 2 Baseline Image Processing System (Client) with Extensible Functionality without filtering

System is running here in a fully autonomous mode and does not use any other information except the image, thus a set of guidelines (recommendations) is laid by developer in advance during its creation and distribution (Fig.2). For example, when photo images are to be downloading, algorithms for color correction, quality improving, and

texture analysis will be actualized at GUI, and so on. When dealing with binary images morphological and distance transformations, skeletonization, blobs extraction will be at frontend of system interface.

One may see some restrictions for this scheme, but through the script-implementation of the intellectual part, advanced users have the ability to change both algorithmic (computational) functionality and interface. Also competitive versions of code and interface modifications can be easily distributed to users of the given application. (replacement of scripts in the Intranet, Internet, via e-mail, etc.). Despite its apparent simplicity, the above approach has proved to be very functional in practice.

4.2 Intelligent Agent With Collaborative Filtering

GUI for this case is generated by Lua-scripts based on calculating characteristics of processed image (for example: display panel for texture characteristics if texture is under investigation). GUI here is also generated by Lua-scripts based on calculating characteristics of used image together with IA. However, recommendations are being giving under Collaborative Filtering which collects and processes statistics of certain function usage (Fig.3).

The main difficulty is still remained how to create a controlling script – IA which could support logic adaptation of GUI for a given type of images. This problem is easily and efficiently solved for a fixed set of image features and pre-set number of scripts through training on the test sample. However, we originally positioned the system as a flexible and scalable one, so a question arises to adapt IA for advanced feature sets and algorithms.

To address these difficulties, it is proposed a method to adapt the logic of the Agent based on program usage history. IA calculates characteristics of a test images and puts into a local database a fact of using a specific set of algorithms and frequency of its use for a given characteristics (features). Further, this local database is analyzed to construct new user interface, to make it actual and personalized.

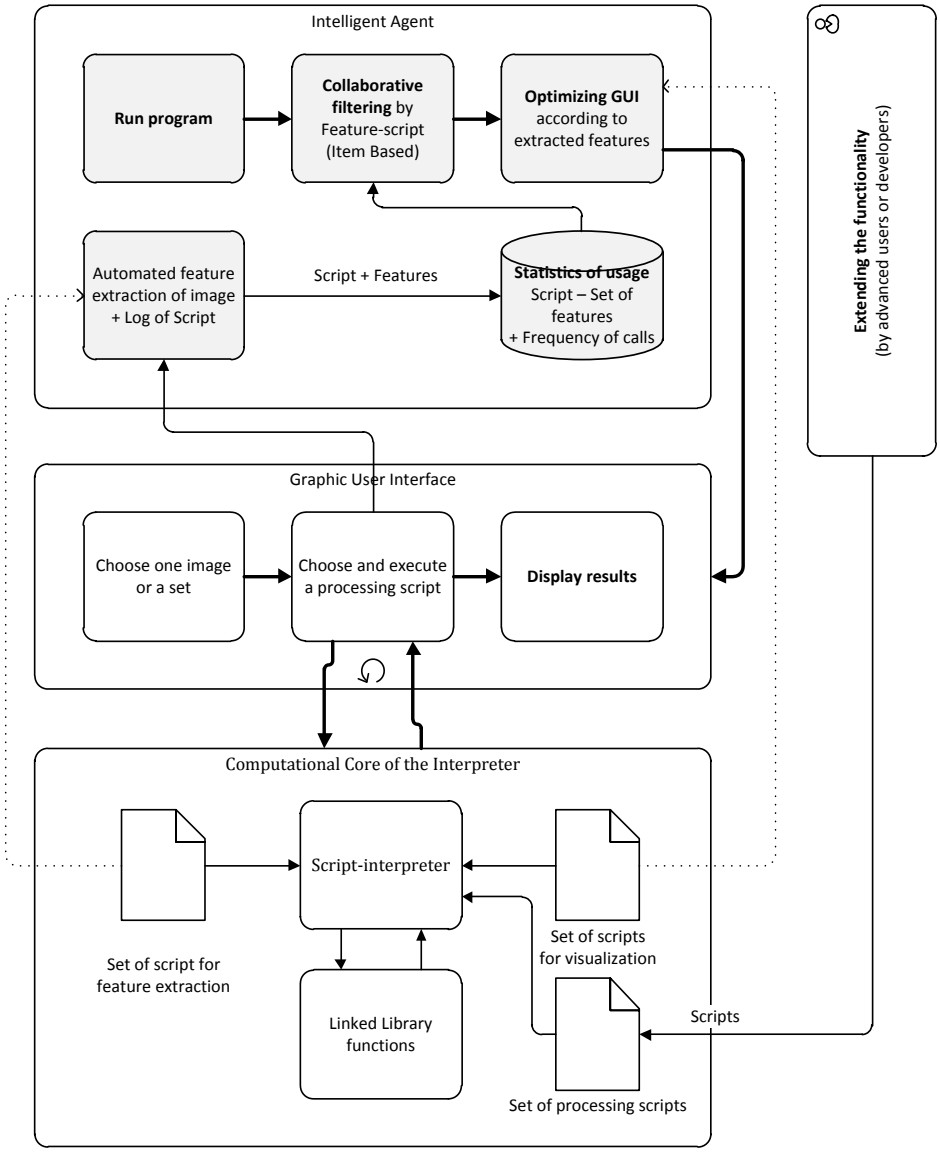

Figure 3 Baseline Image Processing System (Client) with Extensible Functionality supported with Collaborative Filtering

Thus local database with such statistics can not only recommend a unique interface for a given images with certain properties, but also to form a personalized basis (slightly variable backbone) GUI for the user, showing features for certain domain.

4.2.1 Operating principle

Operating principle lies in three different modes of collaborating system used with different information volume about preferences of a particular user:

- So called “cold start” when sufficient information has not yet gathered (for example, just after registration) and recommendations is made for an average user.
- Passive user’s feedback when no user action is required since all necessary information is gathered passively by taking statistics of specific functions usage statistics (intensity of use).
- Active user’s feedback when user manually estimates algorithms by using a five-point scale marks, so the quality of the recommendations increases significantly if base assessments increase. This strategy is optimal for fast start.

So it is offered for the first scenario (“cold start”) to personalize the interface to use either average estimate of algorithm by all users, estimate averaged by user in the field of knowledge (if available). One can specify fields of image types: satellite images, medical CT images, histological or cytological images, nanoscopic structures, etc.

For the second and third scenarios we propose to use collaborative filtering.

We are assuming that adding “social” component and the use of other users’ reviews will allow them to adapt quickly a “weakly variable backbone GUI” "even for “cold start”. It will be done right after registration based on the user’s field of interest and profile.

4.2.2 Collaborative Filtering

Collaborative filtering – this is one of approaches for forecasting in recommendation systems, which use known preferences and estimates of users’ group to predict the unknown preferences of another person. [9]

The system which is discussing here has properties similar with recommendation system. It basic assumption is following: those users who are assessed these or that objects of any kind in the past, could tend to give similar estimates for other subjects in the future. Decisions are made individually for each user, but used information gathered from database. Thus, collaborative filtering is different from a simple approach that gives the average score for each object of interest, for example, based on the number of votes cast. Research in this area is actively underway at the moment, so there are several methods which use different mathematical apparatus to solve such a problems.

There are two main approaches for CF: item-based and user-based.

Item-based strategy builds an item-to-item (image- to- image in our case) matrix, determining relationships between pairs of images and then recommends a set of the most suitable scenarios for current image (on the basis of image to scenario usage statistics) (Fig. 4).

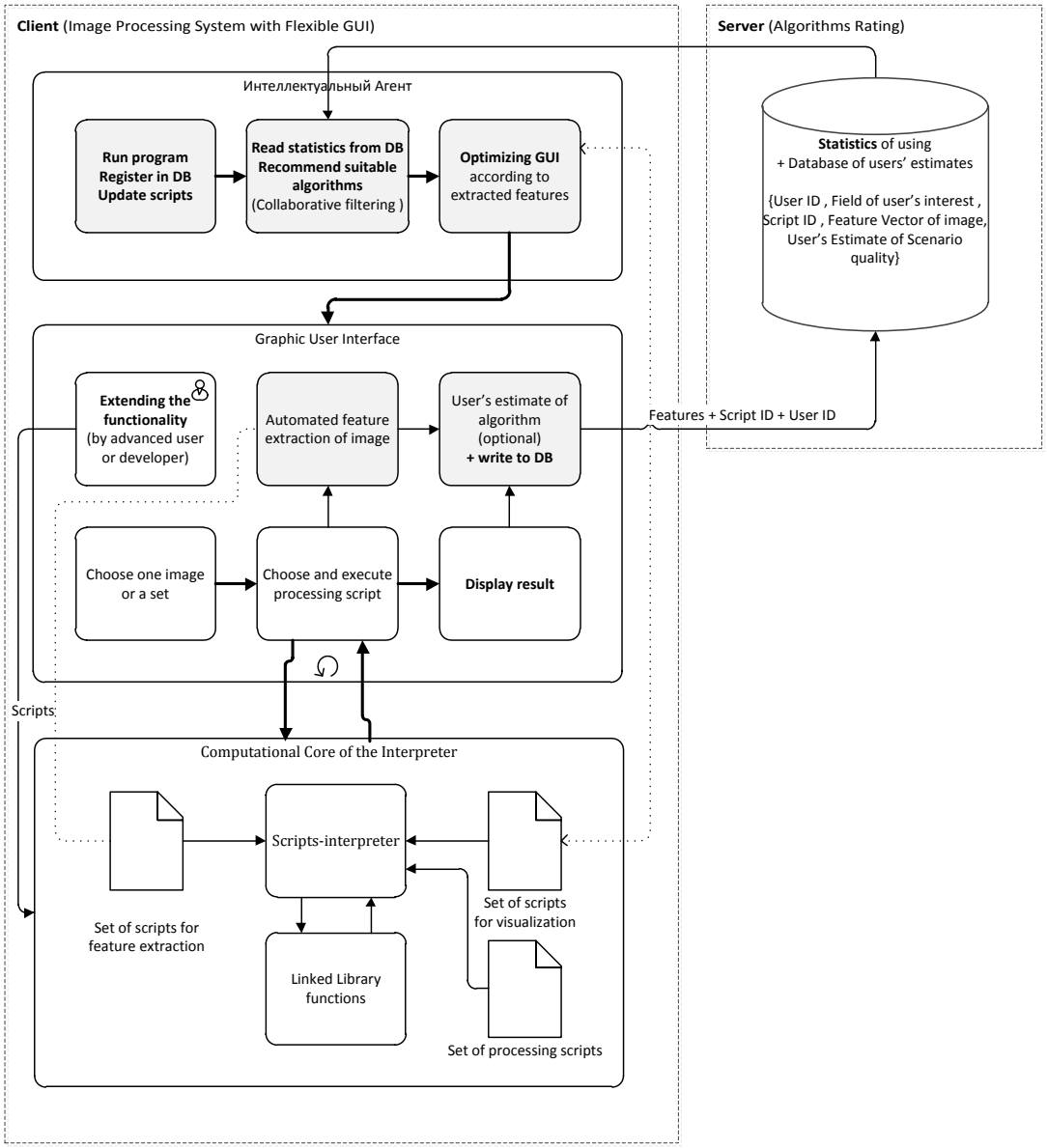

Figure 4 Client-Server architecture with remote server for user statistics collection

User-based strategy looks for users with similar rating patterns with the active user and makes recommendations based on their preferences.

Describing these approaches more precisely is out of current topic so the only point here is to understand the difference between second and third scenarios which lies in counting user's action or not. Thus, "Item-to-Item" collaborative approach proposed by Amazon [10] is reasonable for passive feedback. Comparing to the original approach, we

have been made a slight modification: the event of script is accepted for further processing if the same script has been executed at least 10 times.

For active feedback a "Slope One" algorithm [11] is used since it is a well-proven algorithm for solution-based collaborative filtering for items (objects) with users' estimates.

5 REMOTE REPOSITORY FOR EXTENDING FUNCTIONALITY

Overall the server part of the proposed concept allows us to gather statistics of scripts' usage and effectively apply techniques of collaborative filtering to generate personalized user interface based on the Log (history) of his reviews (or statistics regarding scripts' use) as reflected in Fig.5.

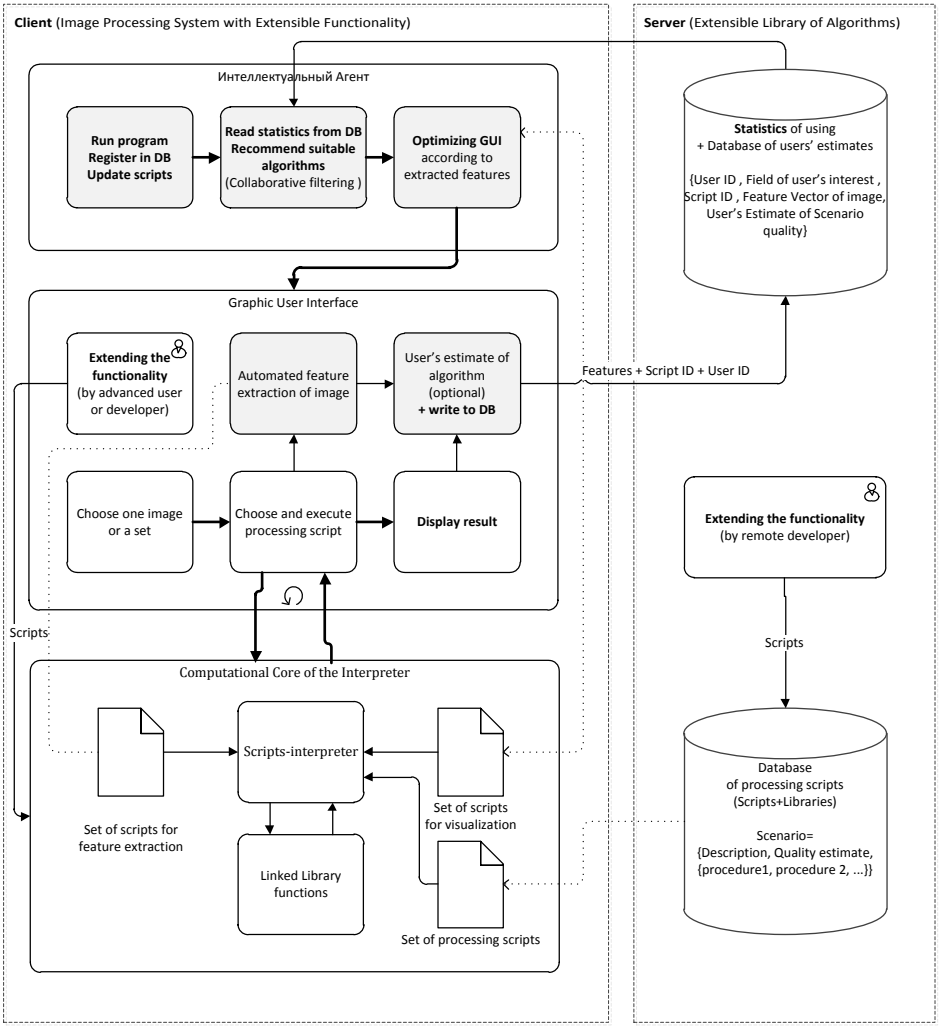

Figure 5 Client-Server architecture with remote repository supported with Intelligent Agent

However, developing the idea a convenient way of central storage (Repository) can be considered, so to apply simple mechanisms for update the database of image processing algorithms, for instance.

Central Repository of image processing algorithms has a number of advantages, among which:

- using actual scripts for imaging and image processing;
- hot fix and add scripts;
- minimum cost for extending the functionality of a program;
- add new scripts without recompiling directly from the interface of the client part of the program;
- crowd-sourcing script developing and deployment (joint development, approbation and including into script repository).

6 CONCLUSION

Described methodology is based on using script language as a kernel of system to modify Graphic User Interface (GUI) and generate new functions to extend overall functionality, so to put a flexibility making work much more convenient. Thus it increases system's usability. It is supposed that proposed conception can be extended to application of software as itself. So the future trend is to generate and use sets of new programmable functions which can be reconfigured by Lua-scripts under Intelligent Agent support. Finally, as outlook we are coming to a web-service or Cloud in application domain with remote Repository of adjustable and reliable algorithms, thin clients and personalized workplaces for solving this or that applied problem.

REFERENCES

[1] Belotserkovsky, A: Bulding flexible environment for automated tumor CT-monitoring: Innovative Technologies for Medicine. Proc. of 4th International Forum, Bialystok, Poland, 1-3 december 2010. - P. 22.

[2] IEEE. Standard Glossary of Software Engineering Terminology 610.12-1990, Vol. 1. Los Alamitos: IEEE Press, 1999.

[3] Eden A.H., Mens T. : Measuring Software Flexibility. IEEE Software Vol. 153, No. 3, June 2006

[4] Bloom B., Hirzel M.: Robust Scripting via Patterns. Proc. on Dynamic Languages Symposium (DLS), 2012, pp 29-40.

[5] Ierusalimschy R., Programming With Multiple Paradigms In Lua. Functional and Constraint Logic Programming Lecture Notes in Computer Science. Volume 5979, 2010

[6] Marceau G.:: The Speed, Size And Dependability Of Programming Languages. Blog "Square Root Of X Divided By Zero", 2009 (http://gmarceau.qc.ca/blog/2009/05/speed-size-and-dependability-of.html).

[7] Ierusalimschy R., Henrique de Figueiredo L., Celes W.: Passing a Language Through The Eye Of a Needle," Queue 9 (5) 2011.

[8] Nedzvedz A., Belotserkovsky A., Gurevich I., Trusova Yu., Yashina V.: Self-developing Software For Image Processing Tasks. 8th Open German-Russian Workshop "Pattern Recognition and Image Understanding", November 21-26, 2011, Nizhny Novgorod, The Russian Federation: Workshop Proceedings. - Nizhny Novgorod Lobachevsky State University, 2011.

[9] Su X., Khoshgoftaar T. M.: A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 2009

[10] Linden G., Smith B., York J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, January/February 2003.

[11] Lemire D., Maclachlan A.: Slope One Predictors for Online Rating-Based Collaborative Filtering. In SIAM Data Mining, Newport Beach, California, April 21-23, 2005.