UDC 004.042

O. Linevych

HETEROGENEOUS ARCHITECTURE FRAMEWORK FOR VIDEO ANALYSIS SYSTEM

Abstract: From the point of view of neurobiology the processes of human analysis of sound, images, video are investigated. From the point of view of software engineering, architectural solutions of systems that automate these processes are analysed. Based on the obtained data for the new automation system it has been developed an architectural solution that improves the quality of the video analysis process due to considering images and sounds; has a flexible and scalable structure compared to others, so the system can adapt faster to changes in requirements.

Keywords: Video recognition, architectural styles, software quality metrics, neural networks, LSTM.

Problem description

Currently, software systems that automate the analysis of processes / events work with data formats such as audio, image and video format. Video contains more information about processes / events, as it contains synthesized audio and visual information about them. Hence, video analysis was chosen as an analysis problem that can be solved more effectively. Humans perform analysis more efficiently than automated systems because they perceive audio and visual information. This fact was confirmed after a study of seventy-six analysis systems, which analysed in 98% only one aspect - sound or image. Two percent of systems that analysed both aspects did so less effectively than humans. Such systems could identify objects and short-term events by marker sounds, such as a dog's growl or a gunshot. However, no system has been found that builds a meaningful logically related description of video processes based on audio and visual information synthesis.

Formulation of the problem

To identify the steps of human analysis that will be automated by the system, to analyse what abstractions can represent the process of video analysis in the human body; to create based on abstractions potential mechanisms (software components), their properties and behaviour; to develop an efficient, flexible, scalable architectural solution; to compare the developed architectural solution and existing ones.

Problem decision

The process of human analysis according to [1] is represented by stages:interest in a particular event/object (reaction to a stimulus accompanied by concentration);

1. classification (recognition) of the perceived event / object;

2. evaluation - to evaluate the information about the event / object in the "knowledge base" of the brain;

3. inference if necessary, based on evaluation. Inference is experience. The experience is stored in memory.

A stage is a process that, according to General system theory [2], is considered as a system. Process objects can also be represented as systems with certain properties and behaviours. Process processes and objects are described in classes or logically connected class associations - components. The component can be described by type and its detailed implementation.

Described above human analysis stages were translated into an architecture solution of a structure:

1. Stage of interest is described as a separate component in IMotiveService and MotiveService classes.

2. Recognition stage. A person recognizes audio (component described in classes IAudioRecognition, AudioRecognition) and visual (class IVisionRecognition) information. However, visual information comes in several types - images or text. Therefore, visual information can be specified in two classes: ImageRecognition and TextRecognition, based on this component of visual information recognition is described by 3 classes - IVisionRecognition, ImageRecognition, TextRecognition.

3. Stage of evaluation of the received information. Described as a separate component in the IPriorityService, PriorityService classes.

4. Stage of inference. Described as a separate component in the classes IRecognitionInference, RecognitionInference.

5. As an auxiliary component was selected a mechanism that manages memory. It is described in IMemoryService, MemoryService classes. Memory is used to store inferences (certain patterns of behaviour, for example) or edit them (adaptation - assimilation and accommodation). Note that the human brain uses different types of memory in accordance with the tasks.

The described components are visualized in a class diagram:

To compare the created architectural solution with another existed solutions were selected:

1. popular today API Keras (used by NASA, YouTube, Waymo), which is part of the project of neuro-electronic intelligent robotic operating system [3]. Open for general use on GitHub. Functionality: recognition of a set of images in video, text, audio by means of neural networks of various types;

2. blackboard robotic system HEARSAY-II [4] language recognition on the basis of which systems for monitoring sonar signals (HASP), airspace monitoring (TRICERO), etc. were created. The system is not open source, but it is able to check some parts of code in its documentation.

Items 8-12 are calculated for the recognition components.

Comparison is described in Table 1. Analysis system comparison.

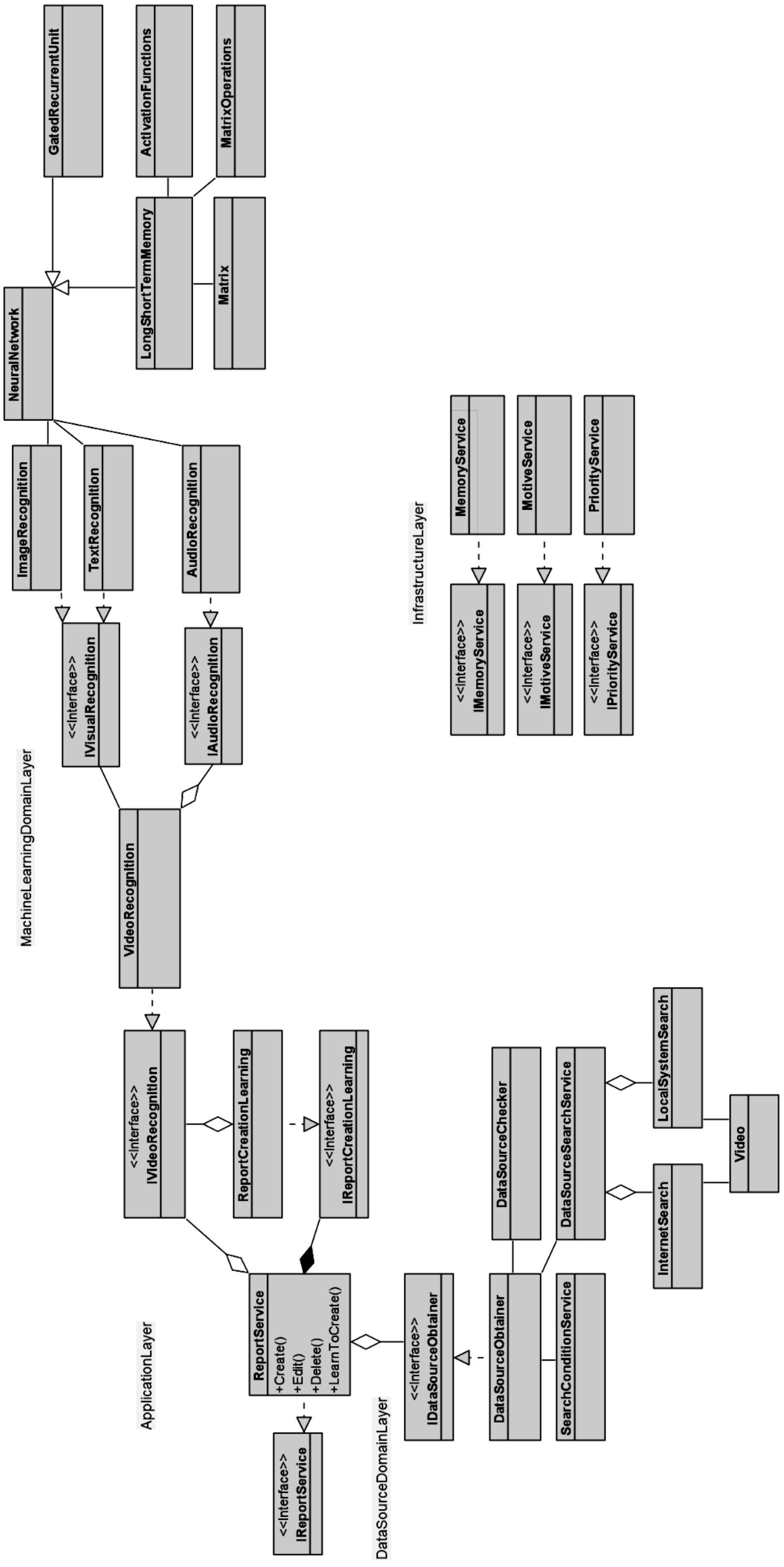

Figure 1. Video analysis system class diagram

Table 1

Analysis system comparison

№	Trait	Analysis System	Kesar	HEARSAY-II
11	system tasks	video image / sound recognition, combination of recognition and presentation of recognition in text form as a report	video, audio image recognition and text recognition in the form of a report	conversation in the form of dialogue with a person (the task of language recognition)
в2	architectural style of the system	object-oriented and service-oriented based on layers with components developed in feedback, feedforward styles, combined mechanisms of event-driven style	object-oriented and service-oriented	interpreter, blackboard, layered
33	decomposition according to the rules of categorization of abstractions	+	+- the components are highly connected and therefore constructed in violation of the rules	+
4	availability of interfaces (allows you to replace the implementation)	+ to analysis have 4 interfaces - there is a possibility to replace algorithm of training of the analysis, the analysis of video / images / sounds / text	- there are no interfaces for video analysis	+ replaceable components: 1 for each knowledge source + it is possible to replace the implementation of the language recognition status board
5	ability to substitute algorithm	+ (due to interfaces it(i.e. using of Dependency inversions principle) it is possible)	- (no interfaces for algorithms, thus it is hard to substitute)	+- (recognition component is in a separated component, thus can be substituted)
6	the ability to change the data type	+	-	-
7	ability to edit component functions	+	-	+
8	instability of the analysis component [5] $= Ce \div (Ce + Ca)$ Ca is afferent couplings - incoming component dependencies; Ce is efferent couplings - outcoming	c. video analysis: 3/(3+1) = 0,75. Other c. Analyses are encapsulated	c. video analysis (video_transformers.py): 10/(10 + 1) = 0,9. Unstable. This problem can be solved by encapsulating (from customer classes) data	closed information

Continuation of the table

№	Trait	Analysis System	Kesar	HEARSAY-II
9	depth of inheritance(Depth of inheritance [6]). characterizes the system as one that can be expanded and not duplicate code, which saves space and speeds up code execution	neural networks component is inherited DIT = 1 (no deep inheritance, neural networks component can be effectively expanded)	neural networks component is inherited (file base_layer.py base class Layer) DIT = 1 (no deep inheritance, neural networks component can be effectively expanded)	closed information

10	cyclomatic complexity McCabe(M) [7]
	
	A high score (more than 10) means that it is difficult to read the code
		for class VideoRecognition.cs
		
		M =< 3-5
			for class video_transformers.py, training.py…
			
			M > 4 more than in 24 functions.
			
			Function build(self, input_shape) M = 18
			
			High cyclomatic complexity in most classes
				closed information

11	Design smells[8]			
	flexibility	+ flexible, because there are interfaces, the mechanism of inheritance is used, polymorphism for work with neural networks	- no interfaces found, thus inflexible	+ clear division at the level of abstraction, the absence of monolithic classes
	rigidity	- components are separated by interfaces, implementation does not have direct dependencies	+ because of monolithic classes with a large number of responsibilities	-
	immobility	- System is divided by domain processes on components, its parts are separated by interfaces and inheritance mechanisms, thus they can be substituted fast	+ because of monolithic tightly bound classes labour-intensive to make, for reuse in another system, k. analysis or related	+-

End of table

№	Trait	Analysis System	Kesar	HEARSAY-II
	code opacity	+ Classes, function, variables are written according to domain terminology clean code practices, thus code can be read fast	- Classes, function, variables are written not according to domain terminology, to understand code it takes a lot of time due to monolithic classes, high value of cyclomatic complexity, ambiguous names of variables	+ Classes, function, variable names are written according to domain terminology
12	Abstraction quality metrics [9]			
	coupling[10]	for video analysis component = 4 3 (aggregate recognition component) + 1 (theoretical error handler)	for video analysis (video_transformers.py) = 10 (aggregate components)	blackboard component that contain with recognized data status depends on knowledge sources, thus = knowledge sources amount(according to diagram its more then 6)
	cohesion (LCOM4, characterized by the number of responsibilities of the class)	for video analysis component = 1 image/audio/text analysis components = 1	for video analysis = 5 Responsibilities > 1: uploading, editing video, working with video list, working with model, preparing video	speech analysis component (according to its blackboard architecture) = 1 (because speech component has 1 responsibility)
	completeness	+ (public functions are interrelated according and put classes devised/invented by categorization methods)	- (public functions logically connected spread out over incorrect classes chosen not by according categorization methods)	+ (public functions are interrelated according and put classes devised/invented by categorization methods)
	sufficiency	+	- classes have more than 1 responsibility (violation of the Single Responsibility principle)	-
11 3	see the metrics of the generalized evolutionary process for the parity of all components of the analysis, formula (1) [10]	10 video analysis component + к. image/audio/text analysis + neural networks c. + matrix c. + matrices operations + activation functions + memory c. + motives c. 20 through an additional mechanism of motives and work with memory costs of evolutionary changes by 2 more	10 analysis c. + base layer c. 1 + 7 neural networks components = 18	-

$$\overset{\mu}{Modules}(\varepsilon) \triangleq \Sigma_{m\in\Delta Modules\,(i_{old}, i_{adjusted})}\,\mu(m) \quad (1)$$

Description:

1. Modules represent any notion of module that is appropriate for the circumstances, such as class, procedure, method, and package.
2. The notation t ≜ m (often) means: "t is defined to be m" or "t is equal by definition to m" (often under certain conditions).
3. μ represents any software complexity metric that is meaningful with relation to a particular module m. Software complexity metric that was used for the formula is a parity of all components of the analysis.

Conclusions

The developed solution is more effective, as it analyses both aspects of video (sound and image) and synthesizes the results of the analysis to build a logically related description. Also, the solution is flexible, as each step of human analysis was extracted to a separate component, which allows you to scale the system according to new requirements. Since the system was divided on the components according to the principle of Dependency inversion principle (with SOLID), the system can be edited in parallel by as many developers as components, as each component does not depend on the other direction.

REFERENCES

1. Lefrançois G. R. Theories of human learning. Belmont, CA: Thomson/Wadsworth, 2006. 418 p.
2. Simon H. The architecture of complexity. Proceedings of the American Philosophical Society. 1962. № 467.
3. API системи розпізнавання: сайт. URL: Keras. URL: https://keras.io/about/ (accessed 2 October 2021).
4. Nii P. Blackboard systems at the architecture level. Expert Systems with Applications. 1994. Vol. 7, № 1. P. 43-54.
5. Martin R. C., Martin M. Agile Principles, Patterns, and Practices in C# (Robert C. Martin Series). Hoboken: Prentice Hall PTR, 2006.
6. Martin R. OO Design quality metrics. An analysis of dependencies. Cranbrook Road: Green Oaks, 1994.
7. McCabe T. J. A Complexity Measure. IEEE Transactions on Software Engineering. 1976. Vol. SE-2, № 4. P. 308-320.
8. Eder J., Kappel G. Coupling and Cohesion in Object-oriented Systems. Institut fur Informatik: Universitat Klagenfurt, 1986.
9. Booch G. Object-Oriented Analysis and Design with Applications. San Jose: Addison-Wesley, 2007. 677 p.
10. Eden A. H. Measuring software flexibility using evolution complexity, 2006.