UDC 004.413:338.5

Nikolay O. Sidorov, D. E., Prof.
Vladimir A. Khomenko, assoc. Prof.
Viktor T. Nedovodeev, asst. Prof.

REENGINEERING OF THE AIR SIMULATORS LEGACY SOFTWARE

There are the technical complexes consisting of components, parts of which are actively used, but the rest has lost working capacity owing to moral and physical deterioration. An example of such a complex is the aviation-flight complex "plane-simulator". High cost of components which continue to be used (plane) do the actual task of restoring and supporting the out-of-order components (simulator). The considerable part of such complexes is the software, which owing to replacement of the obsolete and physically worn out hardware requires the rework. The rework method is reengineering.

Розглянуто успадковані технічні комплекси, які складаються з компонентів, частина яких активно використовується, а інша втратила працездатність унаслідок морального та фізичного зносу. Прикладом такого комплексу є авіаційно-пілотажний комплекс «літак-тренажер». Висока вартість компонентів, які ще використовуються (літак) роблять актуальним завдання відновлення та підтримки працездатності компонентів, що втратили працездатність (тренажер). Значною частиною таких комплексів є програмне забезпечення, яке через змінення морально застарілого та фізично зношеного апаратного забезпечення потребує відповідної переробки методом реінженерії.

Software Reengineering

In general, the software reengineering requires two processes - reverse and forward [1; 2].

Input of the reverse process (fig. 1) are the legacy software and the additional information about the domain.

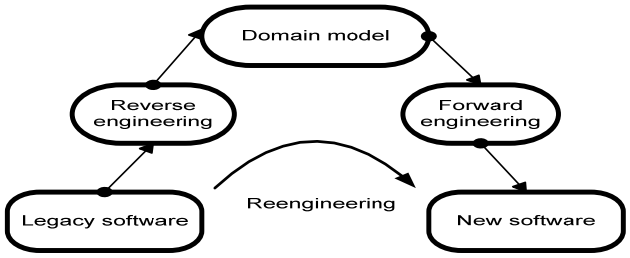

Fig.1. Processes of reengineering

The result of this process is the model - representation of the domain. This model then is used in the forward process for the new software creation.

In a context of engineering the domain of the software is an application area for which the software [3] is developed. The domain model is a description of the domain which is created by performance of one of the reverse engineering methods – design information recovery [4] or the domain analysis [3]. Irrespective of the method, for creation of the domain model the combination of the code, existing documentation, experience of staff, the common knowledge of a problem and application area [4] are used. Representation of domain model, depending on solved tasks and a domain maturity can be in the form of a taxonomy, functional models or domain language [4].

If model construction is carried out by the first of the specified methods of reverse engineering the legacy software plays an important role.

Reengineering is an effective method of reuse of the software – prolongations of the legacy software useful period which can be applied to reuse in different aspects – restoring, reuse and reworking of the software [5].

Software restoring is usually fulfilled in the process of its support. Reengineering appears as a method of struggle against ageing of the software which is characterised by number of symptoms [6]: code "pollution", loss of knowledge of the software, a bad lexicon (style) [7], easing of cohesion of the components, architecture "stratification". The reasons for these symptoms occurrence are eliminated in the operating software [8].

Preparation for a reuse can be fulfilled in the operating software (on its replication) or on the liquidated. Usually separate components of the software which are preliminary processed by application reengineering are reused – owing to changing of their functionality or owing to migration (new computers, the operating system, programming language).

Software rework at reuse is fulfilled when software migration is carried out. The case of legacy software migration to a new hardware platform is especially hard. The migration task arises owing to obsolescence and physical deterioration of the hardware platform. It causes the impossibility of using the computer and legacy software or its separate working parts execution. In this case, as a rule, both the operating system and the programming language are replaced.

Implementation of reverse engineering processes is associated wis two tasks solving [9]: evaluation of the expenses necessary for construction of domain model; estimation of quality of the reverse engineering processes and whole reengineering. The first task solving depends on the maturity of domain and can be grounded on models of software cost estimation requires [10]. Thus, the more mature domain is needed the less cost for reverse engineering processes realisation. Quality of reverse engineering processes is usually estimated, showing the developed software adequacy to the legacy software or the subject domain model [9].

The article under consideration suggests the method which is developed for application on the software, that is in operation in the subject domain [11] and presents the results of its usage for the aviation simulator complex TL410M (airplane L410) software rework after the hardware platform was replaced. The computer complex of the simulator constructed on the computer "Robotron" basis, was obsolete morally and physically, therefore requires the replacement by the modern hardware. The complex hasn't been maintained for more than fifteen years; there were disabled computers, units of the data exchange system, a considerable part of indicators in a cockpit and on the instructor console. Thereof executing of the simulator legacy software became impossible.

The rest of this article consists of three parts. In the second part the method of object controlled reengineering of legacy software is presented; in the third – case study of simulator software reengineering is described; in the fourth – adequacy of the reworked software is considered.

Object controlled reengineering of the legacy software

The article considers the method of reengineering of legacy software which was created for work in a subject domain. It is so-called E – programs [11], that are used on automation of the person or society activity. Thus, the software becomes a part of real environment. As a rule, the software of this type functions on hybrid (digital-analogue) computer complexes, and its considerable part is related to processing of the information circulating between real object or its model and the computer complex [12]. The structure of such complexes includes analogue to – digital and digital-to-analogue converters, sensors of real object or its model.

Features of a subject domain and E-programs define that legacy software reengineering of the considered type requires solving the traditional problems of recovering the design information and information about a real object or model. It also requires the special approach to the problem solving of the adequacy proof of the reworked software functioning in the real object or its model behaviour.

Real object information recovered at the reverse engineering process is the set of input parameters and their characteristics; characteristics of sensors, indicators and actuators of real objects or models. Information recover requires the usage of the traditional way: source code and documentation analysis and experimental researches of the real object or model.

Constructed software functioning adequacy proof cannot be carried out in traditional ways – comparison of the results of legacy and reworked software execution or comparison of results of reverse engineering with behaviour of corresponding model of a subject domain [9]. The first way cannot be used because there is no computing equipment on which it is possible to execute the legacy software, and the second – because the model of a subject domain, presented at the documentation, as a rule, contains errors. Besides, special position of the E – programs in the real world specifies that the basic attention at the adequacy proof should be given to the detailed analysis of the program behaviour in real operating conditions [11]. Thus, proof of functioning adequacy of the developed software real object or imitating model characteristics and properties in the real object should play the main role. It is the essence of the suggested method of the software reengineering (fig. 2).

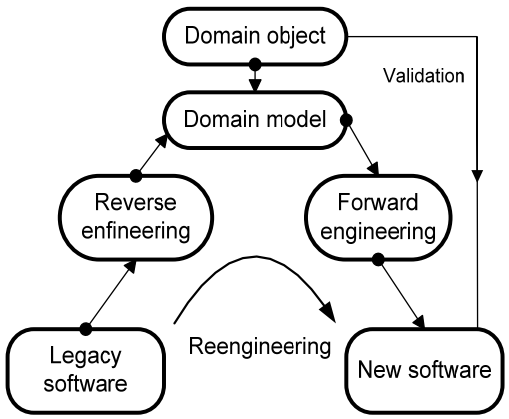

Fig. 2. The reengineering method scheme

Reengineering of the legacy software of aviation simulator TL 410M

The developed method was applied to the complex aviation simulator TL 410M (fig. 3).

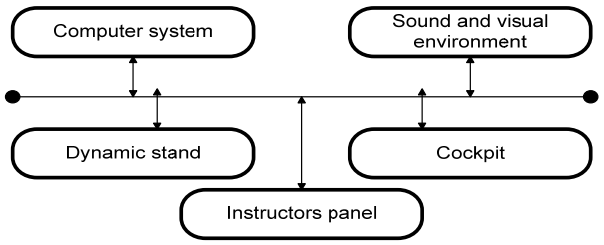

Fig.3. General scheme of simulator

The simulator computer system was built on the basis of computer "ROBOTRON 4201" and the analogue-digital data exchange system. The software was written using autocode. Listings of the legacy software are presented at the documentation that includes seven volumes with total amount nearby 32000 LOC. The simulator had got the out-of-order analogue sound surround simulator. For visual environment imitation the television simulator on the face-to-face monochrome projective system and the stationary tablet on instructor workplace (co-ordinators, an airdrome breadboard model) have been used. The dynamic stand did not work, either. The instructor panel contained the indicators that duplicate one in a pilot's cockpit and the television receiver for visual environment picture. The pilot's cockpit simulated the cockpit of a real airplane L410.

The general scheme of the implemented hardware and software migration is shown on the fig. 4.

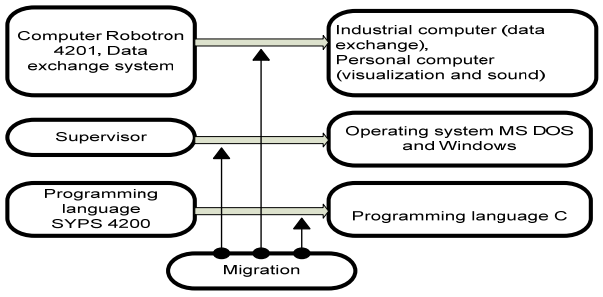

Fig.4. Simulator hardware and software migration

The main computer "Robotron" and the object communication device have been replaced by the industrial computer. The visualisation television system has been replaced by the computer system, based on the personal computer and a projector. On the industrial computer the MS DOS operating system was used and on the personal computer it was Windows. Migration of the legacy software was done from the assembler language to the high level language C. The restoration of the simulator electrical equipment that provides communication between computer system and a cockpit (cable system, power supply, functional nodes and intercom) was also executed.

As the created software could not be checked up on correctness of functioning by performance of the legacy software (there was no computer) the reverse engineering, except traditional processes, included process of additional mutual check of the legacy code and model. Checking up of the domain mathematical model and the legacy source code were raised due to such reasons:

– the limited information about modelling principles of airplane flight dynamics and systems which have been used by the developer during creation of a simulator (1972);

– presence of errors present in technical documentation that was casually or deliberately brought in the model descriptions and the source code.

The following aspects were checked:

– scaling of variables – by the recalculation of equations factors of the simulators mathematical models (comparison of the expressions resulted in the documentation and their interpretation in a code listings has shown essential distinctions);

– realisations of factors - are used by the legacy software developers due to the limited characteristics of the modelling computers;

– realisations of the mathematical problems solving methods.

The main bulk of the errors was in factors of the model equations that describe dynamics of flight. Thus, for the factors adjustment the parts of the legacy source code were used in which calculation factors were defined. Interpretation of these parts «behind the table» allowed to define the values of erroneous factors. The following example shows the casual or deliberate error in mathematical model.

In power-plant mathematical model the analogue parameter *nv* – number of the air screw rotors is presented by the modelling equation

$$n_v = \frac{k}{1+\tau p} \qquad (1)$$

where

k is amplification factor;

τ is screw time constant;

p is Laplas operator.

After a transformation, the definition (1) has the following differential equation form:

$$\frac{dn_v}{dt} = \frac{k - n_v}{\tau} = f(n_v). \quad (2)$$

According to the algorithm, applied in the modelling procedure, the equation (2) should be integrated by Euler's method:

$$n_{v(i+1)} = n_v + h \cdot f(n_{v(i)}), \quad (3)$$

where

h is an integration step.

In the source program procedure the equation solving (2) is represented by the following text:

```
16334 01 02 0416        S27     LDA DEVP”
16335 100400                    UVR
16336 140040                    LOA
16337 0405 76                   KAR 2
16340 00 04 0101                SPA AA2
16341 0405 75                   KAR 3
16342 140401                    EKA
16343 00 06 0101                ADD AA2
16344 00 06 1757                ADD K12
16345 01 07 0512                SUB NVP”
16346 0405 75                   KAR 3
16347 01 06 0512                ADD NVP”
…
16325 01 04 0512                SPA NVP”
```

where

DEV is value that specifies control lever position of the air screw;

K12 = 0,595 is the constant.

Comparison of figure (3) and result of interpretation shows that the text does not include integration step *h* (at engines work modelling it equals 0,12). This error was due to the calculations „behind the table” - the received transient process did not correspond to physical sense. Similar errors are present in other power-plant parameters calculation executed by Euler's method.

The legacy software (fig. 5) characterises discrete process of modelling with the 60 ms period.

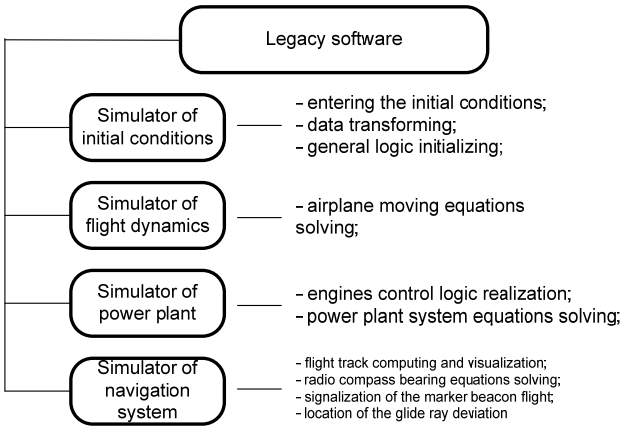

Fig.5. Legacy software structure

This period is used for calculating four simulators parameters and for finishing input-output operations. Thus, necessary speed was reached due to the following factors:

– simplification of the airplane flight dynamics and systems mathematical models;

– replacement of the real (dynamic) processes analytical description by dependences in the form of decision tables;

– application of simple and fast functions interpolation methods;

– use of a simple method of differential equations integration (Euler's method).

After the source code models check the reengineering of the legacy software was carried out in two stages: reverse engineering – creation of the high level algorithmic representation; forward engineering – creation of the new C-code software based on this algorithms.

For the reverse engineering the special tool – abstractor was built (fig. 6) [13; 14].

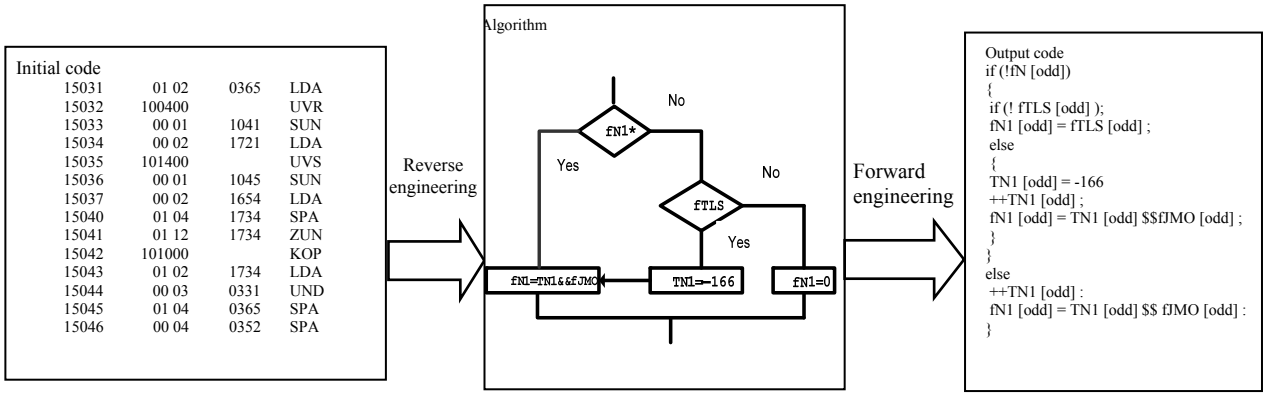

Fig.6. Reverse engineering result (part of the code)

Data exchange system

Components and structure of the new software of the simulator computer system is defined by the following:

– object of control is simulator, a real-time system, which requires the defined recall time;

– the computing system of a simulator should provide input and output of plenty of parameters (nearly 120);

– visualization of the simulator flight environment should be realized on the high-speed computer with the special characteristics of a video subsystem.

These factors define the distributed architecture of the simulator computer system, that includes several specialized computers and peripheral devices (fig. 7): an industrial computer for simulator models computing and input-output of data; computer for realization of the visual environment and noise simulators; a computer for the instructor board.

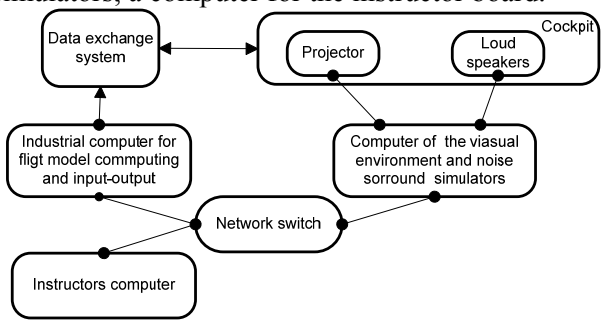

Fig. 7. Hardware architecture of the simulator modeling system

Data exchange between computers is organized through a local computer network on the Ethernet and protocol TCP/IP basis. Equipment of the computing system includes the digital-analogue IO-devices, the computer projector and the sound amplifier and loudspeakers. IO-devices are connected with one of the following types: analog-digital converters (ADC); digital-to-analog converters (DAC); logic signals IO-devices. All devices are the boards which are installed on the industrial computer on the ISA system bus.

The software of the data exchange system

The software of the data exchange system works on an industrial computer and is designed to provide the exchange of airplane simulators (models) with the simulator cockpit equipment. This software was created by adjustment of the specially developed pattern [15]. The software provides the following tasks:

– configuring of the IO-devices;

– control of the IO-devices;

– realization of network exchange under TCP/IP-protocol;

– provision of the standard interface in normalized parameters for exchange of IO-devices with models of the airplane.

The two-level modular architecture of the software provides the flexibility and scalability of a IO-subsystem (fig. 8).

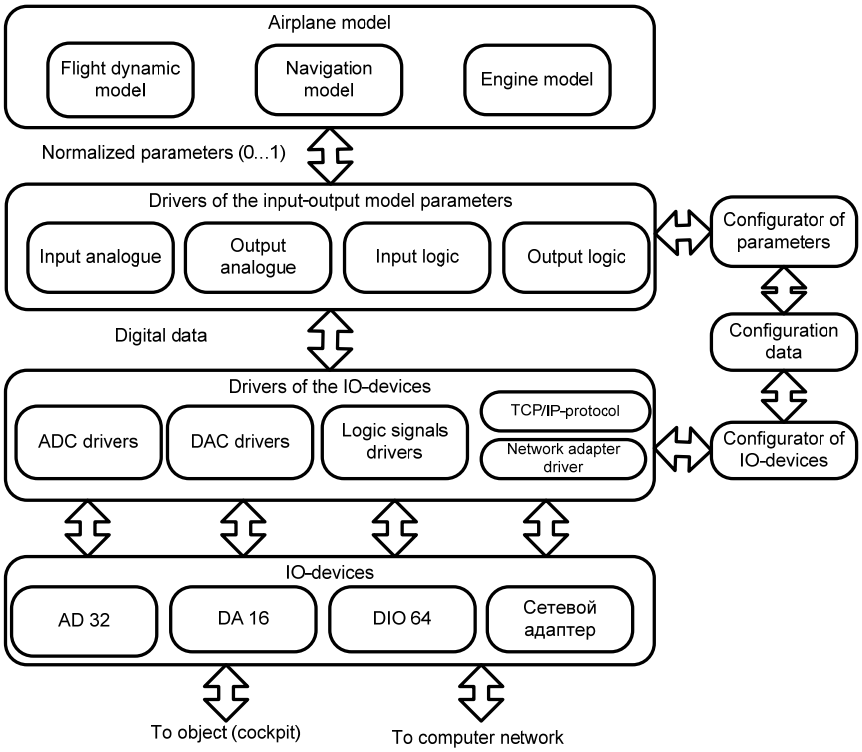

Fig. 8. Architecture of the simulator data exchange system

It includes the levels of IO-device drivers and drivers of parameters. Drivers functions provide control of signal and data conversion and digit data delivery. Driver level provides configuring of IO-devices during start and exchange control. The level of parameters drivers provides configuring of parameters which are inputted, outputted and converted - digital values of signals to normalized analogue values of models and vice versa.

The configurator of IO-devices reads the configuration data (numbers of boards, the addresses, numbers of interruptions) for each of configuration files, configures the devices automatically and tests their functioning.

The configurator of parameters stores the configuration and reads the data, necessary for normalization of each parameter. Each analogue parameter has the following configuration sets: parameter number; minimal and maximal values of parameter, minimal and maximal values of signals.

The software of the noise and visual environment simulators

Imitation of a pilot's sound surround in a simulator cockpit provides the acoustical information about simulated airplane modes. The sound in the airplane cockpit has the complex character and is formed by several sources of noise - engines and air screws, the wind outside a cockpit, the equipment inside, the landing gears and other equipment. The noise parameters depend on the work modes of their sources and flight modes. Noise simulators of legacy aviation simulators have the hardware realization based on mixture of sounds, created by several special generators. Each generator creates a sound of one airplane noise source and changes its characteristics depending on the source work parameters, transferred to the noise simulator from the simulator computer.

Three approaches were considered to restoration of the noise simulator during simulator reengineering (fig. 9). The essence of the first lies in the legacy simulator usage. It is obvious, that it can be applied, if it is possible to restore legacy imitator. In this case one problem must be solved - joining the simulator to the new computing system. The essence of the second approach lies in the development of a new noise simulator on the basis of the information about old simulator structure and characteristics. Such an approach demands the presence of the documentation about the legacy simulator or realization of the reverse engineering. The essence of the third approach lies in the development of a new imitator on the basis of the information received from the object of modeling. This approach was realized during the reengineering. It needs the development of the own imitator structure and measurement, analysis and formalization of the airplane sounds.

Any of the specified approaches requires the check of adequacy of the sound, received by means of the restored simulator, to noise of the airplane. For this purpose it is necessary to compare its characteristics to the characteristics which have been got from object (plane), or received by reverse engineering from the legacy simulator (fig. 9).

The airplane sound samples were used at the new noise simulator and reproduced by the player with adjustable parameters (play speed, loudness, echo effect, etc.). Such a method provides high speed of simulator development due to library of standard components (players, mixers, equalizers) and high realness of noise.

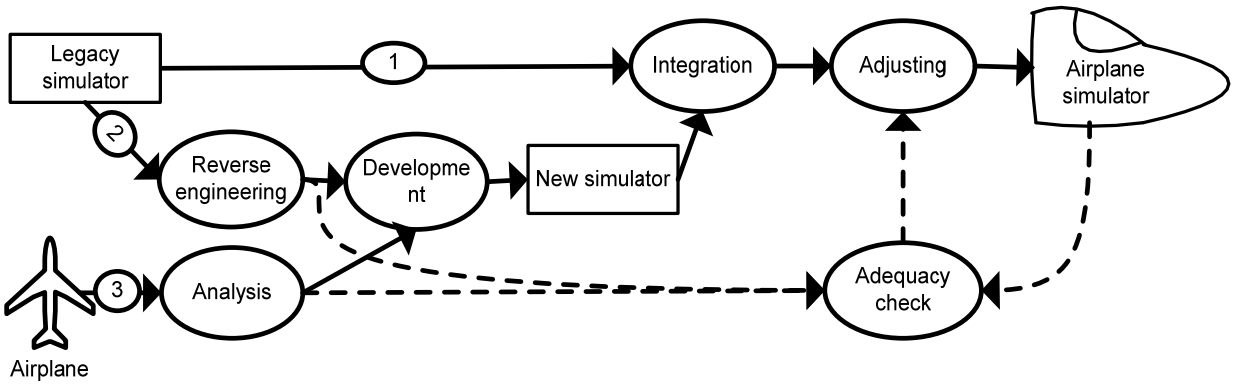

Fig. 9. Approaches to restoration of the noise simulator

In the process of reengineering the visual environment imitator was replaced [16]. It related to the progress at the visualization technologies obtained within last two decades. The first digital visualization imitators of CGA type appeared in the 70-ies. In the 90-ies the obtained digital image synthesizers qualities and working speeds allowed their usage in simulators. In the restored simulator, digital synthesis of images is carried out on the computer, which receives parameters of flight necessary for image formation (airplane coordinates, course, roll, pitch) from flight dynamics model. For the visual environment simulator integration into system the software pattern of exchange data system [15] is used.

Adequacy of the reworked software

For checking the reworked software functioning adequacy on the real object the quantitative and qualitative estimation are used.

The quantitative estimation was carried out in such ways:

– point - comparison of values of the calculated operational parameters with reference points in the description of simulator TL410 by the simulator developer technique;

– interval - comparison of the calculated characteristics with characteristics which are resulted in documents of the real airplane (flight guidance, technical operation guidance, the description of the plant system, the description on the avionics, data of flight record system).

Estimation of a quality of simulator behavior in different flight modes and stages is carried out by experts - pilots. Results of point estimation application are presented in the table, and results of interval estimation - on fig. 10, 11.

There are two approaches to a quantitative estimation of adequacy - determined and statistical [17].

Results of reworked software adequacy check based on the determined approach have shown the efficiency of the developed method of the legacy software reverse engineering.

The statistical approach to an estimation of adequacy is planned to be carried out after statistical experiment.

Point estimation of the TL 410M simulator adequacy

Flight part	Controlled parameter	Control value	Defined value
Acceleration	Time from the moment of release of block before achievement of speed V = 150 km/h	$t = 14 \pm 2$ s	14.7 s
Ascensional rate	Vertical speed of rise of the plane	$V_y = 10 \pm 1.5$ m/s	10.72 м/s
Flight characteristics	Cruising speed, rotation moment, pitch	$V = 250 \pm 25$ km/h; $M_к = 49 \pm 5$ %; $\upsilon = 4.5^0$	258.7 km/h 45.7%; 4.19^0
Acceleration in flight	Time of increase of the cruise speed of the plane $V = 200$ km/h, $V = 300$km/h	$t = 32 \pm 4$ s	31.44 s
Braking in flight	Time of decrease of cruise speed $V = 300$ km/h to $V = 200$ km/h	$t = 29 \pm 4$ s	31.14 s

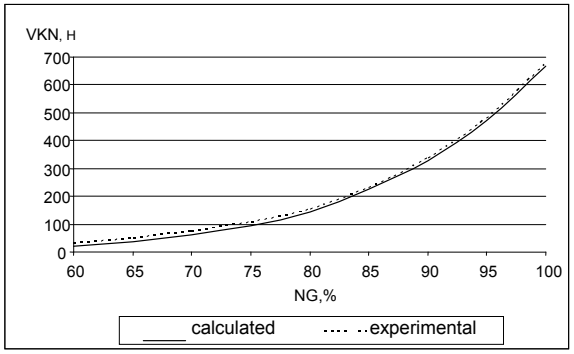

Fig. 10. Engine power (throttle characteristic)

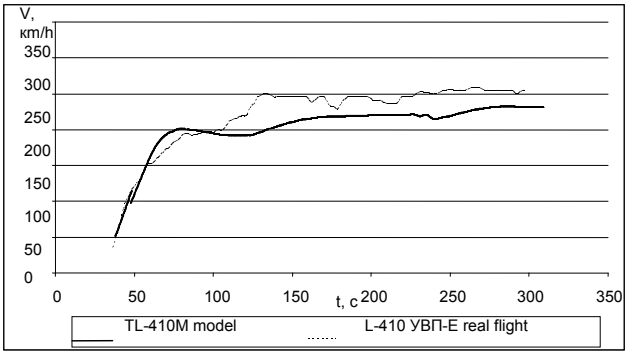

Fig. 11. Flight speed (dynamic characteristics)

The essence of the statistical approach is defined by the following [17]:
– each of compared objects (the airplane and simulator) has the statistical indeterminism;
– the volume of the information registered in flight is limited and is given to the researcher irregularly;
– adequacy is represented as a random variable, which probable distribution is formed consistently (in process of flight information receipt) on the basis of the statistical hypotheses test theory.
A method was developed for the statistical approach realization which on an example of a separately taken moment parameter adequacy estimation on the defined stage of flight has a following view:
1) on the simulator the statistical experiment on modeling of the defined flight situation is carried out and for the chosen parameter the modeling estimation of dispersion d^M is calculated;
2) the first real flight is carried out and is fixed the $x_1(t)$ - track of the chosen parameter change in time;
3) providing the real flight conditions, the researcher models a similar track on a simulator $y_1(t)$;
4) for the chosen section of time t^* is calculated the closure

$$z_1 = x_1 - y_1,$$

which is considered as the first sample element of the random variable z;
5) as the random variable z is formed by a combination of the large number of various random elements (the rests, closure, errors) according to the central limiting theorem of probability theory, distribution of the random variable z submits to the normal law

$$g_z(z/a) = \frac{1}{\sqrt{2\pi d^M (1-a)/a}} \exp\left[-\frac{z^2}{2d^M(1-a)/a}\right].$$

Then posterior distribution of adequacy, on receiving information z_1 is represented by the formula:

$$g_A(a/z_1) \sim f_A(a) g_z(z/a) = K_1 \left(\frac{a}{1-a}\right)^{\alpha_1} \exp\left[-\frac{\beta_1 a}{1-a}\right], \quad (4)$$

where
K_1 is continuous coefficient of proportion;
$\alpha_1 = 0{,}5;$
$\beta_1 = 0{,}5(x_1 - y_1)^2 / d^M.$

The second real flight is carried out and $x_2(t)$ - the next track of the chosen parameter change is fixed and the stimulator playbacks $y_2(t)$ - model track of the same parameter and $z_2 = x_2 - y_2$ is calculated, $f_A(a)$ is used as posterior distributor and now posterior distribution (4), received at the previous step is used.
New posterior distribution (taking into account z_1 and z_2) looks like:

$$g_A(a/z_2) = K_2 \left(\frac{a}{a-1}\right)^{\alpha_1+\alpha_2} \exp\left[-\frac{a}{1-a}(\beta_1 + \beta_2)\right],$$

where

$K_2 = \text{const};;$

$\alpha_2 = 0{,}5;$

$\beta_2 = 0{,}5 \cdot (x_2 - y_2)^2 / d^M.$

After items (2-7) multiplying repetition the posterior distribution of adequacy comes nearer to the stable form, and it's mode defines true value of object adequacy (airplane) and the software (simulator) on the researched parameter.

Conclusion

Application of the software reverse engineering allows not only to prolong the out-of-date scientific and technical complexes use, but also to realize its improving support.
For example, for a flight-modeling complex the reverse engineering provides the following: an opportunity of new functions addition; perfection of simulators models; application of modern integration methods.

References

1. *Сидоров М.О., Іванова Л.М., Хоменко В.А.* Методологічні принципи реінженерії програмного забезпечення успадкованих авіаційних тренажерів // Матеріали VIII міжнар. наук.-техн. конф. "Авіа-2007". – К.: НАУ, – 2007. – Т.1. – С. 13.119–13.122.
2. *Chikofsky E.J., Gross J.H.* Reverse Engineering and Design Recovery: Taxonomy // IEEE Software. – Jan. 1990. – P. 13–17.
3. *Prietto-Diaz R.*. Domain Analysis: An Introduction.- Software engineering Notes. – 1990. – Vol. 15, № 2. – P. 47–54.
4. *Biggerstaff T.J.* Design Recovery for Maintenance and Reuse // Computer. – 1989. – July. – P. 36–49.

5. *Сидоров Н.А.* Восстановление, переработка и повторное использование программного обеспечения // УСиМ. – 1998. – № 4. – С. 71–79.
6. *Visaggio G.* Ageing of a Data Intensive Legacy System: Symptoms and Remedies // J. Software Maintenance and Evolution. – 2001. – Vol. 13, No 5. – P. 281–308.
7. *Сидоров Н.А.* Стилистика программного обеспечения // Проблемы программирования. – 2006. – № 2–3. – С. 245–255.
8. *Bianci A., Caivano D., Visaggio G.* Iterative Reengineering of Legacy Systems // IEEE Transactions of Software Engineering. – 2003. – Vol. 29, No. 3. – P. 225–241.
9. *Rugaber S., Stirewale R.* Model-Driven Reverse Engineering // IEEE Software. – 2004. – Jul/Avg. – P. 45–53.
10. *Сидоров Н.А., Баценко Д.В., Василенко Ю.Н, Щебетин Ю.В.* Модели, методы и средства оценки стоимости программного обеспечения // Проблемы программирования. – К., 2006. – № 2–3. – С. 290–299.
11. *Леман М.М.* Программы, жизненные циклы и законы эволюции программного обеспечения // ТИИЭР. – Proc.IEEE. – 1980. – Т. 68. – С. 26–46.
12. *Советов Б.Я., Яковлев С.А.* Моделирование систем: Учеб. для вузов. – М.: Высш. шк., 2001. – 343 с.
13. *Сидоров Е.Н.* Метод реінженерії успадкованого програмного забезпечення авіаційного тренажера // Тез. доп. всеукр. конф. аспірантів і студентів «Інженерія програмного забезпечення 2007». – К., 2007. – С. 26.
14. Маніцков М.К. Зворотна інженерія, як основа технології відновлення роботи авіаційного тренажера // Тез. доп. всеукр. конф. аспірантів і студентів «Інженерія програмного забезпечення 2007». – К., 2007. – С. 4.
15. Хоменко В.А., Сидоров Е.Н, Мендзебровский И.Б. Шаблон программного обеспечения устройств связи с объектом авиационных тренажеров // Проблемы программирования. – К., 2008. – С. 30–40.
16. Кузнецов С.В., Холод К.О. Формирование матриц проекции для компьютерных генераторов изображения при использовании полупрофессиональных проекционных систем // Матеріали IV Міжнар. наук. конф. студентів та молодих учених. – К., 2004. – С. 35.
17. Недоводеев В.Т. Байсевская оценка адекватности модели полета // Сб. науч. тр. – К.: КИИГА, 1992. – 40–50 с.

The editors received the article on 3 June 2008.