розробити модель захисту мовної інформації від витоку по технічних каналах;

розробити метод підвищення ефективності захисту мовної інформації за рахунок корекції спектру створюваних шумів;

розробити методику захисту мовної інформації на основі стохастичних диференціальних рівнянь зі змінними коефіцієнтами;

виробити рекомендації щодо захисту мовної інформації на ОІД спецпризначення.

ДЖЕРЕЛА

1. Концепція технічного захисту інформації в Україні [Текст]: постанова Кабінету Міністрів України від 8 жовтня 1997 року № 1126 // Урядовий кур'єр. – 1997. 12 листопада. – С. 3. [Електронний ресурс] – Режим доступу: http://zakon.rada.gov.ua/cgibin/laws/main.cgi?nreg=1126-97-%EF\.

2. Доктрина інформаційної безпеки України [Текст]: указ Президента України від 8 липня 2009 року № 514/2009 // Офіційний вісник України. – 2009. № 52. С. 7. [Електронний ресурс] – Режим доступу: http://zakon1.rada.gov.ua/cgi-bin/laws/main.cgi?nreg=514%2F2009.

3. Хорев А.А. Способы и средства защиты информации. – М.: МО РФ, 2000. – 316 с.

4. Хорев А.А. Техническая защита информации: учеб. пособие для студентов вузов./ В 3-х томах. Т. 1. Технические каналы утечки информации. – М.: НПЦ «Аналитика», 2008 – 436 с.

EFFECTIVENESS EVALUATION OF ALGORITHMS TESTING AT THE STAGE OF SOFTWARE DETAILED DESIGNING

Drobko Olena, Kolodiazhnyi Ihor

(Scientific supervisor PhD in Information Technology Oleksandr Dorenskyi)

Central Ukrainian National Technical University

Nowadays information systems and technologies develop rapidly. They are accompanied by the development of new software that should be secure, highly productive, flexible, functional, convenient, portable as well as given the opportunity to be upgraded and improved. Moreover, designing of a software should be swift, effective and low-cost.

According to [1] software engineering consists of consistent implementation of the following processes: Software Requirements Analysis Process, Software Architectural Design Process, Software Detailed Design Process, Software Construction Process, Software Integration Process, Software Qualification Testing Process. It means software testing should be performed before software implementation. However, today there are available methods of testing algorithms. That is searching for errors in algorithms before starting

actual coding. It is suggested in work [2]. So, there is a current task effectiveness evaluation of algorithms testing at the stage of software detailed designing compared with software testing at the stage of qualification testing.

To evaluate the effectiveness it is necessary to analyse the definition of effectiveness itself. The effectiveness itself is an abstract notion so we will base upon the criteria of the cost of finding and correcting the mistake.

The feature of a mistake in software is the increase of the cost of finding and correcting the mistake on every next stage of life cycle of software [2]. This can lead to fundamental overrun of IT-project cost. For instance the cost of finding the mistake incomperison with the stages of analysis and designing increases: at the stage of realisation 4 times more, at the stage of software components testing 5 times more, after software components integration from 150 times more (fig. 1 [3]). The other information about the increase of price to correct mistakes depends on the stage of software lifecycle is addressed in other works [2, 3].

So, the sooner error or defect is detected in software the cheaper it will cost the less resources will be spent on correction. Testing implementation will be particularly important because the stage of computational algorithm implementation in software lifecycle is initial and main. Testing the algorithm right after its development (at the stage of software detailed design) we can get 2 options. The first one is the algorithm doesn't contain errors and we can turn to the next software lifecycle stage (Software Construction Process). The second one is the algorithm contains errors that will be corrected and removed. According to [3] error correction needs 25 times less resources (time, money and so on) than if software were developed in according to a traditional approach.

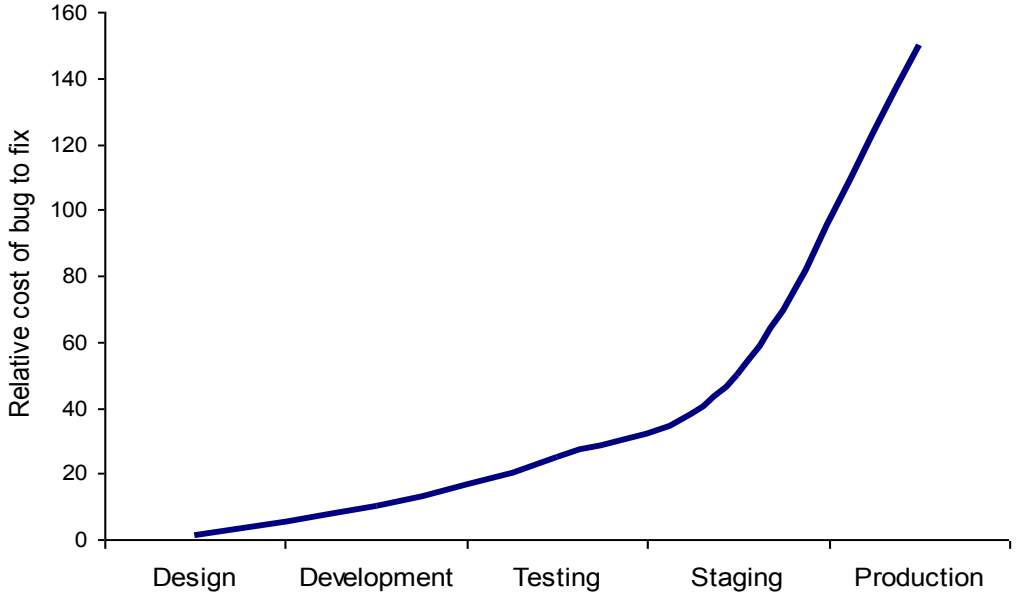

Fig. 1 – Relative cost amount of finding and correcting mistakes in software depends on software lifestyle stage

Gained research results confirm algorithms testing effectiveness at the stage of software detailed design compared with software testing at the stage of qualification testing that is expressed as a 25-fold cost reduction of finding and correcting the mistakes in software.

REFERENCES

1. ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes. – ISO/IEC-IEEE, 2008. – 122 p. – (International Standard).

2. Доренський О. П. Інформаційна технологія автоматизованої перевірки проектних рішень щодо поведінки об’єктів програмного забезпечення / О. П. Доренський // Перспективні напрями наукових досліджень – 2015 : Міжнар. наук.-практ. конф. (Братислава, 17-22 жов. 2015 р.) : матеріали. – К.: Вид-во “Центр навчальної літератури”, 2015. – Т. 2. – С. 207-209.

3. Pomorova O. Intelligent assessment and prediction of software characteristics at the design stage / O. Pomorova, T. Hovorushchenko // American Journal of Software Engineering and Applications. – 2013. – Is. 2(2). – P. 25-31.

МОДЕЛЬ ПОВЕДІНКИ КАМЕРИ ПРИ ПЕРЕМІЩЕННІ КОРИСТУВАЧА В ТРИВИМІРНОМУ ПРОСТОРІ

Єгошкін Д. І., Гук Н. А.

Дніпровський національний університет імені Олеся Гончара, м. Дніпро

Розглядається задача побудови реалістичної моделі поведінки камери при переміщенні спостерігача (гравця) у тривимірному просторі. Камера – точка огляду сцени з певними характеристиками. Під реалістичної поведінкою камери розуміється така поведінка, яку надає спостерігачу уяву, що зображення рухається разом з ним. Камери бувають двох типів – Target (Спрямована) і Free (Вільна) [1]. З аналізу літератури відомо декілька способів адаптації руху камери до руху спостерігача:

1. Ключові кадри – цей вид анімації використовує послідовність заздалегідь розставлених ключових кадрів або позиції точок, створених вручну аніматором. Генерування інших точок проводиться функцією за допомогою морфинга, мікшування, змішування, складання [2,3].

2. Запис руху – даний вид анімації записується за допомогою спеціального обладнання з реальних об'єктів, що рухаються, в тому числі людей і тварин, потім дані зображуються у вигляді хмари рухомих точок. Поширений приклад такої техніки – Motion capture. Даний вид анімації є досить дорогим, оскільки потребує залучення акторів для запису анімації, а також спеціального обладнання [3].

3. Процедурна анімація – повністю або частково розраховується комп'ютером. Даний вид анімації вимагає мінімальну кількість оперативної пам'яті, оскільки анімація представляється у вигляді функції