Trunov O.M., PhD., Prof., Vice rector, ant@kma.mk.ua
Volkova S.O., PhD. St., stitlana.volkova@gmail.com
Mykolayiv State Petro Mohyla University in consortium with “Kyiv-Mohyla academy”
10, 68th Desantnykiv, 54003, Mykolayiv, Ukraine

MEDICAL SYSTEMS: QUALITY AND SAFETY IN ROBOTIC SURGERY

In computer-controlled products, such as modern robotic medical systems and complexes, both hardware and software, should meet not only its functional parameters. The clear requirements to a system’s cost, time and also performance are imposed during its development. More specifically, in such products the quality of software is as important as that of the hardware. For example, in 1955 the software cost accounted for approximately 18% of the total development cost of a software product; in 1985 it increased to 82%. This reversal has further increased the importance of medical software development in accordance to required level of quality and necessary rate of safety [1].

It is common knowledge that improper work of medical software can have unforeseeable consequences, which are related to the state of human health. That’s why medical systems are instantly related to the critical systems. Let’s formulate the main definitions [2]: a life-critical system or safety-critical system is a system whose failure or malfunction may result in: death or serious injury to people or loss or severe damage to equipment or environmental harm. The classification for critical systems is presented on Fig.1.

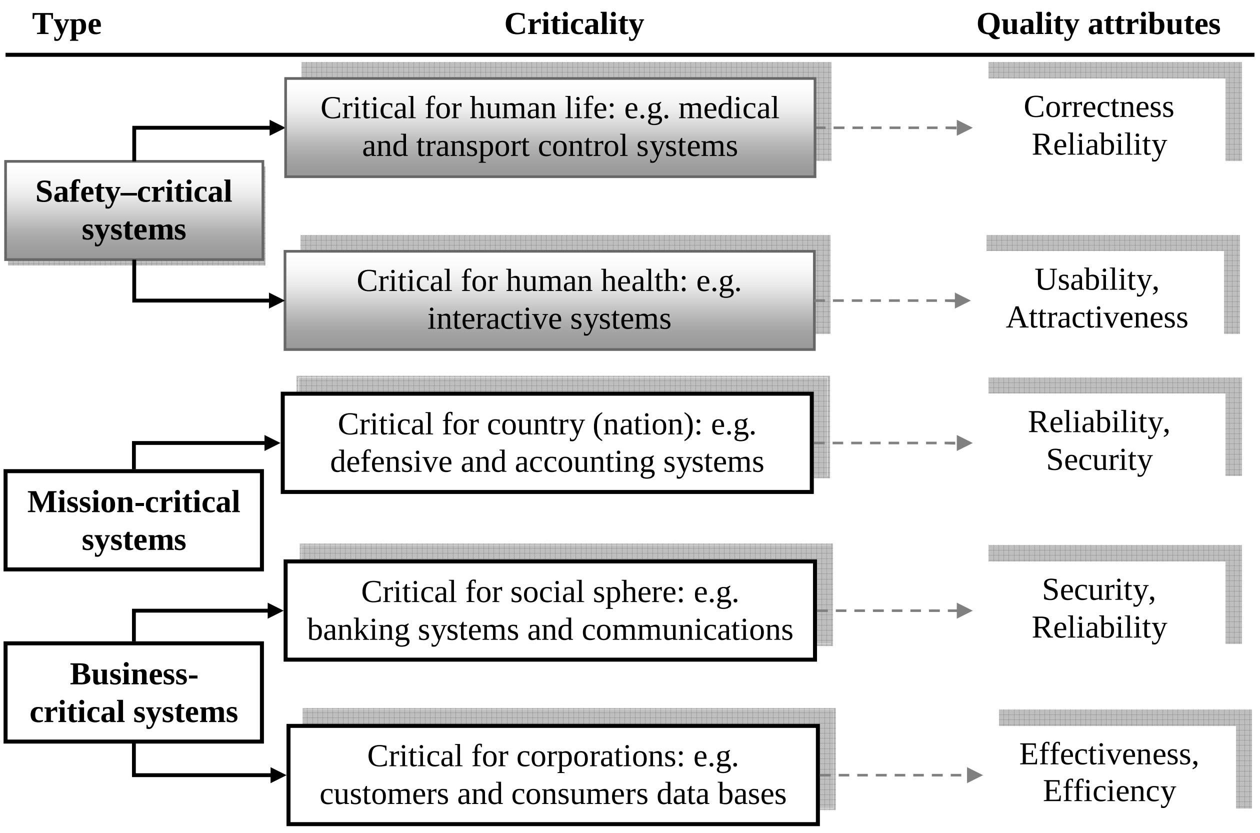
Type
Criticality
Quality attributes
Safety–critical systems
Mission-critical systems
Business-critical systems
Critical for human life: e.g. medical and transport control systems
Critical for human health: e.g. interactive systems
Critical for country (nation): e.g. defensive and accounting systems
Critical for social sphere: e.g. banking systems and communications
Critical for corporations: e.g. customers and consumers data bases
Correctness Reliability
Usability, Attractiveness
Reliability, Security
Security, Reliability
Effectiveness, Efficiency

Fig.1. The general classification for critical systems

Therefore the definition of well-known facts about medical systems and equipment failures is discussed in [1].

- Two patients died and a third was severely injured due to software errors in a computer-controlled therapeutic radiation machine called Therac 25. The patients received approximately 125 times the normal doze of 200 rads.
- Two people died in incident involving heart pacemakers because of software errors.
- A newly installed computer-aided dispatch system for the London Ambulance Service failed because of software faults, resulting in the wrong ambulance being sent to an incident.

- An infusion pump delivered the maximum rate instead of intended rate subsequent entering certain valid data because of a software error.
- A pacemaker lost stored data when any of the parameters were altered.
- An artificial intelligence medical system provided the wrong advice, leading to a drug overdose.
- A diagnostic laboratory instrument containing a software error resulted in incorrect reports of patient data.
- A medical device wrongly processed negative numbers, causing a failure with specific data.
- A multiple-patient monitoring system failed to store collected data with the correct patient due to a software fault.

A subsequent investigation identified many causes of these incidents, including unjustifiable confidence in software, poor software engineering practices and procedures, and removal of independent hardware safety features. Thus, a single failure in the software caused an accident and unrealistic and unfinished risk assessment [3].

In accordance to the [4], the distribution of medical systems devices and software errors by different medical areas is described on Fig.2.

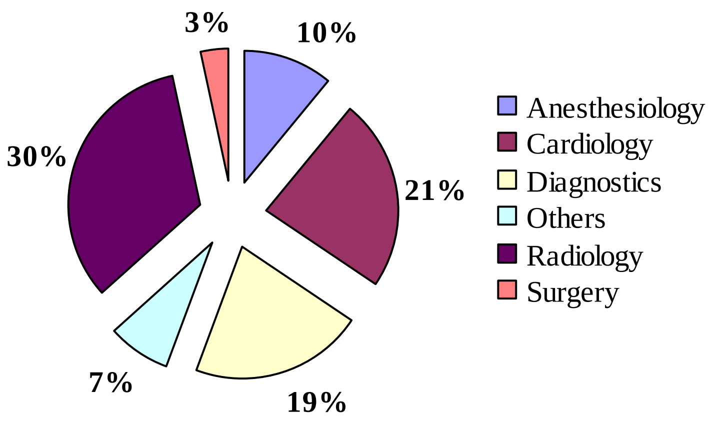

Fig.2. Distribution of medical systems errors by different medical spheres

The medical device's faults, which include software defects, are described on Fig.3. The diagram shows that total number of medical software recalls from 1983-1997 is 383. The years 1994, 1995, 1996 have 11%, 10%, and 9% of the software recalls. One possibility for this higher percentage in later years may be the rapid increase of software in medical devices. The amount of software in general consumer products is doubling every two to three years [4].

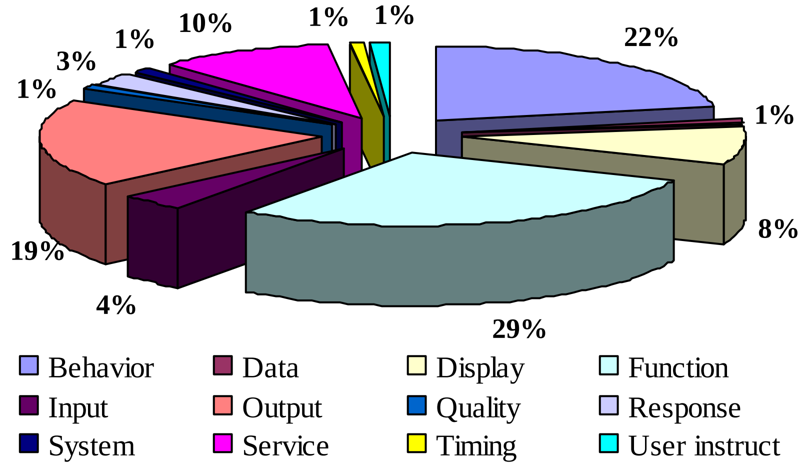

Fig.3. The medical system's failures distribution by symptoms

Also, it is necessary to admit the importance of software quality and reliability in robotic surgery, especially in accordance to its fast development. Three major advances aided by surgical robots have been: remote surgery; minimally invasive surgery; unmanned surgery.

Major potential advantages of robotic surgery are precision and miniaturization. Further advantages are articulation beyond normal manipulation and three-dimensional magnification [5, 6].

Let's discuss minimally invasive surgery in details. Minimally invasive surgical procedures avoid open invasive surgery in favor of closed or local surgery with fewer traumas [7]. These procedures involve use of laparoscopic devices and remote-control manipulation of instruments with indirect observation of the surgical field through an endoscope or similar device, and are carried out through the skin or through a body cavity or anatomical opening [8]. Special medical equipment may be used, such as:

- Fiber optic cables;
- Miniature video cameras;
- Special surgical instruments handled via tubes inserted into the body through small openings in its surface.

The images of the interior of the body are transmitted to an external video monitor and the surgeon has the possibility of making a diagnosis, visually identifying internal features and acting surgically on them. However, the safety and effectiveness of each procedure must be demonstrated with randomized controlled trials.

Let's review the example of some occasions with medical robotic surgery system [9]. Surgery robots can break down in different ways. They can have hardware failures, e.g. broken arms. They can have software failures and the robot is made to stop working and will not allow you to move any instruments. From 299 robotic operations, it was happened several problems, such as: 1 hardware failure, 2 software failures, 2 other operations were affected by robotic failures that were discovered before the operation was started, 1 case was cancelled and 1 was delayed until our robot was fixed.

Thus, **actual problem** consists in not only creation of new medical software, but in the creation of the medical system with required quality, reliability and safety. Moreover, there are many researches in the area of medical software quality, reliability and safety [1-4, 13-15]. Nevertheless, let's discuss main definitions in accordance to international standards, and let's formulate typical design methods of software engineering for providing essential level of quality, reliability and safety.

Medical software engineering for life-critical systems is particularly difficult, partly because of main software defects mostly appeared on the specification phase (Fig.3). After all phases of the software development these early injected defects will cause the serious software faults [10].

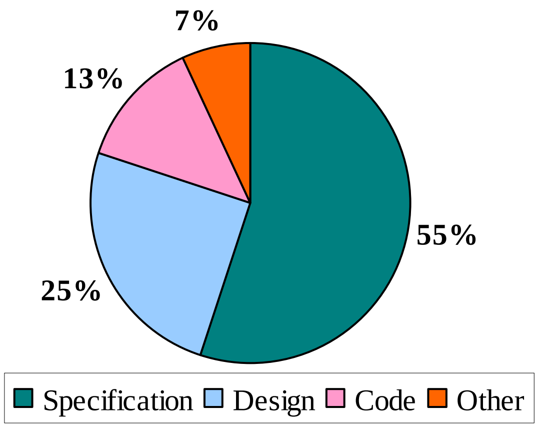

Fig.4. The medical system's failures distribution by the life cycle phase

In accordance with ISO 9126 [11], the management of software quality is based on the control of current quality level of software and corrections of software development processes, deployment and support with the purpose of adding properties which provide satisfaction of customer's and end user's requirements. The intensive development of normative base in the area of estimation and providing of software quality is testified to actuality the decision of this task. It is necessary to consider such circumstance as the **process of providing the quality** which is selected separately, as one of the organizational processes, in the international standard of ISO 12207 [12] that regulates the software life cycle (LC).

That's why, it is essential to discuss the **basic approaches for providing of required level of medical software's quality**.

- First approach is to carefully code, inspect, document, test, verify and analyze the system.
- Second approach is to certify a production system, a compiler, and then generate the system's code from specifications.
- Third approach uses formal methods to generate proofs that the code meets requirements.

The **estimation of software reliability for medical diagnostic systems** based on modern approaches and techniques can be considered through:

- Inspection methods which assume verification of software in accordance to the normative document's requirements by the informal analysis of documentation and development processes;
- Use of the specific metrics which allow to evaluate the level of software quality and reliability based on probability measured analysis of software behaviors;
- Application of mathematical models for the estimation of probabilistic indexes of the reliability.

Medical software safety is a conservation of human life and its effectiveness and the prevention of damage to items as per operational requirements. Hence, risks of medical sort are usually managed with the methods and tools of safety engineering [13]. **Typical design methods** include:

- The probabilistic risk assessment.
- The method that combines failure modes and effects analyses with fault tree analysis [14].

All of these approaches improve the software quality in safety-critical systems by testing or eliminating manual steps in the development process [15], because people make mistakes, and these mistakes are the most common cause of potential life-threatening errors.

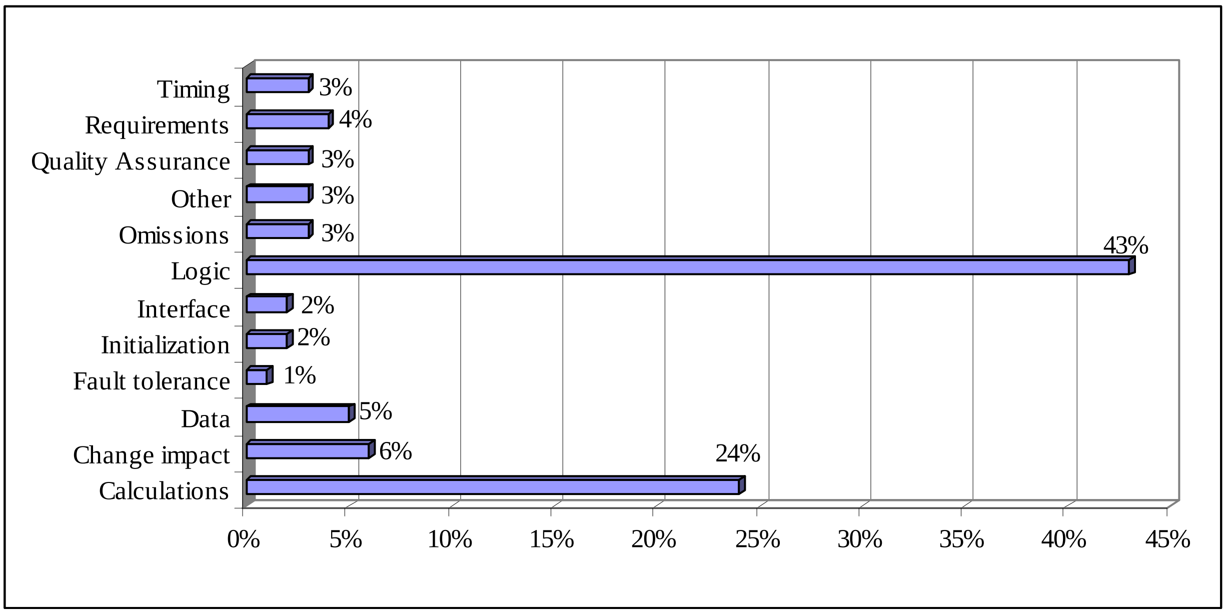

Fig.5. Fault class distribution

For providing an essential level of software quality many medical companies implement software quality assurance programs to ensure that software product and software development processes conform to specified requirements.

The conception of providing the quality of the medical systems consists of such steps:

- ***Collect requirements*** → ***Develop a plan*** → ***Establish mission statement of SQA*** → ***Establish a policy and standard*** → ***Highlight relevant activities*** → ***Develop operating procedures*** → ***Train and promote the program*** → ***Review program for improvements.***

The first step involves gathering all information related to the program, including internal and external requirements, organizational culture and industry standards. The second step establishes a plan that includes factors such as important activities to be accomplished, required resources, and a schedule of expected completion dates. The third step establishes a mission statement software quality assurance that is traceable to the company's mission statement and approved by the upper-level management. The fourth step develops a policy and a standard. The policy focuses on the important organizational concerns, such as software quality assurance having the authority to perform independent evaluations of software development processes and products, and the authority to take appropriate corrective measures to improve the quality of software products and development processes. The standard provides description supporting the policy. The fifth step identifies specific activities. This includes analysis of the existing activities (if any) being performed, and the identification of the most valuable and additional activities necessary to satisfy the mission statement.

Consequently, the support of required medical system’s quality and safety, especial in robotic surgery, is essential and can be provided with contemporary methods and tools of medical software quality and safety engineering. The conception of providing the quality, reliability and safety of the medical systems consists of:

- Evaluation of system quality, reliability and safety on all stages of software life cycle;
- Realization of specific tools for estimation of the software reliability and quality;
- Application of the modern progressive techniques during software development and testing, for example Test Driven Development (TDD).

References

1. Dhillon, Balbir S., Medical Device Reliability and Associated Areas, CRC Press, Boca. Raton, Florida, 2000, 240p.
2. http://shemesh.larc.nasa.gov/fm/fm-why-def-life-critical.html
3. Therac-25 Accidents - http://sunnyday.mit.edu/papers/therac.pdf
4. Dolores R Wallage, D. Richard Kuhn, Failure modes in medical device software: an analysis of 15 years of recall data - http://csrc.nist.gov/staff/Kuhn/final-rqse.pdf
5. Rahimi, M, Xiadong, X (1991), "Framework for software safety verification of industrial robot operations", Computers & Industrial Engineering, Vol. 20 No.2, pp.279-87.
6. Dowler, N.J. (1995), "Applying software dependability principles to medical robotics", Computing & Control Engineering Journal, Vol. 6 No.5, pp.222-5.
7. http://www.thieme.de/fz/min/ - Journal Minimally Invasive Neurosurgery
8. Berlinger NT: Robotic Surgery - Squeezing into Tight Places. New England Journal of Medicine 354:2099-2101, 2006
9. http://www.njurology.com/RoboticSurgeryBlog/robotic_surgery_basics/
10. Cem Kaner, James Bach, & Bret Pettichord, *Lessons Learned in Software Testing*, John Wiley & Sons, 2002.
11. ISO/IEC 9126-1. Software engineering – Product quality – Part 1: Quality model, 2001. – 26 p.; ISO/IEC TR 9126-2. Software engineering – Product quality – Part 2: External metrics, 2003. – 86 p.; ISO/IEC TR 9126-3. Software engineering – Product quality – Part 3: Internal metrics, 2003. – 66 p.; ISO/IEC TR 9126-4. Software engineering – Product quality – Part 4: Quality in use metrics, 2004. – 70 p.
12. ISO/IEC 12207. Information technology -- Software life cycle processes
13. .IEC 61025-90 Fault Tree Analysis – Chicago: International Electrotechnical Commission, 1990. - 41p.
14. Beerthuizen, P.G, Kruidhof, W. (2001), "System and software safety analysis for the ERA control computer", Reliability Engineering & System Safety, Vol. 71 No.3, pp.285-97.
15. W. Gibbs, “Software’s Chronic Crisis,” Sci. Am. (Int.Ed.) 271, 3 (sept.1994), 72-81.
16. Capability Maturity Model / M.C. Paulk, B. Curtis, M.B. Chrissis, Ch. V. Weber // IEEE Software. – 1993. – Vol. 10, №4. – P. 18–27.