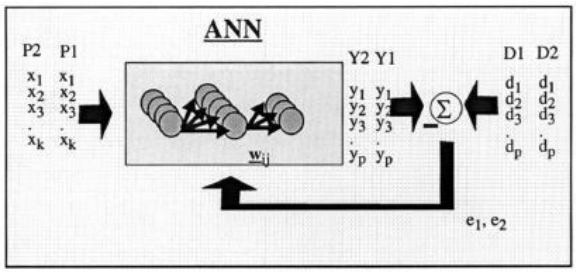
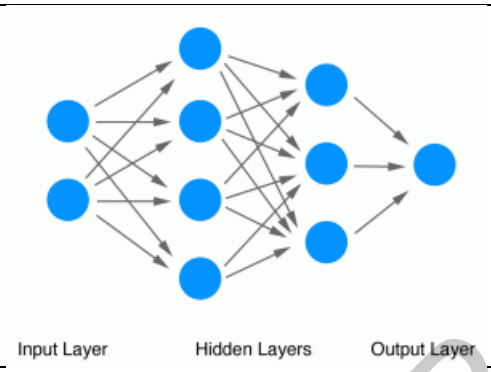


Picture 1 – The style of a neural computation



Picture 2 – Simple artificial neural network topology

Neural networks are good at providing very fast, very close approximations of the correct answer. Their applications can be categorized into classification, recognition and identification, assessment, monitoring and control, forecasting and prediction. Among the tasks for which they are well suited are handwriting recognition, foreign language translation, process control, financial forecasting, medical data interpretation, artificial intelligence research and parallel processing implementations of conventional processing tasks.

One drawback for using artificial neural networks, particularly in robotics, is that they require a large diversity of training for real-world operation. To implement large and effective software neural networks, much processing and storage resources need to be committed. Simulating even the most simplified form of Von Neumann technology may compel an engineer to fill many millions of database rows for its connections – which can lead to excessive RAM and HD necessities. Furthermore, the engineer of such systems will often need to simulate the transmission of signals through many of these connections and their associated neurons – which must often be matched with incredible amounts of CPU processing power and time.

The explanations of biological and artificial nature of neural networks, examples of training and using of neural networks were given. Also advantages and disadvantages of such systems were examined.

It's evident fact that artificial neural networks are very perspective and modern technologies which will be developing in future.

Bibliography:

1. Gurney, K. An Introduction to Neural Networks. – London: Routledge, 1997.
2. Haykin, S. Neural Networks: A Comprehensive Foundation. – Prentice Hall, 1999.
3. Lawrence, J. Introduction to Neural Networks. – California Scientific Software Press, 1994.
4. Artificial neural network. (April, 2012). Retrieved from http://en.wikipedia.org/wiki/Artificial_neural_network.

NANOMEDICINE

Belarusian State University of Informatics and Radioelectronics
Minsk, The Republic of Belarus

Skripelyova Anastasia

Lazarenko A. M. - higher lecturer

The problem of nanomedicineis studied in this paper. It's the most promising brunch of science with a wide range of opportunities and If the nanoconcept holds together, it could be the groundwork for a new industrial revolution.

Nanomedicine may be defined as the monitoring, repair, construction and control of human biological systems at the molecular level, using engineered nanodevices and nanostructures. It is used for the diagnosis, prevention and treatment of disease and to gain to increased understanding of complex underlying disease mechanism.

Achievement and future prospects for nanomedicine:

- 1st generation product (2000): dispersed and contact nanostructure (Ex-:colloids), product incorporating nanostructure (Ex-:Polymer, nanostructured metal);
- 2nd active nanostructure(2000-2005) : bio-active, health effect (Ex-:targeted drugs, biodevices), physico chemical active adaptive structure (Ex-:amplifier, actuators);
- 3rdnanosystem(2005-2010) : guided assembling (Ex-:robotics, evolutionary biosystems);
- 4th molecular nanosystems (2010-2020): Ex-: molecular devices 'by design'.

Nanomedicine has a limited number of current applications. Current research and development efforts are concentrated in six primary categories: antimicrobial properties, biopharmaceutics, implantable materials, implantable devices, diagnostic tools.

Biopharmaceutics can be divided into two main areas: drug delivery and drug discovery. It main task is to find and to create a new way to deliver drugs into the body. Nano and micro technologies are part of the latest

advanced solutions and new paradigms for decreasing the discovery and development times for new drugs, and potentially reducing the development costs.

Implantable materials: nanotechnology brings a variety of new high surface area biocompatible nanomaterials and coatings to increase the adhesion, durability and lifespan of implants. Nanostructures are being researched for the preparation and improvement of tissue regeneration scaffolds.

Implantable devices: micro and nanosized sensors can make use of a wide range of technologies that most effectively detect a targeted chemical or physical property. Implantable sensors can also work with a series of medical devices that administer treatment automatically if required.

Diagnostic tools are based on two areas: genetic testing and imaging. As example we can consider such devices as nanoparticle probes. Nanoparticles with a magnetic core are attached to a cancer antibody that attracts cancer cells. The nanoparticles are also linked with a dye which is highly visible on an MRI. When these nanoprobes latch onto cancer cells they can be detected on the MRI. The cancer cells can then be destroyed by laser or low dosage killing agents that attack only the diseased cells.

A numerous novel nanomedicine-related application are under development or nearing commercialization. New nanotechnologies may offer the only hope for systematic, affordable, and long term improvements to the health status of our population. This is because nano therapies could, in the long run, be much more economical, effective and safe and could greatly reduce the cost of or substantially eliminate current medical procedures. So, nanomedicine is future medicine.

List of references:
1. Robert A. Freitas Jr. Nanomedicine- Georgetown, 1999;
2. http://www.foresight.org/Nanomedicine/

AGILE SOFTWARE DEVELOPMENT MANAGEMENT

Belarusian State University of Informatics and Radioelectronics
Minsk, The Republic of Belarus

Turach D. V.

Lazarenko A. M. – senior lecturer

Computer science is a young science. Computer programmers my age were trained by engineers. That training dictated how we approached software development for an entire generation. But now after decades of building software to be expensive, unwanted, and unreliable we have come to realize software is different. Building software is more like creating a work of art, it requires creativity in design and ample craftsmanship to complete. Software remains malleable, often illogical, and incomplete forever. Agile software development is based on fundamental changes to what we considered essential to software development ten years ago.

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle.

Everyone realize the way a team works together is far more important than any process. While a new process can easily improve team productivity by a fraction, enabling team to work effectively as a cohesive unit can improve productivity by several times. Of course to be eligible for such a big improvement you must be working at a fraction of your potential now. Unfortunately, it isn't that uncommon.

The most brilliant programmers alive working competitively in an ego-rich environment can’t get as much done as ordinary programmers working cooperatively as a self disciplined and self-organizing team. Therefore, everyone need a process where team empowerment and collaboration thrive to reach your full potential.

The first change is making the customer, the one who funds the software development, a valuable and essential team member. When the dead line gets close a traditional approach to reducing scope is to let the developers decide what will work properly and what won't. Instead let the customer make scope decisions a little at a time throughout the project.

When customer, or domain expert works directly with the development team everyone learns something new about the problem. True domain expertise and experience is essential to finding a simple, elegant, correct solution. A document can have plenty of information, but real knowledge is hard to put on paper. Left alone programmers must assume they know everything they need. When asking questions is difficult or slow the knowledge gap grows. The system will get built, but it won't solve the problem like one guided by an expert on a daily basis.

Perhaps the biggest problem with software development is changing requirements. Agile processes accept the reality of change versus the hunt for complete, rigid specifications. There are domains where requirements can't change, but most projects have changing requirements. For most projects readily accepting changes can actually cost less than ensuring requirements will never change.

Agile can produce working software starting with the first week of development so why not show it to the customer? Agile can learn so much more about the project requirements in the context of a working system. The changes team get this way are usually the most important to implement.