IBMs initiative is made for a purpose of making computers train and maintain themselves, similar to our autonomic nervous system. To create system with functions that we don’t need to take care of. Analogy for heart, lungs, immune system…

IBM has singled out 4 properties that are needed for autonomy: self-configuration, self-optimization, self-healing and self-protection.

-self-configuration (you can add devices, autonomic insert itself in a right way, can start IP addresses but also service discovery)

-self-healing (if there is a problem with one component it would self-heal or it could switch to additional equipment, migrate to another machine)

-self-protection (in case of some attack, it could self-protect, take measures)

There are many more self-aware, self-learning, operate in heterogeneous computing environment, anticipate and adapt to user needs. There are also different levels of quality that is not only reactive that adapts to something new but can also anticipate what is coming.

-Term babysitting in today’s case means that for some systems to be able to run over some extended period of time humans have to take care of it(be in the control loop). They need to look over the systems, to look what the system is doing, to think about what to do and act to correct what could go wrong. Without humans in the loop things would break down.

In one point IBM started being Service Company Instead Of Hardware Company. And it started to be bigger expense, so the goal was to get people out of the loop, or as it is nicely put, to get people on the loop. Use less people but still use some people.

The second level of the upgrade to Autonomic systems is to get engineers out of the loop, meaning that the systems could construct the new solutions and eventually inventing new algorithms trough clever way of learning.

-Grand Challenge of building computing systems that regulate themselves. In simple terms put, new hardware and software is much cheaper to produce than install to the systems. For example for one dollar spent on equipment, you need 10 dollars to spend on management. Hardware is cheap today, but keeping the hardware running is the main cost.

-Automatic means good preprogramed system that works well until everything goes as planned in program. As soon as something goes the wrong way, the human intervention is needed.

Autonomic on the other hand is a system that can self-regulate itself in order to get a solution to given problem. And for that it doesn’t need human’s intervention.

Regression does testing of the new upgrade of the system to be sure that the addition improves and not regress the system (win, Linux update).

Example with SW upgrades Autonomic SW update would deploy the SW, run it, test it and in case of any problems it could identify the component that is causing problems, isolate it revert the whole installation and re-start it without the part that is causing the problem.

**Self-configuration**

In every system configuration there are administrative or personal directives that are like guidelines for configuring the system. Underneath of that the system does the right way of deploying, installing and configuring. For example in some company the games are banned on all computers in HR and as such they are not being included in update. Field workers laptops do not need database information’s of employees because they are customer oriented, etc.

Self-configuration is needed in network configurations (configuring addresses, routes etc.). In parallel computing it is needed for resource assigning. In complex systems it is necessary to auto configure system depending on current environment. It is not possible for human to predict all the possible scenarios that might happen because it is too complex.

According to IBM autonomic system is able to (re-) configure itself according to high-level policies. Autonomic system is self –configuring, but self-configuring system is not necessarily autonomic.

**Self-optimization**

It can be done in Design time, changing the software architecture for best performance, in implementation time by the choice of algorithms, optimizing compiler etc.

In computing center there is a lot of things that you can optimize. So much that it is getting harder and harder for people to understand it. For example, compilers have so many settings that it is sometimes impossible for a human to realize the best optimization pattern. IBM has come up with research paper about an algorithm that gets best optimization settings. Compiler optimization is important for business. For example Intel needs the best optimized compiler settings to show the best performance of its new chip.

Runtime optimization is for example done with TCP that is optimizing the bandwidth, and limiting it if necessary. It can adapt, but it is only parameter tuning. It is still on the level of adaptation that is for now considered advanced, but it is still another step to go to evolution inventing new solution.

Optimization is not always linear process. It can often depend of different parameters. For example WLAN, Ethernet, GPRS - cost vs. speed optimization.

**Self-healing**

Self-healing is today mostly reactive. That means that at the point when something is already gone wrong, we activate the routine that is trying to fix it. Proactive approach is what we are trying to achieve. Then we could predict failures.

Solving some problem is done in couple of steps. First we need to find a cause of the failure, then to find a cure witch we then need to test and possibly deploy. IBM definition the system automatically detects, and repairs localized software and hardware problems. It is distributed approach where decision taking is done locally. The idea is that all the elements doing that will produce system that globally works correctly.

In computer systems the healing is mostly based on log analysis, searching through the log files by looking for some alarm message or some strange pattern form. After that the action takes place.

Job control language - job failures are analyzed, and job is automatically restarted.

Database might have self-diagnosing element that checks if index file is corrupted or re-computing the index file is needed.

Hardware might have some signaling in case of overheating or power outage so it can send safety message.

IBM approach is based on adding the self-healing features instead of implementing them from the start. Self-healing inside the application could be much more efficient by being able to choose algorithms form the start or change data structure.

Still it is just choosing between parameters, and the main goal is integrity, the peace of code that understands his goal and has freedom to choose how to achieve that goal. It could go through algorithm space. It has the ways of solving the problem but it can also invent the new way. It means that code can change, not only the parameters.

**Self-protection**

According to IBM, self-protecting systems are the systems that can defend against malicious attacks, cascading failures and can prevent system wide failures.

It is a very difficult task to achieve, because the attacks are not known in advance. It is also very hard to know what good self-protecting system is because there are not tests that show it. The main idea is that the infected part of the system be isolated without major consequences for whole system.

Nowadays network monitoring system registers and disconnects suspicious computers, semi or rarely fully automatic. Systems can also check if the client machines have latest patches and in that way prevent known attacks.

IBM's proposed methodology says that the whole system needs to be broken into smaller but still autonomic entities each being dedicated to fulfill specific task and by that reduces the complexity of designing a large system. To ensure an autonomic global functioning of all the elements, autonomic manager is needed. Manager is working on principle of CONTROL LOOP that checks the state in each element and acts accordingly. Manager needs to have SENSOR that collects data or states of the element. MONITOR collects and filters data from multiple sensors. Then the data is analyzed and compared to preset policies and goals. After that there is a need to determine if corrective actions need to be performed. Then the action plan needs to be executed, so the managed element can be re-configured. In the complex system there are a lot of managers (for database, server, etc.). Its configuration depends of engineer.

**Policies**

Autonomic system by itself doesn’t know what desired behavior is. Through the policies, we can instruct it on the low level with a goal to achieve desired type of behavior of the whole system. Price of wrong set of policies can be great, because it leads to behavior that we don’t want.

Plan is IBMs component of policies. It re-configures system behavior based on analysis of current behavior. TCP and congestion control are the good example of that.

Analysis is the process where we look for some change in behavior according to the given set of policies. Policies focus our attention to analyze right things in the system.

Knowledge base has some information’s of specific instances of our system. Goal of it is to make analyzing easier and system more efficient. It is some kind of history of the system. And like us it tries to learn from it.

**Assessment**

At the end we can say that IBMs initiative has made great theoretical base for development of autonomic systems. It doesn’t provide concrete solutions, but it points us in the right way of thinking. It is a kind of template for the future.

**Evolution**

Conditional expressions approach is mainly reactive. It has predefined responses to some kind of behavior.

Online algorithms approach considers the probability of future events. It measures system state and reacts accordingly. In that way it is trying to avoid over-reactions that usually cause grater problem that the problem itself.

Generic or parameterized algorithms have set of predefined algorithms that are used in specific occasions. Parameters of algorithms can be tuned to get desired behavior.

Algorithm selection is taking it to next level by allowing choice of the best algorithm form the set of possible solutions that is given.

Evolution implements learning in the system. It allows inventing new algorithms for some case where current ones are not sufficient.

Last point of the lecture is that software is getting old with time as much as hardware. It is important to have software that lasts as long as human lifetime or maybe more. In that way software needs to be adaptive enough to extend its life span without human intervention. Who knows if the photos that we keep today are going to be usable in future.