Hans van Leunen, 2000-06-01

Run with the hare and hunt with the dogs

Merging contradicting concepts

Merging two concepts that at first sight are completely contradictive may offer great benefits. This will be the case when each of these concepts offers its own advantages that cannot be found in the other concept. An open technology seems to be the complete converse of a closed technology in which as much as is possible is guarded. It is shown here that these technologies can be merged with great potential rewards.

Open technology

Here the concept of open technology is taken as a mechanism, for which all interested parties are given as much as is possible full access to all aspects of the technology that supports this mechanism. If everybody has full access, then the chance is large that everybody uses his ability to add adaptations or to actuate modifications. This will quickly lead to complete disorder. For that reason an open technology cannot exist without a scrupulous and clear change control. This task is usually delegated to a standards institution or to a change control board.

Potential contributors may shrank from offering unique intelligent property to an open technology community. The only control that is left over the contributed knowledge works via the membership of the community. So, the only advantage that can be derived from the contribution is the expertise that the contributor has with respect to competing members. However, this consideration neglects the fact that the contribution may facilitate functionality that otherwise would not become available.

The biggest advantage of an open technology is the visibility of all aspects of the technology. This visibility can cause a considerable trust in the technology and in its products. Both the strengths and the weaknesses will quickly become apparent. Any failing aspect will rapidly be detected and will be taken up by a large group of stakeholders. The repair will be performed in a speedy and well-coordinated action. Usually this coordinated action occurs much better and faster than a single stakeholder could ever perform.

Another advantage of open technologies is the high degree of reusability that is stimulated by this approach. The ease of access of all aspects of the open technology plays an important role in the stimulation of reuse.

An open set of services can be configured in much finer detail than a precooked set of packaged services that together cover the same or an even broader overall service. This feature becomes important in resource-restricted applications.

In an open technology, the income of parties is raised in a different way than in a protected technology. The income cannot be obtained by selling the open products. Instead, the income can be obtained from services, such as documentation, education, consultancy, distribution of products and from configuration services. Income can also be raised from tools that implement these services. In an open technology, the costs occur at different places than in a protected technology. However, in an open technology the cost and the income are distributed in a more honest way. Power has less influence. Repression gets no chance.

Closed technology

In a closed and protected solution it is much easier to guard unique intelligent property against unwanted theft. In this way it becomes possible to bring this asset to financial fruition, without losing the knowledge to an open community. The alternative is a continuous and in-depth technological renewal, such that an uninterrupted flow of offerings can be maintained. Only few companies are capable of sustaining that approach for a significant period. That’s why there exists a strong trend towards shielding of intellectual property such that it can be exploited to retain a technological leadership for a significant amount of time.

An attendant advantage of a closed solution is the fact that the integrity of the offering can be guaranteed much better. The internal state of the item is well guarded and cannot be affected by the outside world in an uncontrollable way. This enables the construction of configurations that are better manageable than constructs that are put together from modules that are open to the elements. Consequently, well-encapsulated components may gather a continuously increasing degree of trust when they are applied in different circumstances. This is a different way of trust build-up than the way trust is gathered in open technologies. Trusted components are of great value in the construction of reliable systems and can be very profitable to their supplier.

Every closed item always has a public side. Via this interface, it can be controlled and will it pursue its actions on its environment. Another part of the public side is the way the item is presented and promoted to the public. The environment contains other components and an infrastructure that enables the contributing components to communicate and to cooperate. If a component must live in a given environment, then it must conform to at least a subset of the communication protocols that are resident in that environment.

One of the biggest advantages of closed solutions is, that in this way unique domain knowledge can be encapsulated and offered to users that are no experts in that domain. The shield and the interface take care of the fact that the component can be controlled and used without expert domain knowledge.

Merging open and closed technologies

Closed solutions can be merged fruitfully with an infrastructure that is supported by an open technology. Prerequisite is the fact that the closed component must conform to the requirements of the communication protocols and the relation management of the open infrastructure. In this way, the open technology delivers the environment in which the closed solutions can live and cooperate.

This symbiosis will flourish only when proper agreements about the used communication protocols and the relation management have been established. In order to reduce complexity and to increase efficiency the number of possible choices for the protocols must be reduced to a bare minimum. Preferably, a uniform scaleable communication protocol must be used. It must be scaleable because even the most primitive component that can only implement the lowest level of the protocol, must be able to join in. On the other hand sophisticated components must not be limited in their capabilities. Therefore, they will also implement their part of the higher levels of the communication protocol. However, most of the implementation of the higher levels of the protocol will take place in the supporting infrastructure.

It is possible to accept different types of component models. Currently the best-known component models are Microsoft's COM, Suns's JavaBeans and the CORBA component model. Usually the different models will use different communication protocols. If that is the case, then the infrastructure must provide bridges that cross the gaps between sets of different communication protocols. The requirement of a uniform and scaleable communication protocol becomes prominent for resource-restricted applications. Resource restricted applications must stick to a single component model, or ways must be found, such that bridgeless communication between the used component models becomes possible.

Because every aspect of an open technology is accessible to all interested parties, it is in principle possible to let an intelligent toolkit generate a tailored supporting infrastructure. The toolkit can derive the necessary data from the design and will be able to generate a tailored infrastructure that exactly fits the needs of the chosen components. The advantage of this approach is that the application designer is relieved from the requirement that he needs deep knowledge of the infrastructure of the system. He can stick to his expert domain knowledge and rely on his development environment to add the necessary infrastructure parts. The responsibility for the different system parts are delegated to parties that are best equipped to serve this. The tools must work according to predefined standards. This prevents that the tools or their vendors will become too dominant and via a detour, in fact a closed rather than an open solution is obtained. In case of resource restricted applications, the capability to generate a tailored infrastructure is a crucial requirement.

It is possible to design components according to the principles of the open technology concept. If after a while a product with proven robustness is obtained, the design can be frozen and the product can be

converted to a closed solution. The difference with the normal closed solution is that all aspects of the product have been published. The correspondence is, that the product is treated as a well encapsulated item.

Exchange of information

Open technology is best served with a properly structured and effective exchange of information concerning the availability and the properties of both open and closed items that are related to that technology. The Internet is well suited for this purpose. The information exchange can make use of well-organized repositories that contain information, which is both humanly and mechanically readable. This can be realized by using XML based scripts. The same site can be used to promote and merchandise the available products. An important task of the repositories is the exchange of information on reusable design elements. Important is the capability to exchange the specification of interfaces. When a component designer applies interfaces that are already very popular, then the chance is larger that his target product will become reusable in combination with other components.
The public side of closed solutions can also be published in this way. An intelligent tool may be able to generate skeletons that are derived from the retrieved information. These skeletons can be used in designs and in prototypes in order to check if the component fits into the target configuration. With such technology at hand system design and build times and corresponding costs can be reduced a great deal.

Upshot

In this way, it becomes possible to create an open market for components. The capabilities of the open technology are used to take care of the supporting infrastructure. It is possible to create packages of hardware components and software components that together constitute a consistent service. Such a package represents a hybrid component. The package contains a package manager that at load time and at initialisation time in cooperation with the system relation manager takes care of the establishment of the necessary interrelations. It may also install initial values of other properties of the inserted components.

The infrastructure includes the operating system. If a toolkit generates a tailored supporting infrastructure then it means that as part of that process also a tailored supporting operating system is generated.

When all necessary parts of this technology are in place, then the design and build of systems will be done in one stroke. The project risk will be reduced significantly, because the feasibility of a concept becomes clear in a early stage of the product development cycle, while at the same time the system generation process will be much faster than it currently performs.
Competition will increase because the threshold for joining in the process is much lower. Small firms get a chance to operate in a niche of the market and still survive by implementing their unique knowledge into reusable components. All involved participants can concentrate on their strengths and need not invest in partial solutions that are better served by other parties.

Configuring systems from available components is a better manageable build process than generating the same functionality in the form of a monolith. It will become easier to build larger and more complex systems. Once implemented in a reusable component, new concepts can be spread rather easily in a series of different products. Therefore, innovations will more quickly reach a broader part of the market. Products in the market place will become more sophisticated and will still have reasonable prices. The end-customer will be the one that will benefit the most of this approach.

Linux

Linux is the prominent example of how successful the Open Source approach can be. Linux is a version of UNIX. It has become a realistic threat to existing operating systems, such as Microsoft Windows and other UNIX variants. UNIX exists already for several decades and the underlying concepts are partly superseded. Besides of that, Linux does not serve all purposes. Due to its size and its poor real-time behaviour, it is not suitable for resource restricted real-time applications. Currently Linux does not deliver an infrastructure that supports freely deployable components. However adding

such an infrastructure is not too difficult. Thus, it is possible that soon an infrastructure that has much in common with ActiveX in the MS Windows platform will be available in Linux.

Currently much attention is spent to the deployment of Linux in other applications than purely server, desktop or workstation applications. This has resulted in the generation of several versions of embedded Linux. Again the size of embedded Linux and its poor real-time behaviour may hamper the application in resource restricted applications. Tools that help deconfigure Linux come to the rescue, but do not solve the requirements of applications that are severely resource restricted.

Most resource restricted real-time embedded applications make use of a traditional RTKOS and when required, some extra service layers. Traditional RTKOSs usually are closed solutions. The RTKOS and each of the added service layers come with their own price tag. In practice the design requires a perfectly tailored infrastructure. This poses problems to the traditional RTKOSs because they come in closed packages that cannot be configured to any desired detail. The corresponding licence burden would also offend the RTKOS vendor.

Where traditional RTKOS vendors work via a build-up procedure will the suppliers of Linux work via a deconfiguration service. Linux itself has no price tag. The deconfiguration service must be paid. This is converse to the traditional RTKOS approach.

Embedded Linux is no proper solution in extreme resource restricted applications. As is indicated also the traditional RTKOSs don't bring the ultimate solution. In that case the RTKOSs that are delivered according the Open Source approach may deliver a better solution. Tools may generate an exactly tailored real-time kernel operating system and in addition the exact set of extra services that are needed on top of that kernel. Since the OS does not require licence fees, the licence burden is avoided.

If a tailored supporting infrastructure including the operating system kernel can automatically be generated by an appropriate toolkit and if selected reusable and easily deployable components can fill in the rest of the required functionality, then there is no further need for a traditional operating system. So, one may ask what purpose Linux will serve in solving the software development problems of the future.
Certainly, Linux has played its role as an eye-opener for the capabilities of the Open Source approach. However, Linux does not solve the problems of the software generation process that bother most planners. The world is asking for solutions that help coping with the increasing size and complexity of applications. With that respect, Linux leads into a side track. It empowers software developers that were educated with UNIX in applying that skill. It does not bring any revolutionary technology that would initiate the required paradigm shift.

According to the introduction that is given above, it is better to focus the attention to small RTKOSs that are supported by the Open Source approach and that can automatically be generated by appropriate tools. The development environment can then exploit the resources brought by an open market of reusable components. In that way the limited human resources are spent in the most efficient way.