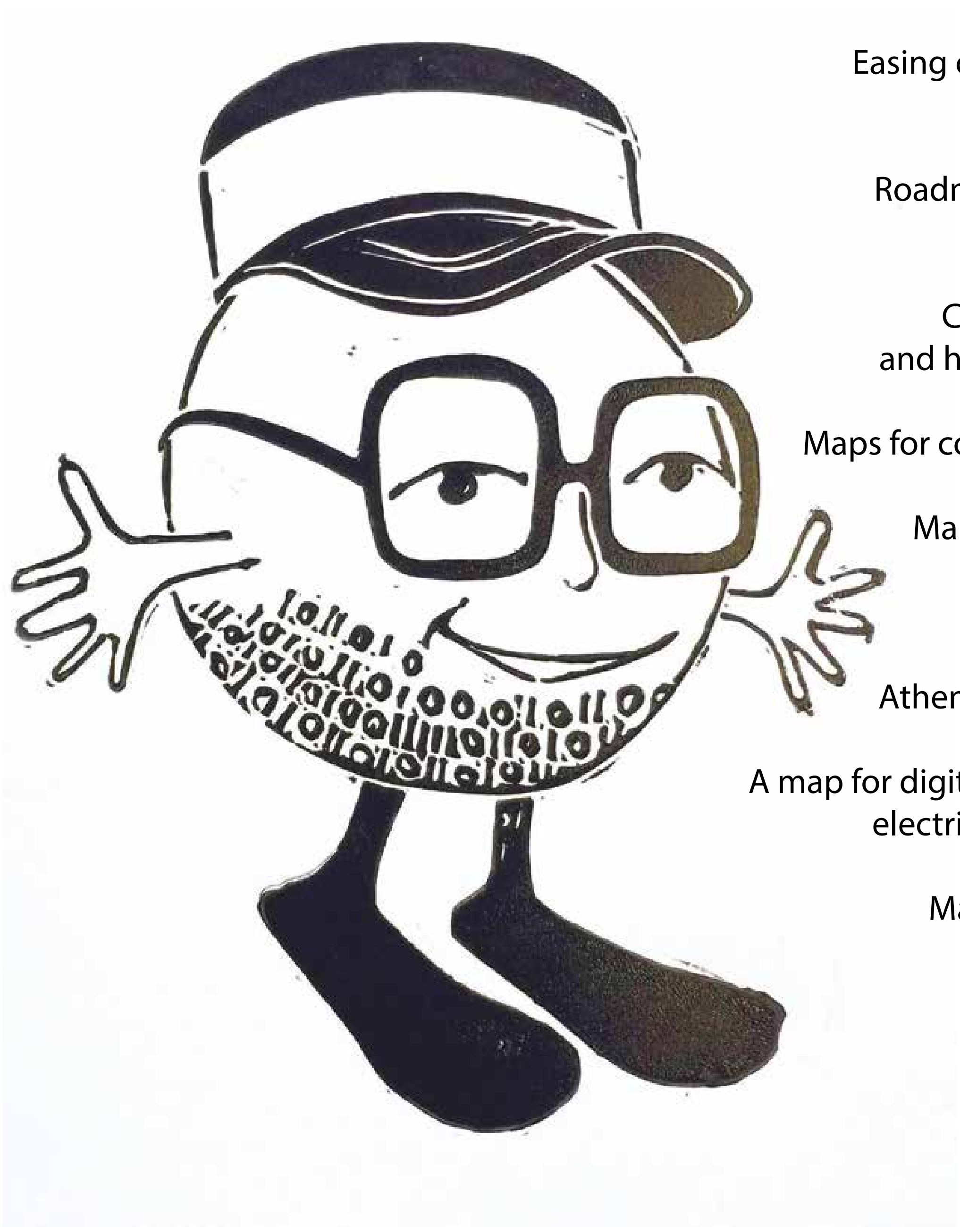
SOFTWARE FOR DOMAIN EXPERTS
helping free thinkers work better

Special Conference Report

Mapmaking for Software Projects

Report from Athens forum in May 2019



Sponsored by:

Ulysses

Maps – and making software easier to understand

Map-making can be a very useful tool in software projects – to make software which is more transparent and maintainable, help co-ordinate all of the people involved, make it easier to do complex tasks, have a plan for development, and ensure you achieve your desired goals. We explored different ways to use maps at our May 2019 Software for Domain Experts forum in Athens

SOFTWARE FOR DOMAIN EXPERTS
helping free thinkers work better

This is a report from the fifth Software for Domain Experts Conference held at The Cube, 8 Kleisovis Str, Athens in May 2019

Report written by Karl Jeffery,
Editor,
Software for Domain Experts
jeffery@d-e-j.com
Tel 44 208 150 5292

Conference chaired by:
Dimitris Lyras

Layout by Laura Jones, Very Vermilion Ltd

Cover art by Alexandra Mckenzie

Software for Domain Experts Ltd.
c/o Future Energy Publishing,
39-41 North Road,
London, N7 9DP, UK
softwarefordomainexperts.com

Some presentations and videos from the event are available on the event web page for free download, www.bit.ly/SFDEAth6

Map making can be a very useful tool in software projects – to help make software which is more transparent and maintainable, and help co-ordinate the people involved, which can be other software developers, other software companies, and customers.

It can make it easier to achieve and verify cybersecurity, because people can understand how the security systems work and what situations they may leave unprotected.

Being able to build software around a map can make it easier to do complex technical tasks, like run software with hardware accelerators on a cloud system.

It can also make it easier to make software which achieves its desired goals – or to say the opposite, to avoid putting really bad software into customers' hands, as the Athens Metro did.

It can give you a plan for development.

We explored a range of different benefits of using maps in software development in our May 2019 Software for Domain Experts forum in Athens.

Maritime maintenance

Dimitris Lyras, director of Lyras Shipping and chair of the conference, talked about the challenges making software to manage maintenance in shipping operations, and how map making can help do it better.

Dimitris Lyras, director of Lyras Shipping and chair of the conference.

Many companies make "asset management software", which has a database of maintenance tasks which need to be done, and provides crew with a schedule for doing it.

But this is only of limited value in real life shipping operations, where machinery can fail unpredictably, and difficult knowledge decisions need to be made about the cause and regarding which work is urgent and which work can be postponed.

The income stream for a shipping company comes from cargo owners, who sign contracts with the ship-owner, including loss of income for delays. So the most important question is whether or not postponing any maintenance work may lead to loss of income.

Answering this question is beyond any asset management system available today, because none of them have any understanding of the relative effect of different malfunctions, he said.

Also much maintenance work involves diagnosis and fault finding, which requires understanding different causes and effects. A maintenance engineer does this modelling in their head. It would be possible to build software which could help with this, for example if it could draw on many years of experience the company has with similar equipment, and explain what went wrong on previous occasions and how it was fixed. But nobody makes this software either, he said.

Furthermore, if engineers do not feel that the software fits with their mental model, they won't use it – and they won't be inclined to work together with any software company or digital project manager unless they can see a way that the software will "persist" or follow their mental model.

Another limitation of typical asset maintenance software is that it uses unique part numbering to identify different parts. Manufacturers exploit this by often selling a part which is identical to another one coming from the same manufacturing source, at a higher the price, because it is the "official spare part," and its part number is in companies' asset management software.

But if software would describe parts by their functionality or specification, rather than only by their part number, it would be possible to use the lower cost but otherwise identical alternative, he said.

A further limitation of asset maintenance software is that it will typically have its database in a rigid structure – so very hard to change over time. It may have been made decades ago.

Small additions are made, such as adding some fields, which can add to the complexity of the software, and get to a situation where no individual person is confident about the effect of the change in the system.

A better approach to building maintenance management software could start with a map of the high level goals, and then you gradually add more granularity to get to the same level of detail as a conventional system. So starting with a goal model and then getting down to the specific model of different spare parts, manufacturers, tasks. The database would be a component of the software but not at its core.

Maritime cybersecurity

Mapmaking could also help in maritime cybersecurity, Mr Lyras said.

There have been a number of high profile hacks of shipping companies (particularly Maersk in 2017). Shipping company IT departments see this and recognise they need to improve security. But they try to do that without getting a specific understanding of the threat landscape, and where their defences are weakest.

A better approach might be to make a map of the company systems especially in the area of sensitive data and critical actions , showing where its 'assets' of interest to a hacker are, and the pathways where a hacker may gain entry, and then assessing whether the measures to control entry are proportionate.

Such a map would identify that certain aspect of the corporate systems have higher criticality to different businesses. For example a container shipping company is heavily on a booking system for its basic income. This means thousands of transactions on a booking system that can be disrupted. A tanker shipping company is hardly reliant on systems for tis income because the transactions are much less numerous and therefore not reliant on a system to aggregate and manage them.

Easing communication between developers and business

Stefanos Christakis, ERP project manager with shipping company Euronav, talked about the challenges of getting software developers and 'business' to see a common picture, or map, of where the software development is going – and how it might be improved.

Stefanos Christakis, ERP Project Manager with shipping company Euronav, said there is often a gap in understanding between the "business" side of a company and the software developers – and suggested ways to improve it.

The normal process of developing software starts with a plan for what you want to build, which sometimes an idea from a business owner, or something you see in the market, he said.

More senior people in a company have a vision of something great they want to do in software, but they are not sure how to implement it, they can't break it down, they can't get into details.

People will often ask for something they've seen on another piece of software, or say how they want it to work.

The software project management team can take this as a starting point, but usually they need to dig into it a lot further to understand what exactly they are being expected to deliver – and explain why certain elements are not possible or would be very hard to make.

Software project managers need to understand the "scenario", the sequence of actions which will happen.

People sometimes make "use case" documents, about how people will work with the software, but these are sometimes considered surplus to requirements.

Stefanos Christakis, ERP Project Manager with shipping company Euronav

The software project team do some research and come up with a design.

One way to avoid the gap is to do mock-ups, so people can see what the software will look like, before it is actually built, so the final product fits with their expectations, or if not, changes can be easily made. The testing process can be done with real company data.

Often after building a mock-up design, you realise there is still a gap between what the software users want and expect, and what the software developers can reasonably deliver, he said.

Or sometimes the opposite can happen, the software users are sceptical about whether it offers any benefit. They might say, I don't mind if they use a phone, fax or software, so long as it gets done.

The software project team can also emphasise how it offers an improvement on the current way of working, in terms of reaching an organisations' goals, he said.

There can also be people with short term requirements, people saying they want to fix something right now. But this might just be putting out fires, rather than improving software quality, and is probably increasing complexity and reducing maintainability.

It does not always end up as a fight between software developers and the rest of the business, but "a lot of times it is," he said. "It is a different way of thinking."

It helps if developers have domain expertise so they understand how the software will be used. Domain expertise is very important in conveying the story to the developer, he said.

You might also need people who see a business problem in different levels, for example someone who is going to do a business task, and someone who understands what it will cost and mean for the company in a wider sense.

Sometimes the most critical people don't get involved because they don't see themselves as stakeholders, since they won't actually use the software themselves, although they do care about the results of it. Or they don't see software as directly relevant to them, just a tool for certain staff members.

It is important that project managers understand the environment. "We need to be critical – don't always accept the first thing they tell you," he said.

Companies everywhere say they are keen to have more digital processes, but they are a long way off achieving it, he said.

A five step process for the development of a technology product – Christos Lytras

Christos Lytras, Managing Partner of innovation consultancy Hippocampus.io, presented a process for how to develop a technical product.

Developing a technical product, which fulfils a market need, a.k.a. has paying customers, is extremely hard to do – the common mistake is for companies to make something which looks like a product but does not actually provide any value to the end user at all.

Christos Lytras, presented a roadmap for how to develop a technical product which delivers value to its customers, and revenue for its developers.

Christos Lytras

A first step is to identify the problem. You are building something to solve a problem. And if the customer is a complex company, there is no unique single problem. You might need to identify the most important one and address that first.

You need to identify how this problem relates to the person who is going to buy or use your product. They may have different perspectives. For example if you are making a computer game for children, which parents would pay for, it needs to solve separate 'problems' both for the parents and the children.

It is not easy to predict how someone perceives their own problem. You should find a group of people with common needs, and ask them if they agree with your solution, he said.

Many technology companies develop solutions for problems which customers do not themselves recognise, such as the need for an Uber type company selling different services (other than car rides).

If you can find ten users which share a common problem, you can go to the third step, which is to develop conceptual solutions, including with different technical approaches or user experience approaches, he said.

The next step is to build a prototype as cost effectively as you can, and show it to your customers and say, would this solve your problem and would you pay for it.

Many times people create something which "looks and smells like a product" but is not something someone is willing to pay for.

This is sometimes called finding a "product market fit."

You have used a number of assumptions about what customers actually want in order to build your prototype– you need to see through interacting with them and the prototype if they actually hold.

It is useful to build software which aims to fulfil one task, or fulfils the important features, not every need the customer has.

Hippocampus.io is an innovation consultancy working with large corporates on developing new products internally and small technology / product companies, in scaling and growing internationally Greece and Central / South Eastern Europe. It was founded in 2016.

Cybersecurity certification – and how mapmaking can help

Dr. Andreas Mitrakas, head of the Data Security & Standardisation Unit, with the European Union Agency for Network and Information Security (ENISA), talked about plans to develop a cybersecurity certification framework in Europe.

For someone to certify that a product or service is cybersecurity, they would need to be able to see the product's 'map' for how that cybersecurity is managed – how access is managed to data and servers which need to be kept secure.

This might involve making a kind of map of how the software system functions, which someone else could understand, and easily see how security is maintained.

The security assessment would be risk based, where companies make a self-assessment of their risks and how they mitigate them.

A risk based approach can mean the security processes change over time, reflecting changing risks or stakeholder requirements. It means that companies can control the assessment criteria so manage the costs of it.

If companies go through this self-assessment process, their reward is to have a different burden of proof in a court. If you are accused of being wrong, it is up to the accuser to prove you are wrong, rather than your burden to prove you are right, he said.

A static cybersecurity requirement could never be effective in today's technology environment, with consumer goods produced at high volume and with a short lifespan, and new advanced technology products being developed all the time.

The process is made more complex by the fact that products can have many layers, and identifying the producer of each layer is sometimes not obvious. Electronics components have different parts made by different companies. So it is very difficult to ask the producer of the product to take complete responsibility for it, as you could for example with foodstuffs or goods without electronics.

Dr. Andreas Mitrakas, head of the Data Security & Standardisation Unit, with the European Union Agency for Network and Information Security (ENISA)

A higher security standard would apply for any products to be used for government or military actions, or for systems used in essential services where security is an issue, such as SCADA based automation systems.

There can be a big business opportunity in an emerging sector of testing, inspection and certification, estimated at Eur 20bn in Europe, he said.

ENISA's role in the European Union is to co-ordinate the work of member states and the EU Commission on cybersecurity, and co-chair the stakeholder groups.

It is setting up working groups with a range of experts, to work out how such a certification framework would work, he said.

Maps for complex software projects

Telecom company NOKIA is involved in highly complex software projects, which can involve hundreds of developers in multiple countries. Creating a map for how they will work together, and how their various components will interact, is essential

NOKIA produces software and hardware products which are used by Communication Service Providers (CSPs) to run their network services. Polizois Kokkonis, senior product manager, Nokia Networks, said he is currently involved in a project to build a cloud distributed database, involving 200 developers in 4 countries, with multiple components that need to interact with each other.

The developers coming from different backgrounds do not always share the same terminology, which may be an obstacle in communication. For example, the understanding of the term "functional test" is different in France than in Greece. It is important to understand what people are saying, so people understand what they should build, or what assumptions are valid.

Polizois Kokkonis, senior product manager, Nokia Networks

Over recent years, the telecom sector has shifted from a using proprietary hardware to more open products and cloud based solutions. The systems have become far more interconnected. End user experience is crucial for the operators.

Software project requirements can change in scale and scope very easily, far more than in for example construction projects, where a project to build a house is unlikely to change into a project to build a hotel.

There are usually constraints in cost, leading to time, you can spend on the programming work, which can lead to decisions about how much time to spend perfecting code. All this highlights the importance of communication between the team members.

One way to address a complex project is to ensure shared mental models and adopt an incremental / evolutionary approach. As an example, all of us can fill a glass of water with a tap without too much thinking. But if we defined what we do in terms of technical specifications – the water flowrate, the capacity of the glass, the level of water in the glass, and when to switch off the tap, we make an intuitive task unnecessarily complex.

In the same sense, many technical discussions get more technical than they need to be, by focussing on the too detailed elaboration in specification documents, rather than relying on the efficient communication between skilled experts, he said.

We're often unaware of the actual use of the services. For example, a telecom network could be dimensioned to handle a certain call volume and make a certain revenue. But then measurements reveal a huge number of "unanswered calls" in some countries, where people use the simple "ring" as a form of free of charge communication. The network practically does the same work for setting the call up – irrespective of being answered or not, but no revenues are received from "unanswered calls".

It's a key difference if software is built for a single customer – a project, or for addressing multiple customers – a defined market, and it's the product manager's task to identify the minimum viable product (MVP) features and guide the team to start with these.

Making it easier to speed up cloud software

Software companies making cloud products increasingly use "hardware accelerators" to make their products run faster, making maximum use of all available computer hardware assets. But this can make the software much more complex. MicroLab is aiming to make it simpler.

Software companies making products for running on cloud servers are increasingly using "hardware accelerators", software tools which make the software run faster, by making better use of all available hardware assets.

Hardware accelerators add an extra layer of complexity to software which is already highly complex.

To make it easier to understand, MicroLab is developing hardware accelerator tools which can be easily plugged into existing software, designed for specific tasks such as data compression, encryption, data analytics and machine learning.

MicroLab is the National Technical University of Athens (NTUA) Microprocessors Laboratory & Digital Systems Laboratory.

These hardware accelerators can then be simply bought by the software companies, for example on the Amazon Web Services (AWS) market place, said Christoforos Kachris of MicroLab.

This enables software companies to see their products more simply – as a map of building blocks – with software connected with a hardware accelerator – rather than taking all of their software apart, he said. Software developers do not need to get into the nitty gritty of how software runs on the hardware.

There is a growing need for hardware accelerators, because the demands for cloud computing are going up all the time – but cloud computing centres are not able to meet this demand simply by adding in more chips and servers, because the requirement for power, to cool the servers down, just gets too large.

Already today, the world's data centres use as much power as an entire mid-sized country, such as Germany or Italy, he said.

The hardware accelerators take advantage of the fact that not all cloud computing applications have the same demands on hardware. Some need more memory, some need more computing power, some need more input-output.

But usually every software system runs on the same standard computer server. This is equivalent to using the same mini-van for all deliveries, small or large, rather than having a delivery vehicle appropriate to what you are delivering, Mr Kachris said.

The hardware accelerator can enable each software application to use the processing, storage and input-output assets it needs, while leaving the assets it does not need for other software applications. Companies can instantly see accelerations of around 16x in software speed, he said.

Questions to Mr Kachris included whether trends for having dedicated processors addressing the variety of software functions especially under varying security demands is an emerging trend. This no doubt requires software mapping in order to manage at deployment time.

A failed software project – Athens metro

The new Athens metro ticketing system, introduced in 2017, could be reasonably considered a failure – increasing the time to issue a ticket from 25 seconds to 65 seconds, and creating much frustration and confusion among customers, said Tasos Makris of Gourdomichalis Maritime. A map might have helped ensure that the software delivered what was intended

In 2016, the Athens Metro introduced a new ticketing system. The previous system, with a small range of tickets delivered by entering coins, took on average 25 seconds, and everybody knew how to use it.

The new system involved screens, complex dialogues, confusing products, and a minimum of 65 seconds to issue a ticket, said Tasos Makris, usability consultant with Gourdomichalis Maritime. There were long queues, much confusion, and sometimes handwritten instructions from station managers as to how to use it.

So this could reasonably be considered a software failure. Perhaps a better map would have helped keep the project on track.

As a customer "I felt cheated by what they did," he said. "It was a big failure. It was a case of the user being ignored. You see people standing in front of the machine not knowing what to do. Everyone saw the queues and felt the frustration. I feel we should take a lesson from what happened with this project."

People sometimes cite 5 rules of usability – efficiency, effectiveness, error tolerance, easy to learn, engaging. This system broke them all, he said.

The first step was to choose a language – although the order of languages did not make sense, with English, the most popular second language after Greek, not being the second choice. Languages were represented by a country flag, so English was shown as a Union Jack flag, which could have been confusing to an American.

Tasos Makris, usability consultant with Gourdomichalis Maritime.

The next step was to "select a product" – itself not intuitive since a ticket is not normally thought of as a "product". And the first "product option" was a bus ticket to Athens airport, although it was a metro station. The most popular choice of ticket, a single 90 minute ticket, was last on the list.

There was a delay between each screen. "I could not understand why we had to wait for the machine to process some information which is trivial," he said.

There were voice commands from the machine, which were different to the written commands on the screen. With several machines in a row, it could be hard to identify which machine they were coming from.

There was a logic to how bank notes should be fed into the machine, which was not explained to the user.

There was an error in the calculation of change on the screen, showing that a Eur 20 note paying for a Eur 13.50 ticket would give zero change.

In some stations, the station staff made handwritten signs showing how to operate the machine – although the instructions should have been on the ticket machine screen.

The long delays between screens may have been due to an internet communication with a central server – in which case some other way could have been found to do it. In fact, a few months before the conference, the machines suddenly speeded up, he said, now taking 35 seconds.

Another usability error was that if you were using a ticket on the bus, the most important information, of how much time you had left on the ticket, was only displayed for 2 seconds, after a notice of how many journeys you have left on your ticket.

Mr Makris believes that IT skills in the public sector of Greece may be very poor, a belief compounded by an experience trying to check into an aeroplane of a state owned company Olympic via a mobile phone app, with 5 fellow IT managers, spending 10 minutes and ultimately failing.

"The interface for the mobile app was totally useless. If you try to do the same thing on a PC / laptop it works very well," he said.

People developing such systems should think hard about how to build the user interface. They should ensure the user knows how to use the software – because otherwise the software is "useless", he said. "Software is being built by experts and used usually by untrained people."

There could be a simple solution, such as better wording.

With a better map behind the design, the experience of using the ticket machine could have been completely intuitive, people don't need to think at all when they use it. Help is exactly where people need it.

One possible reason is that developers and engineers just do not personally care or have any empathy for the user, they are considering only the technological problem, and they may be incapable of seeing something from someone else's point of view, he said.

A map for digital communications in an electricity distribution networkticketing system

Greek electricity distributor HEDNO is aiming to simplify the communications between the various entities involved in the network by adopting a "common information model" for all digital communications.

Greek electricity distributor HEDNO (Hellenic Electricity Distribution Network Operator) is a regulated organisation for distributing electricity in Greece, responsible for medium and low voltage distribution. It distributes 4GW of power across Greece, with a 236,000km network.

There is a big move towards 'smart grids', with many new pieces of equipment and software which is added to the grid including smart meters, renewable energy sources, electric vehicles, batteries, market players and flexibility services as demand response. All these systems need to talk to each other.

The overall network need to manage risks, maintain performance and enable top management visibility and grid performance through predefined KPI's.

HEDNO is aiming to simplify the communications between the various entities involved within and outside the boundaries of the organization by adopting a common map or "common information model" (IEC@CIM) for all the digital communications in XML file format. CIM model is maintained in UML by IEC (International Electro technical Commission).

Maria Symponi, IT consultant at HEDNO.

Each communication entity is required to translate its digital communication data into the model specified by the CIM, explained Maria Symponi, IT consultant at HEDNO. There have been efforts in the past to make 1:1 integration between different systems which have been proven costly and complicate to manage. In the new approach there is a central enterprise service bus (ESB) to handle the way information is published among different entities.

Communication entities so far include SCADA systems for network automation, outage management system, Geographic Information System for storing geospatial data, a system for taking data from customer calls and AMR (Automated Meter Reading) systems. Those are more or less closed systems with proprietary software.

This effort is implemented in the context of a European Horizon 2020 funded program called WiseGRID. WiseGRID program has 20+ involved parties and five European pilot sites. Its primary goal is to use state of the art ICT services smartening the distribution grid, offering flexibility to consumers and prosumers. WiseGRID technological solutions help incorporating renewable energy resources and storage systems along with electric mobility to the grid.

Mapmaking – why we need more than UML

Many software people advocate Universal Modelling Language (UML) for software modelling or map making. But it is not up to the job of making maps for today's software, being based around entities not goals.

Many software developers see "Universal Modelling Language" (UML) as the way to make models – or maps – for software projects.

But it proves very limited in complex organisational software projects such as the ones described here – because it is built around objects, Dimitris Lyras explained.

In the early days of organisational software, when software was basically managing bureaucratic objects like purchase orders, invoices and bank accounts as used in common transactions, an object-centric modelling approach would have been appropriate if no changes or improvements were expected.

But expectations have increased, we are seeking to make software to help support people in organisations who have specific goals to achieve, and bureaucratic objects are rarely critical in that.

To illustrate this, consider that object-orientated designs will typically be used for accounts software, transaction management software, managing a library or catalogue, managing documents, managing personnel records, or other administrative functions in a company.

But no company ever says that they have achieved their high level goals simply by having some system to organise their bureaucracy, he said.

A high level goal could be that of the electrical company, seeking to unify the systems across different business units, the shipping company seeking to serve company gaols in an explicit way with the software that is being designed and implemented, a new product company linking all its development efforts to known client goals and measuring the effect of each of their proposed features, a metro software system with an easy to manage user interface and clear performance goals applied to transactions with adequate provisions for simulations under real conditions.

Liking functionality to goals no matter how fine grained the functionality and regardless of how encompassing the goals is the role of software mapping. Even unifying attributes to integrate system as in the electrical utility example is a goal based mapping exercise involving the effect of the varying semantics on common goals.

It is possible to see how good object based administrative-type software can put in place a low level baseline for all of these cases, but probably have little bearing between success or failure.

But other sorts of software could make a real contribution. For example, as discussed above, software tools which can relate decisions about ship maintenance expenditure with the high level shipping company goals and the prioritisation of these many competing goals that varies every day.

Maps do not need to be technical entities in themselves – they can be something drawn with a pen.

But if they are used as a basis for making software, they should accommodate all concerns easily. Easy is the key factor here. The expression of the requirements and dependencies no matter what they are. And then to evolve into sufficient granularity – to tell programmers everything they need to know – leaving nothing to further to interpretation.