Traveling Architects – A New Way of Herding Cats

Aino Vonge Corry[1], Klaus Marius Hansen[1], and David Svensson[2]

[1] Department of Computer Science, University of Aarhus
Aabogade 34, 8200 Aarhus M
{apaipi,klaus.m.hansen}@daimi.au.dk
[2] Department of Computer Science, Lund University
Ole Römers väg 3, 223 63 Lund, Sweden
david@cs.lth.se
The PalCom Project, http://www.ist-palcom.org

Abstract. Making software developers work towards a common goal may be likened to herding cats. If we further spread developers around the globe, we run increased risks of being unable to design and impose coherent software architectures on projects, potentially leading to lower quality of the resulting systems. Based on our experiences in a large, distributed research and development project, *PalCom*, we propose that employing techniques from active user involvement in general (and from participatory design in particular) may help in designing and sharing quality software architectures. In particular, we present the *Traveling Architects* technique in which a group of architects visit development locations in order to engage developers and end users in software architecture work. We argue that using techniques such as these may potentially lead to higher quality of software architectures in particular for systems developed in a distributed setting.

1 Introduction

Consider the following scenario, taking place in a distributed software development project, where development teams at different sites cooperate towards forming a common architecture:

> *Two Traveling Architects, Mario and Lisa, visit a site in order to have a Traveling Architects workshop. The developers at the site have prepared a presentation of the end-users, the architectural requirements and the prototype they have imagined. At this site they work with rehabilitation of people who have had hand surgery. They have a number of scenarios that they want supported with a prototype, such as sharing of ideas between a group of rehabilitation patients.*
>
> *After the presentation, the developers go through the different parts of the prototype they want to build and discuss whether it would be beneficial to implement all of it. Lisa and Mario advice on the use of the*

current common architecture, and note components that are candidates for placement in the project's toolbox of reusable components. They agree that some parts, such as the recording of video at consultations with physiotherapists, has to work in order for the end-users to be able to participate in the next application design meeting. Other parts, like the sharing of data over wireless net, could be simulated, because designing them wouldn't add to the common architecture.

During the meeting, Lisa and Mario create Unified Modeling Language (UML; [19]) diagrams of object and class models and sketch a documentation note on the prototype. After the meeting, the documentation note is finished and sent to the application developers to check for misunderstandings. Returning from the site, a meeting with other architects in the project is held in order to propagate the knowledge of the requirements and the input to the common architecture.

This is an example of the use of the Traveling Architects technique, taken from our work in the EC-funded Integrated Project PalCom [20]. PalCom explores the concept of *palpable computing*, denoting a new kind of ambient computing which is concerned with the above and other user-oriented challenges in complex and dynamic ambient computing environments. The two primary goals of the PalCom project are to explore the concept of 'palpability' and to design an open software architecture for palpable computing. In the following section we will present more about the project as a background, before discussing the Traveling Architects technique in more detail.

2 The PalCom Project

One of the subprojects in PalCom is the "Pregnancy and Maternity" project, where IT support for pregnancies is investigated. The vision is to equip pregnant women with a device called 'the Stone', which can support them during their pregnancy:

Alice comes home, greets her husband Bob and wants to show him something on a device she holds in her hand. The device has a very small screen and suggests using the TV as an external display. After Alice has accepted it, a film is shown on the TV. It is a recording from the ultra sound scan she went to that day because she is pregnant.

Since Alice is also diabetic, she has to measure her blood sugar level regularly. Her measurements are uploaded daily to the national Electronic Health Record (EHR) system, where experts will get a warning if the measurements are out of the ordinary. She has set the Stone to upload the data every night without her need to accept. One night the EHR system is unavailable. The Stone lets Alice know that is has tried to send the data without success. She then places it next to her computer, and the display is now on her computer screen. She points at the notification about the missed send of a message, and a graphical view of the connections is

shown. On this she can follow the connections between devices and see that the Stone looses the connection with the EHR system at the EHR system installation and thus the problem lies at their end, not hers. She chooses for the Stone to try and contact the register until is succeeds and then let her know that it has succeeded.

When the Stone and its environment work as they are supposed to, the user does not have to think about the state of it or how it works. Only in situations where there is a usage breakdown should the user become aware that something should be changed and maybe decide how it should be changed. This visibility/invisibility trade-off is one of the challenges of PalCom.

In the situation where the Stone has to make use of external displays and keyboards, there is a composition of services and devices in the system. This composition happens dynamically and can be decomposed at any time, yet still keeping the system as a whole stable. Composition/decomposition and change/stability are also challenges in the PalCom project.

Twelve companies and universities from six European countries take part in PalCom. The research work is organized into sixteen work packages, and each of the partners are involved in several of them. The activities reported in this paper concern the connection between the work package responsible for the common PalCom architecture, and the *application prototyping work packages*, which work with prototypes at different sites, forming palpable computing in contact with its users.

Figure 1 shows the overall development strategy in the project: results from general design and architecture work influence the application prototypes, and vice versa, throughout the 48-month project.

The application prototyping work packages are the following:

– *WP7 On Site* supports the work of professionals working in the field, in particular landscape architects. One example of work is the prototype Site-

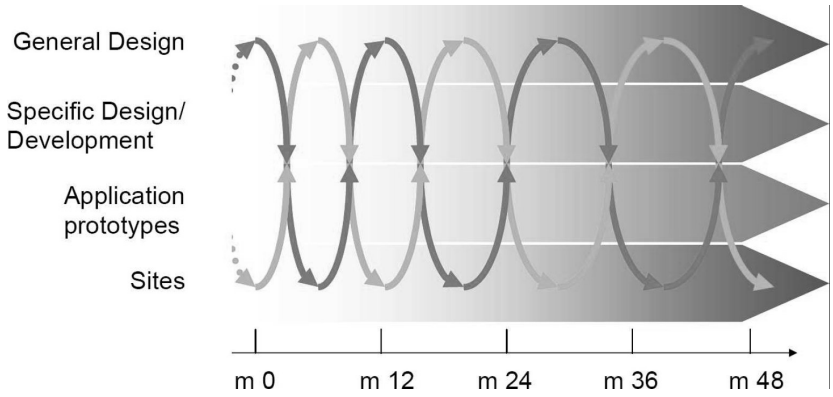

Fig. 1. Overall development strategy in PalCom.

Sticks, which is used for location-based visual assessment at a construction site where a major European bank is building their new headquarters.
- *WP8 Major Incidents* supports first aid personnel at the site of a major incident, during ambulance transport of patients, and after having arrived at the hospital by the use of among others sensor networks. The work in this work package is carried out together with the hospital, the police and the fire department in Aarhus, Denmark.
- *WP9 Pregnancy and Maternity* creates application prototypes supporting women and their families during pregnancy and (early) maternity. The Stone, discussed above, is one of the WP9 prototypes.
- *WP10 Surgical Rehabilitation* builds prototypes in the context of rehabilitation after hand surgery, in cooperation with Malmö University Hospital. One example is a video recording prototype which explores tangible user interfaces in the context of physiotherapy consultation.
- *WP11 Care Community* supports disabled children and adults in rehabilitation, both at the hospital and, e.g., in the swimming pool. One prototype is the Active Surfaces, where a set of computerized tiles to be used in the swimming pool can be configured by a therapist for aid in various sequencing and positioning exercises. Another aspect of this work package is the research in incubator support. The Incubator prototype makes it possible for doctors to manipulate the position of a baby without opening the incubator, preserving the micro-environmental conditions inside.
- *WP12 Transient Locations* supports users seeking connectivity in ad-hoc and hybrid networks. The prototype RASCAL (Resilience and Adaptivity Scenario for Connectivity over Ad-hoc Links) makes use of agent technology for adapting to available network access resources and to get an overview of possible connectivity issues.

3 Introducing the 'Traveling Architects' Technique

The PalCom project has a number of challenges as alluded to by the above that are of high relevance to software architecture design in the project:

- *Distributed development teams* working with actual end-users at various locations. The collaboration with these end users at specific locations are crucial in understanding and designing for local work practice. One example (given in Section 2) is the collaboration with landscape architects.
- *An iterative, experimental, and incremental approach* to development. The project has a high degree of complexity, uncertainty and potential change of requirements. This means that the project employs agile development principles [11]. One consequence of this is that the software architecture is continually evolving and no full set of architectural requirements exists for the software architecture at a given point in time. Further, during our initial Traveling Architects work there were not always clear guidelines that could be readily presented to the prototyping teams. We could often only provide

them with guidelines as to what direction the architecture should progress and whether their architectural requirements were addressed by others.
- *Limited central control* of software architecture design. Creating a software architecture in PalCom is in many ways a consensus-making process. With the outset in the central challenges of palpable computing on the one hand and concrete, usable application prototypes on the other hand, the architecture is gradually designed and evolved, catering for functional as well as quality requirements.

A Definition

To tackle such challenges, it was decided to form a team of architects that would be responsible for maintaining the architectural vision of PalCom by both working from the specific (application prototypes) and the general (the concept of palpable computing and architectural requirements) and for making sure that this vision was shared by all sites. In this way, the concept of 'Traveling Architects' was born. To be more precise, we define the concepts as follows:

> *Traveling Architects: a group of architects responsible for maintaining software architectural assets in a distributed development project by visiting development sites in order to design, evaluate, and enforce a software architecture in active collaboration with developers and possibly end users*

If we dissect the definition, a number of components are of interest in this context: "group of architects", "visiting", and "active collaboration".

With respect to the group of architects, the concept is related to the ArchitectureTeam pattern from Coplien's set of organizational patterns [8]. The main similarity is that they form a team of architects that can communicate about the architecture with each other and the groups of developers. In Coplien's ArchitectureTeam, the team of architects makes the initial architecture, which is not the case for the Traveling Architects. They spread the word of the evolving architecture and collects input to its evolvement. The decision of the concrete architecture is done in a team of architects of which the Traveling Architects are members. This team is built like an ArchitectureDefinitionTeam as described in Mezsaros' patterns [17].

The rationale for actually "visiting" remote development sites is closely connected to the rationale for "active collaboration" which is again closely connected to the concept of 'participatory design'. In fact, the Traveling Architects technique may be seen more generally as part of an attempt to employ participatory design techniques in software architecture work. Thus, we describe this concept and its potential relation to software architecture next.

Participatory Design and Software Architecture

Participatory design [13] is concerned with involving stakeholders (e.g., developers, end users, managers, or customers) of IT systems in the collective design of

these systems. This is done for both moral and practical reasons [7]: Practitioners are the ones who need to live with the consequences of IT systems and they are the ones who have the real competence in what is to supported and who know the real, practical problems. Bringing diverse and indispensable competencies together is thus seen as a main part of doing systems development[3].

Concretely, design is often done collaboratively in workshop-like settings by users and designers/developers and informed by (ethnographic) studies of actual work practice [5]. The following Figure 2 illustrates the concepts at play:

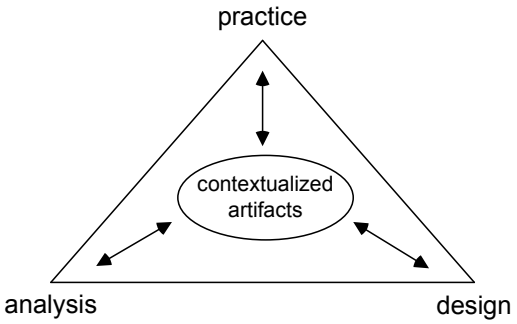

Fig. 2. Artifacts in participatory analysis and design (adapted from Mogensen and Trigg [18])

Here it is illustrated how analysis (e.g., work analysis through ethnography or participatory analysis), design (e.g., design of user interfaces), and practice (e.g., the work of users collaborating in design or analysis) communicate and collaborate through common artifacts. Such artifacts could, e.g., be user stories, sketches on whiteboards, or actual prototypes of applications. A number of techniques may be employed in order to work with and create these artifacts collaboratively: creating mock-ups, cooperative prototyping, future workshops [13], or use of situation cards [18] are examples of such techniques.

An example from PalCom is the use of participatory design at workshops in the *Pregnancy and Maternity* project. A number of healthcare providers involved in pregnancy were gathered at a workshop. They were given an introduction to the ideas formed by the designers. The ideas were based on field work done in the healthcare system and thus grounded in the work of the people present. The healthcare providers did not show great enthusiasm until we presented a concrete artifact to them. As an example of the Stone we had discussed with them

[3] Indeed some of the work on participatory design has influenced current thinking on software methods although techniques such as the On-Site-Customer of eXtreme Programming [4] has taken an analytical standpoint, removing practitioners from their work.

previously, we had brought a PDA with a primitive application. The application could, e.g., send a Java program to a mobile phone in order to make it vibrate at given intervals. This was done in order to help the pregnant women do their birth preparation exercises. The healthcare providers responded to the artifact, now suddenly understanding the concept and able to form new ideas also for extensions of functionality. The change in their attitude, when presented with an artifact, was remarkable. We have seen this effect numerous times in the PalCom project. Sometimes with applications, other times just with screenshots or natural artifacts, such as actual stones.

The main insight we use here is then that the same set of concepts as presented and illustrated above can be applied to the creation of *software architecture* rather than for the design of artifacts *directly supporting* work practice of users. Figure 3 tries to illustrate this. Here, the practice involved is that of

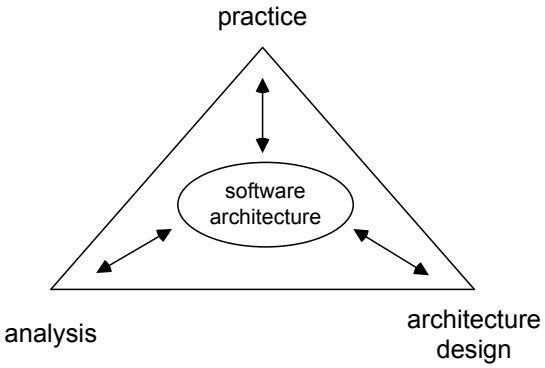

Fig. 3. Participatory design of software architectures

developers (and perhaps transitively or directly that of end-users) including the designs they have produced, application architectures, prototypes etc. The artifact that is collaboratively being constructed is a software architecture in various forms: UML-based descriptions, architectural prototypes [2], verbal accounts continuously being given by developers and architects etc. The next sections will discuss the generalizability of the approach and then detail our experiences with the Traveling Architects approach and discuss concrete details of how to involve developers in the architecture work.

Generalizability

A natural question to pose is to which extent the PalCom experience is unique thus making the experience of the Traveling Architects only relevant to this particular project? Here, we point to two concrete examples of development

efforts that have many of the same project characteristics as PalCom and that could thus reasonably use the presented techniques.

The first example is a set of projects aiming at realizing Danish EHR systems in all hospitals by 2006 [10]. Each of 14 Danish counties have had the responsibility of introducing an EHR system for the public hospitals in that county. This approach was chosen for multiple reasons: one was to be able to use various vendors in order to get vendors to compete on price and quality and another was that local hospitals have competent staff with partially local competences that need to be supported. One consequence of this choice has been that there has been no central control and thus no single software architecture (or integration platform) for all systems. This gives a number of challenges in integrating patient data from the systems, a particular twist to the problem being that the counties will merge to become larger units by 2007. Some of the issues of integrating the disperse systems are discussed in [14].

The other example comes from that of a globally distributed company that one of the authors worked with as part of an effort to build a series of prototypes for a global customer service system. The company had previously built a number of regional service systems. These were built regionally and built in order to support local practices well; customer interaction was, e.g., very different in Asia in terms of customer loyalty than in other parts of the world. Again, this led to (technical) problems when trying to enable service agents to provide service on a global basis among others due to the lack of a common architecture of the systems.

4 The PalCom architecture

To give an overview of the PalCom open architecture that we refer to, we have included a part of our ontology for the architecture in Figure 4 (see [21]). The ontology may be thought of as a (very) logical view of the PalCom architecture, showing the concepts (and relationships) that are realized in the architecture. We do not intend this presentation to be exhaustive, but it gives a picture of what we are using to guide us in our work as Traveling Architects.

Going back to the scenario of Section 2, we see several of the concepts in the ontology in play. Alice is an *Actor* in a PalCom system. She makes use of several different *PalCom Assemblies*. One example is the Assembly formed between the Stone and the TV, which moves the displaying from the Stone to the TV screen for showing the recording from the ultra sound scan. A second Assembly comes alive when Alice connects the blood sugar measurement device to the Stone for storing measurement data, and a third one connects the Stone to the EHR system, managing the upload of data. The role of each Assembly is to coordinate a set of *Services*, communicating over *Communication Channels*. Each service is offered by one *Node*: the TV has a display service, the blood sugar measurement device has a measurement service, and the EHR system has a registry service.

The software implementation of the nodes is what distinguishes a *PalCom Node* from a *Non-PalCom Node*. The Stone is a PalCom node, because it hosts

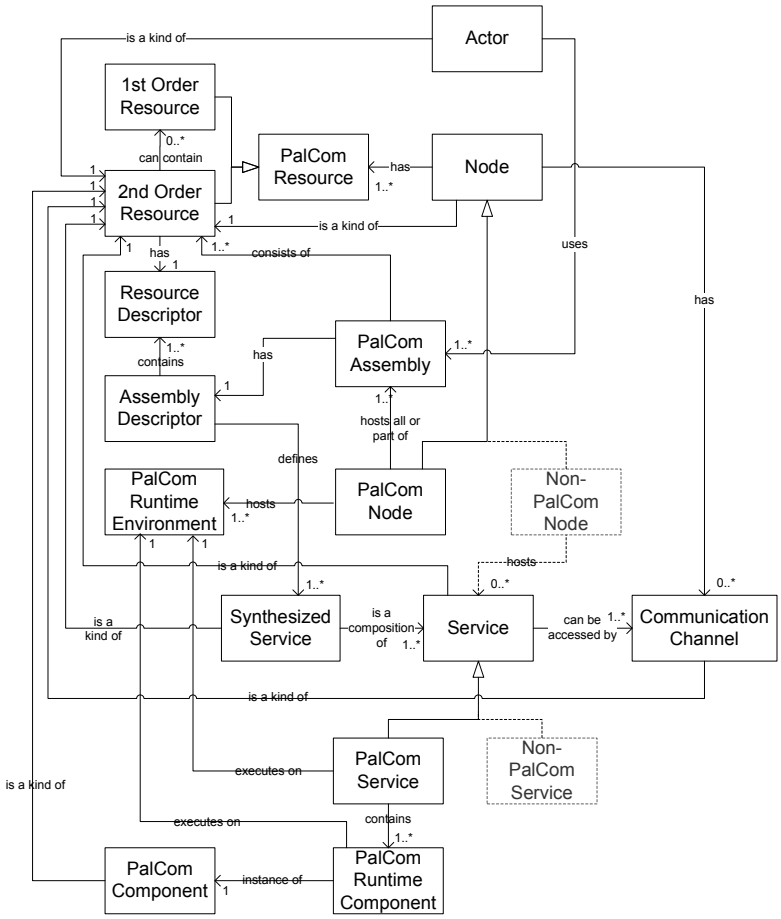

Fig. 4. PalCom concept ontology

a *PalCom Runtime Environment*, which realizes the Stone's services through execution of *PalCom Runtime Components*. The TV, on the other hand, is a non-PalCom node. Its software does not run as PalCom Runtime Components, for reasons of legacy code or hardware restrictions, but its services are externally accessible in the same way as the services of the Stone.

Different kinds of restrictions in hardware are also targeted by the concept of *1st Order Resources*, which are associated with physical devices. Examples are CPU clock speed, memory, bandwidth and power. Taking these into account, together with *2nd Order Resources* such as Services, Nodes, and Actors, makes it possible for a PalCom Runtime Environment to adapt to varying resource conditions.

Figure 5 exemplifies how the different concepts may come together in a PalCom system. It can be seen how the Assembly XYZ references services on three different nodes. The *Synthesized Service* is a service that PalCom Node B offers on behalf of the Assembly, offering combined functionality not given by the individual services themselves.

5 Applying the technique

The idea of Traveling Architects has been implemented during a period of about one year, in the second year of the PalCom project. In total, there have been eight meetings where the Traveling Architects have held workshops with the people from application prototyping work packages, and discussed the architecture of their respective prototypes in relation to the architecture. The general setup of the meetings followed the workshop example given in the introduction. On each occasion, the Traveling Architects team consisted of one or two people. There have also been general architecture meetings, where the Traveling Architects discussed their findings with the ArchitectureDefinitionTeam. Figure 6 shows the time line of the meetings and workshops. Next, we present experiences from these.

5.1 Techniques, artifacts, and meeting types

The meetings and workshops have been organized in different ways, which has given experience from several types of situations that the Traveling Architects have experienced. Different aspects of our work as Traveling Architects were present to varying degrees in the different meetings.

Within each site there has been iterative development based on participatory design. At the meetings we experienced the difference in the maturity of the development of application prototypes. This difference has also influenced the type of meetings held, the techniques applied, and the artifacts present.

Techniques When interviewing the developers at the sites, we did not use an architecture-specific technique like ATAM [15] or QAW [1]. We performed a more ad-hoc, exploratory approach, partly because the prerequisites of the meetings

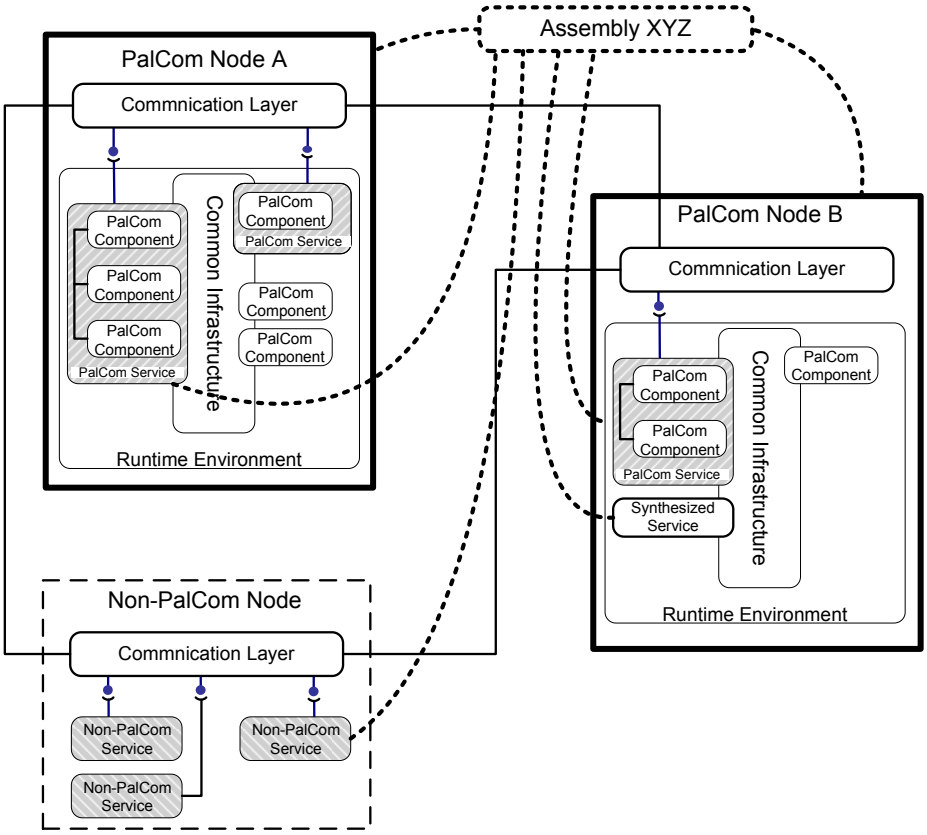

Fig. 5. An example of a PalCom system.

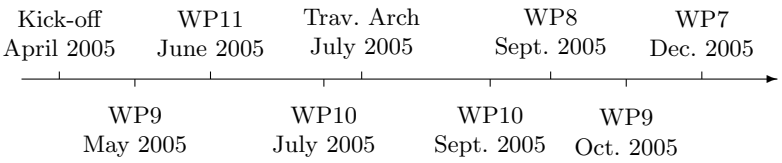

Fig. 6. Traveling Architects meetings.

were so different, partly because we wanted to free ourselves from restrictions in the beginning of the implementation of the concept. Some software development techniques were used, like the implementation of architectural prototypes [2]. When an idea had emerged, it was beneficial to implement and evaluate it before an application was based on it. We also used some of the activities described in RUP [12], e.g. the realization of use cases.

Artifacts Since our task was to design, discuss, and document architecture, UML diagrams were used excessively to describe different views of the architecture. We had different stakeholders present at the meetings and they needed different views on the architecture, as described in [16]. At some meetings, where stakeholders such as end-users and ethnographers were present, we would draw the main use cases in order to determine if we agreed on the overall idea of the system. These use cases were later used to construct test cases to compare the architectural prototypes and applications with.

When other architects and some developers were present, we discussed the architecture seen from a module view with class and package diagrams. Other kinds of developers were more keen on discussing code snippets with the Traveling Architects or looking at allocation views of the design.

Whiteboards were often used as the common physical artifact in the center of the discussion. They have the benefit of enabling a number of people (there were often more than seven people at the meetings) to view the same diagram. Also, changing a figure or a diagram during a discussion is relatively easy on a whiteboard. We would potentially have benefited, though, from having tool support for our cooperative work as presented in [9].

Meeting types When we started the implementation of the Traveling Architects concept, we had imagined that the meetings would all be consisting of an aspect of documentation of the architecture and an aspect of designing architectures. Due to different challenges this was not what happened. At first this seemed frustrating, but eventually the need for Traveling Architects became more evident. The Traveling Architects just had a change of responsibility to often be more communicators than designers. These are examples of the types of meetings we have had:

- *Discovering architectural requirements*
  The developers had a somewhat clear view of the requirements of the application to be built, but they had not articulated them. The Traveling Architects thus had to distill from their knowledge what had architectural significance. Since the Traveling Architects are there to discuss architecture and often are not domain experts, this is not optimal.
- *Architectural reviews*
  The designers had documented the architecture before the meeting and we went through it together. This made the architecture more clear for the designers and gave the Traveling Architects real input for the PalCom architecture. A number of people interested in architecture met to discuss the

architecture that we had defined at one of the prototype meetings. A lot of architectural discussions were triggered. This was interesting for most of the attendants but not so beneficial for the specific prototype architecture, because there were too many architects present.

- *Documenting architectures*
  A prototype had been implemented, but the process of developing it had been unstructured and there was no documentation of it. The Traveling Architects went through parts of the code together with the developers and documented the main parts.
- *Designing architectures*
  Some developers had clear architectural requirements and designing an architecture was a natural next step. This aspect in a meeting was very welcome.
- *Designing prototypes*
  A Traveling Architects meeting could be placed within a design meeting for a prototype to be built. It was useful to have a connection to the PalCom architecture in the discussion, but it was often difficult for the Traveling Architects to give input to the discussion about details in the specific prototype design.

6 Experiences

After some time as Traveling Architects, we look back at the work and the results, and we can see benefits of the work. Generally, the Traveling Architects' visits have worked as good opportunities for the prototype teams to discuss their architectures and have given valuable input to the PalCom architecture per se.

The work has given a number of benefits to the PalCom project, which we believe in themselves motivate the costs in terms of traveling and effort:

- The ArchitectureTeam has got hands-on experience with the prototypes.
- The knowledge of the status of the different prototypes has been spread within the project.
- The deeper insight into the prototypes made it possible for the architects to guide which requirements that are special for the different prototypes and should thus be focused on.
- There has been an increase in cross-partner collaboration.
- We have found new tools for the PalCom "toolbox", where development tools and software components are collected for use by other parts of the project. One example is a video streaming component developed in WP10, which was spotted as directly usable also in WP9.
- The architectures of the application prototypes have become clearer.
- Last, but not least, the software designs were documented.

We have also got an overview of the "hotspots" in the requirements and the architecture. These hotspots are issues spanning over several work packages, such as

- the location and role of the *Assembly Descriptor* in a PalCom system. The Assembly Descriptor specifies an assembly, which defines compositions of services and coordination between them.
- the sharing of data between services on the same device.
- the need for remote contingency handling, i.e. for dealing with errors that occur on other devices than the one an actor is interacting with.
- the distinction in PalCom between a *Service* and a *Resource*.
- the management and storage of data: when should it be handled locally, and when should we use a centralized model?
- the issue of partitioning a system into services of appropriate size and complexity.

In the challenging context of the PalCom project, the Traveling Architects initiative has helped keeping the focus on the architecture, and on the project as a whole. It has also helped speeding up the influences in both directions, between the prototypes and the architecture. Hopefully, in this way it has had a positive impact on the quality of the software architecture. Even though we do not have firm evidence of the way that the technique influences quality attributes, we speculate that the technique itself mostly influences architectural quality attributes (cf. the characterization of Bass et al.[3]) such as buildability and conceptual integrity. When speaking of conceptual integrity as it is described in [6] one normally refers to one system. In PalCom the situation is a bit different since there is no one system, on the contrary there exists an unlimited number of subsystems, that can stand alone or work with the others. And it is this strong challenge of construction of these small subsystems, that makes conceptual integrity important for PalCom. One major goal of the Traveling Architects is to achieve this conceptual integrity.

Enhancing system quality attributes (such as modifiability, availability, and performance) is probably more tied to the possible expertises of the individual architects traveling.

Moreover, It is a well known phenomenon that meeting face-to-face has advantages, compared to other types of communication, such as e-mail or phone conferences, see e.g.[22]. This is emphasized, e.g., in Coplien's patterns LockEmUpTogether and FaceToFaceBeforeWorkingRemotely [8] and was also experienced in our work as Traveling Architects.

Regarding the size of the Traveling Architects team, we feel that two persons was a good size, given the size of the prototype teams of about five persons. It is good not to be alone as Traveling Architect, and to be able to discuss things with the other architect directly at the meetings.

7 Conclusions and Future Work

Traveling Architects is clearly a promising technique. In the PalCom project, it has helped us work towards the definition of a common software architecture, with development teams spread across Europe. The setting is that the teams work in an iterative, experimental and incremental way, in close contact with

end-users, and with limited central control. We have noted a number of concrete benefits already, of which the improved communication and spreading of knowledge in the project is the most apparent—the Traveling Architects have often been as much communicators as architecture designers. In the long run, we believe that our effort will lead to a higher quality of the software architecture.

We have seen advantages of applying concepts from participatory design to the creation of software architectures. Consciously working with UML diagrams, whiteboard drawings and code as artifacts has helped the active collaboration between architects and developers, and made the architecture more concrete for its users, the developers.

A number of different types of Traveling Architects meetings have been tried. Our conclusion here is that the documentation and design of architectures are the main ways of obtaining value from the meetings. What we had not expected, but what came as a positive side effect, was the communication of architectural requirements and ideas that were not already in use.

A relevant question is that of generalizability: can the Traveling Architects concept be successfully applied to projects other than ours, e.g. to industry-only projects? Yes, we believe it can. We have given two examples of large, distributed development projects with limited central control and iterative development strategies, where the technique could presumably have been successfully used. In particular, we think that the hands-on aspect of Traveling Architects is beneficial.

The first round of Traveling Architects has been exploratory and experimental. In our continued work, we plan to apply the technique more systematically. Now that we have more experience with it, we can put up clearer goals for each meeting, and apply more techniques from participatory design such as mock-ups, situation cards, or future workshops. In the end, we hope to be able to provide a strong, scientific evaluation of the technique of Traveling Architects.

Acknowledgements

The research presented in this paper has been partly funded by 6th Framework Programme, Information Society Technologies, Disappearing Computer II, project 002057 'PalCom: Palpable Computing – A new perspective on Ambient Computing' (`http://www.ist-palcom.org`)

References

1. M. R. Barbacci, R. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, and W. G. Wood. Quality Attribute Workshops (QAWs), 2nd edition. Technical Report CMU/SEI-2002-TR-019, 2002.
2. J. E. Bardram, H. B. Christensen, and K. M. Hansen. Architectural Prototyping: An Approach for Grounding Architectural Design and Learning. In *Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture*, pages 15–24, Oslo, Norway, 2004.

3. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2nd edition, 2003.
4. K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
5. J. Blomberg, L. Suchman, and R. Trigg. Reflections on a work-oriented design project. In *Proceedings of PDC'94*, pages 99–110, 1994.
6. F. P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 20th anniversary edition, 1995.
7. M. Christensen, A. Crabtree, C. Damm, K. Hansen, O. Madsen, P. Marqvardsen, P. Mogensen, E. Sandvad, L. Sloth, and M. Thomsen. The M.A.D. experience: Multiperspective Application Development in evolutionary prototyping. In E. Jul, editor, *ECOOP'98 – Object-Oriented Programming. Proceedings of the 12th European Conference*, pages 13–40. Springer Verlag, 1998.
8. J. O. Coplien and N. B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice Hall, 2004.
9. C. Damm, K. Hansen, M. Thomsen, and M. Tyrsted. Creative object-oriented modelling: Support for creativity, flexibility, and collaboration in CASE tools. In *Proceedings of ECOOP'2000*, pages 27–43, 2000.
10. The EHR Observatory. `http://www.epj-observatoriet.dk/english.htm`.
11. M. Fowler. The new methodology. `http://martinfowler.com/articles/newMethodology.html`, 2005.
12. D. Gornik. IBM Rational Unified Process: Best practices for software development teams. Technical Report TP026B, Rev 11/01, IBM, 2001.
13. J. Greenbaum and M. Kyng, editors. Lawrence Erlbaum Associates, 1991.
14. K. M. Hansen and H. B. Christensen. Component Reengineering Workshops: A low-cost approach for assessing specific reengineering costs across product lines. In *Proceedings of the 8th European Conference on Software Maintenance and Reengineering (CSMR 2004)*, pages 154–162. IEEE Press, 2004.
15. R. Kazman, M. Klein, and P. Clements. ATAM: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, 2000.
16. P. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6), 1995.
17. G. Meszaros. Archi-Patterns. In *Proceedings of the conference on Pattern Languages of Programming*, St.Louis, 1997.
18. P. Mogensen and R. Trigg. Using artefacts as triggers for participatory analysis. In M. Muller, S. Kuhn, and J. Meskill, editors, *Proceedings of the Participatory Design Conference (PDC) 1992*, pages 55–62. CPSR, 1992.
19. OMG. Unified Modeling Language specification 1.5. Technical Report formal/2003-03-01, Object Management Group, 2003.
20. The PalCom Project. `http://www.ist-palcom.org`.
21. PalCom. PalCom External Report 31: Deliverable 32 (2.2.1): PalCom Open Architecture – first complete version of basic architecture. Technical report, PalCom Project IST-002057, December 2005.
22. S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson. How does radical collocation help a team succeed? In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 339–346, New York, NY, USA, 2000. ACM Press.