

# Appropriation: A work practice perspective to provisioning

## Ethnographical investigation into Eclipse Modification Strategies

Gunnar Stevens

Fraunhofer FIT

Sankt Augustin, Germany

[gunnar.stevens@fit.fraunhofer.de](mailto:gunnar.stevens@fit.fraunhofer.de)

Sebastian Draxler

University of Siegen

Siegen, Germany

[sebastian.draxler@uni-siegen.de](mailto:sebastian.draxler@uni-siegen.de)

Tobias Schwartz

Fraunhofer FIT

Sankt Augustin, Germany

[tobias.schwartz@fit.fraunhofer.de](mailto:tobias.schwartz@fit.fraunhofer.de)

**Abstract**—In this position paper we present an empirical study about the Eclipse IDE modification practice in a mid-size software development company. Provisioning mainly addressed from a central perspective, neglecting the artful work to take care about the own tools. In this paper we argue that *new provisioning technologies* should take appropriation processes into account. Finally we give a first outline about a software infrastructure, to show how such a technology might look like.

*Provisioning, Appropriation, End User Development, Ethnography, Work Studies*

### I. INTRODUCTION

*What shape should the new provisioning technology take and how will you use it to provision in your scenarios?*

Following the zeitgeist of web 2.0, user driven content, community first, etc. we want to motivate that the core of the new provisioning technology should be an infrastructure for cooperative software adaptation and appropriation.

To make our position clear, we want to present an empirical study, which examines the work practice of software engineering keeping their workbench up-to-date.

Usually provisioning is an approach, which should enable IT-departments or software producers to deliver their applications in an easy way to many users. The major design goal of such solutions is to provide a scalable solution which makes the different needs of the user groups (the lifecycle software, etc.) manageable for an organisation and to avoid compatibility problems regarding the heterogeneous components of a system. The common way to address this issue is a coordination-by-centralisation approach.

The users' perspective was mainly reduced to the usability aspect "*how easy a provision client can be used*", taking the coordination-by-centralisation solution for granted.

In the case of software developer professionals, a tailoristic approach of centralized workbench provisioning is not the right perspective for two reasons: Software development is not assembly line work but a craft, and secondly it is not done in isolation but in teams. This specific work structure has also consequences for tool modification strategies of software engineers. To assemble the right tools,

to know them - the features as well as the bugs - and to take care of them are critical competences of software engineers. In short, the engineer has to appropriate the production facility "to use it constructively, to incorporate it into one's life, for better or worse." [1]. Here, the personal knowledge as well as the knowledge of the team and of the community are major resources for the engineer.

We have condensed this insight into the German term "Kooperative Werkzeugpflege" as a mission statement for the design of an infrastructure for cooperative adaptation and appropriation (cf. section IV). The term can be translated into English as "cooperative tool modification or maintenance", but with the addition that the German word "Pflege" also connotes the aspect of taking care of something.

Before we give our answer to the question of the symposia, we first present an empirical study to show how the specific working structure will shape the modification practices and strategies of engineers to deal with the antagonism that we have found in respect to the topic of tool modification.

### II. RESEARCH CONTEXT

The present research was part of the BMBF project Component based End User-Development (CoEUD) (cf.: <http://www.coeud.org>). The CoEUD project was inspired by the concept of End User Development [2] in general and Component based tailorability [3, 4] in particular.

The CoEUD project was split up into four different application scenarios to identify similarities as well as variants from the perspective of component based end user development. In the following we give a general overview of all application scenarios related to Eclipse RCP before we go into the specific study of tailoring the Eclipse IDE:

- The first scenario deals with the further development of a groupware system that supports cooperative work. Within the project an administration toolbox should support the administrators in tailoring for specific needs. This toolbox will be based on the Eclipse RCP. We have to turn our attention especially to the aspect of tailoring. It is quite common, that administrators tailor their environment through scripts to better fit their work context. This aspect will be

mirrored in the administration toolbox. Here the specific issue is how different administrators could be supported in sharing their tailored artefacts.

- The second scenario adds the functionality of cooperation to a business game used by teachers in schools. This effort should enable them to share didactic concepts with each other. Since the teachers are highly separated and associated to different organisations, it is not possible to settle up a centralized solution across the IT-infrastructures of the different organisation. Therefore a peer-to-peer approach was preferred to enable sharing.
- The third use case was the development of Control System Studio (CSS) which started at DESY and provides a “comprehensive control system application framework” (cf. <http://css.desy.de>). The application was an operator cockpit that allows operators to monitor complex technical systems used for research in particle physics. In addition, the software allows its user to construct own controlling artefacts by assembling graphic components to a personalized control display. There is also an effort to provide an infrastructure which allows the sharing of artefacts within the world wide community of operators. Currently it is not clear which kind of provisioning infrastructure will be implemented, but based on the working structure of the target group an infrastructure for communication, adaptation and recommendation seems to be a promising candidate.
- Finally, our most significant scenario in the scope of provisioning was the adaptation and appropriation of the Eclipse IDE in a mid-sized software development company. Here we regard software developers as lead users [5] in respect to the emerging practice of personalizing software applications by assembling the application using a set of components. Through an empirical study we detected existing necessities as well as constraints for new provisioning technology (cf. next section).

Although CoEUD comes up with a variegated bouquet of application scenarios, we found cooperative adaptation and appropriation in all cases. This may indicate a need for new provisioning technologies.

### III. APPROPRIATION AND THE ACTIVITY OF TOOL MODIFICATION – SOME EMPIRICAL FINDINGS

“Appropriation can be described as a collaborative effort of end users, who perform “appropriation activities” to make sense of the software in their work context. Besides activities to configure the software to fit into the technological, organisational and individual work context of the users (‘Tailoring’), there is a larger area of technology-related communication, demonstration and negotiation activities aimed at establishing a shared understanding of how a software artefact works and what it can contribute to the shared work context.” [6]

The empirical study examined the appropriation and the tailoring of the Eclipse IDE as part of the daily work practice of the software developers in small and medium-sized enterprises (SME) of the German software industry. The focus on SME is motivated by the fact, that software development in Germany is typically conducted in such kind of companies [7].

The research design followed a Business Ethnography approach, which was mainly a combination of ethnographical study and action research [8, 9]. In line with the methodology of Business Ethnography, we conducted the empirical study as follows: First we surveyed the literature regarding the topic of software development with Eclipse to get a general access to the topic. Afterwards we conducted an empirical study by:

- Conducting 10 semi-structured interviews with developers using the Eclipse SDK for daily work. The interviews were recorded and transcribed.
- Observing participants to subject the daily work of developers to a critical scrutiny. We visited 2 SMEs for a fixed period of time (3-5 days). The observations were accomplished in the typical ambient of the developers in order to get a detailed insight of their work activities.

This ethnographical oriented study was part of an action research approach. Here we cooperated intensively with one of the SMEs, which have set up a project to introduce a proprietary provisioning solution, based on Subversion system (SVN)<sup>1</sup>. In three workshops with the project leader, we discussed requirements and implementation opportunities for a cooperative provisioning solution. In these workshops we profited from their first experiments with their solution and grounded our ideas on the administrator perspective.

The interviews were transcribed and the participant observations were written down to field notes providing a corpus for the systematic analysis of the field. In the next step we analyzed these empirical data for specific phenomena related to our research interest. The interest of the first investigation was set in uncovering and documenting situational work practices and strategies in order to identify the (latent) potentials that arise from the modification of the Eclipse IDE.

The interesting portions of empirical data were marked for a further microscopic examination [10] using the method of sequence analysis as suggested by Oevermann [11].

In the following we present a description of our observations. A systematic interpretation is then presented in section III.B.

#### A. Observed phenomena related to tool maintenance

First of all, the employees use the Eclipse IDE as a main tool for software development. In most of the cases no two Eclipse IDE installations looked the same. There are several reasons for this diversity, e.g. individualization and tailoring efforts, as well as “always be up to date” mentality.

---

<sup>1</sup> Cf.: <http://subversion.tigris.org/>

Furthermore no formal regulations for tool configuration exist, fact that promotes such an approach of individualization. But the heterogeneity of the workbenches is also an expression of the fact that tool competency is a core skill in the profession of software developers.

In the case study, we observed different strategies regarding how developers sustain this competence, but face-to-face communication and individual attitude seem to be important factors. Also, software developers actively searched the Web for new Eclipse components and test reports or other information, which may help them doing their daily work.

The fact that no formal regulations for tool configuration exist should not be misinterpreted in the sense that the organizational settings have no effect on the workbench configuration. Based on the empirical material we discovered that a specific project context plays an important role regarding tool modification.

More than anything, projects are the usual form of working together, providing a space for grass root innovation and raising awareness of interesting components or work practices.

In addition, we observed that the specific needs and the dynamics of the projects were a trigger for workbench modification. Most projects need some special tools (like PHP tools or an XSLT editor) and typically a project, starts with one or two developers, but during the lifetime of a project the core team sometimes called upon other developers as experts for specific technologies or tasks (e.g. someone who knows an particular database, or an UI designer). As a consequence of this, the consulted experts have to synchronize their working environment with one of the colleagues in order to cooperate with each other.

But although the project context usually affects the workbench modification, the force of the project context is based on social negotiation processes. This leaves room for self-reliance of the software developers. So, if a developer faces a situation that looks important enough, he may use his own way on workbench modification in order to get the work done and leaves the informal organization or project policy.

Beyond this outline of some triggers for workbench modification, the analysis exposed also particular fears concerning tool modification like a decrease of efficiency or a complete failure of the tools being used. We observed that users have developed their own strategies to deal with this issue. E.g. a senior Java developer, who continuously looked for new features and technologies, found a work practice for dealing with the antagonism of stabilization and innovation. Based on his experiences in the past regarding the spontaneous integration of new components into Eclipse which were accompanied by damages to the point of total blackouts, he changed this practice. Now he uses different Eclipse installations: One for testing the effects of modification and backup installations as a fallback system, and other older installations, which are out-of-date. If after an adequate period of testing no negative effects appeared

and the modifications act satisfying he carries over the modifications to his usual or backup installation of Eclipse.

## B. Analysis of observations

In this section we want to systemize our observation, present our hypothesis that workbench modification is an activity that has to balance out the aspects of integration and stabilization every time again. We first want to present two principal motives behind the workbench modification. Following, we present different arenas, which have been taken into account when balancing out the different aspects.<sup>2</sup> Finally, we want to present different tasks of the activity of workbench modification itself.

### a) Principal motives behind workbench modification

We have found two principal motives with which the activities of workbench modification have to be confronted. The first one is an antagonism, which can be characterized by the two poles:

- stabilization versus innovation (or informally spoken “keep a running system” versus “being up-to-date”)

This antagonism exists because of the fact that on one hand the area of software development is highly dynamic and innovative, where “standstill means a backward step”. On the other hand, each modification has to be appropriate and can stir up trouble into the working routine. This means that it is not for sure, that every modification is a step forward, but nevertheless it is important for the own competency to explore the future.

The empirical analysis showed that in the practice of tool modification the questions of “keep a running system” and “being up-to-date” have to be balanced out each time again and have to take the specific context into account.

The other motive is related to the political-economic question of possession. This question can be characterized through the motive by the dialectical relationship of

- heteronomy versus autonomy (or informally spoken “who defines the production means and how this definition comes into practice”)

As we see in the study, it is a common interest that each person should easily work with each other. There are several reasons why it is worth striving for such a cooperative integration. It supports the cooperative work inside the project, it lowers the burden of dynamical consulting of persons from outside of the project and individuals can benefit from the tool competency of the team. Nevertheless if such cooperative integration overturns heteronomy, it disrupts the craft oriented or art oriented concept of being a software developer professional, of being responsible for

---

<sup>2</sup> We use the word ‘arena’ to mark a place where different topics vie for the attention, rather than to be a coherent force that control the activity in a deterministic way. It was up the context specific circumstances and its interpretation through the people in which way an arena affect the modification of the workbench.

keeping tool competency up-to-date. In addition, it prevents the individual to react immediately to spontaneously emergent situations.

As we see in our case study, where our partners have started to experiment with an administrator centric provision solution, we see that a provision infrastructure will affect the fine balanced system of possession and responsibility. Here we see the major design challenge of finding a solution that supports cooperative integration; spontaneous reaction of software developers to emergent situations; and the development of tool competency and local innovation.

*b) Separation of the different arenas which have to be taken into account in workbench modification*

In order to gain a more profound understanding of workbench modifications as an activity which has to deal with different aspects, we have to separate the arenas which play a role when developers are balancing out the question of tool modification. These arenas are primary analytical categories, but are motivated by the analysis of the empirical data.

- the personal disposition or individual attitude

From the analytical point of view, this is a residual category, explaining the diversity in the adaptation and appropriation strategies that we observed. For instance, in the same company and project we examined quite different modification strategies: e.g. one developer works productively with the so called milestone builds of Eclipse, which was an indicator of his 'early adopter'-attitude. In the same project, another person only uses the official releases ("the real ones"), which was an indicator for his rational attitude.

From the practical point of view this was the most important category, because the organizational responsibility of modifying and maintaining the Eclipse workbench was given to the developers themselves.

- the arena of work practice

The arena of work practice influences the modification of the Eclipse workbench in several ways. For instance, we found agreements on what tools or components to use, depending on what project a developer currently works. Thereby a group of developers, working on a specific project may use the same tools.

Another example is the way a developer instructs a new team member: „If a new colleague starts working here, I would recommend him to begin with the standard IDE, and as a starting point for further exploration, I would show him which extensions - I have integrated in Eclipse - could be interesting for him (based on his experience).“ This citation shows how subtle the balance between autonomy and cooperative integration is tared in practice. The seasoned developer submits an individualized offer, but respecting the autonomy and the tool competence of the other, even if he is a novice.

- the arena of the organization

We found that no formal organisational directives exist, regarding what components to install or how to configure these. The missing directives were an expression of their organisational culture. Based on this culture it was within the responsibility of the software developers to do what is best to get their work done.

- the arena of being a software developer professional

As we mention, being up-to-date has to do with the phenomenon that for software developer professionals it is a value in its own, to be at the cutting edge of technology inventions. Therefore the arena of being a professional developer has it's own laws. In the case of our study, we have seen that the Eclipse culture shapes this arena in a specific way. In particular, we saw that the software developers take the Eclipse rhythm of innovation and its specific release management with milestones, minor and major release building into account. In addition, they use the internet presence of Eclipse (in the form of the major eclipse.org site) to get informed of news related to Eclipse.

*c) Related tasks in beeing up-to-date*

First of all we saw that taking care of the workplace is a complex task. Here we identified at least three different aspects which where related to each other:

- Keeping the workbench up-to-date

This aspect includes all technical issues of installation, updating, configuring and disposing components and component sets.

Eclipse addresses different aspects of workbench modification by different user interfaces. The functionality splits into the "Search for updates ..." dialog for updating existing tools, the "Search for new features ..." dialog to enhance the workbench and the "Product Configuration" dialog to inspect, update or dispose existing features. And a user who wants to inspect his workbench environment at the plug-in level has to use another user interface and if he comes to the conclusion that he wants to dispose a specific plug-in, he has to jump to the file system.

So, even if it makes sense to separate these different aspects analytically, the separation on the user interface level leads to several usability problems. Also the fact, that the user can only inspect his environment at the feature level and not at the plug-in level is problematic if system malfunctions appear.

- Keeping the tool competence up-to-date

Installing a tool is one thing. Being aware of the tool and evaluating it with respect to the own working practices is another. And learning how to use the tool constructively is a third one. At this appropriation level it makes sense to analytically separate the aspects of switching to another version and of integrating a new tool.

The integration of new components or, more specifically, the switch to a new release is motivated by the ambition of being up-to-date. Related to Eclipse, this has two different meanings. In the first place, updating of existing tools was important for the actors. For example the developer in our

study always use the newest version of installed components. Here the meaning of the modification was ‘refreshing tool competence by updating the existing tools’. The integration of special 3<sup>rd</sup> party components into the Eclipse framework to take advantage of new functionality was an example of the second meaning, reflecting the appropriation of new technologies and trends in the software development. Here the modification has the meaning of ‘enhancing tool competency by adding new functionality’.

- Disposing the old workbench.

Form a technical perspective this was a quite surprising observation, because nothing seems to be easier than to delete a file. But from a work practice perspective, the case is not that simple. This is because of two different reasons. The first reason is that some developers are always skeptical about the reliability of the workbench. The other reason is the dynamics of projects. Each project uses it’s own, specific workbench with tools and tool versions. Even if a project is finished, it might happen that the project will be opened again, e.g. to fix a bug.

In Eclipse context, the Configuration Manager (as a part of the org.eclipse.update plug-in family) is responsible for supporting this kind of functionality. But we observed that users developed their own strategies to deal with this problem. This might be an indicator that the functionality and/or the usability of the solution might be insufficient.

#### IV. THE CONCEPT OF APPROPRIATION INFRASTRUCTURE

In the above section we presented our hypothesis that workbench modification is an activity that has to balance out the aspects of integration and stabilization embedded in the dialectical relationship of autonomy and heteronomy. This discussion provides the required background to address the main question of the symposium what shape should the new provisioning technology take and how will you use it to support the activities of appropriation and provisioning in the future.

To address the question of what shape provisioning will take in future, we have first to point out that the workbench preparation is a complex, cooperative task, where different actors (users, power users, administrators, domain developers, etc.) have to communicate and coordinate the appropriation activities. This holds in simple environments of mid-size companies, as described above.

Developers join and leave the projects, each with a different level of commitment to common used infrastructure. Some work close together, but some actors (e.g. domain developers) will never get contact directly with each other. This means that the citation about tailorability in general also holds in the case of Eclipse IDE: *“It is not easy to locate the different accountabilities in the process that lead to a tailoring activity and in the process of the tailoring activity itself. While it is usually clear who actually worked with the tailoring interface of an application, it is not clear whose assumptions, ideas and decisions influenced a new tool configuration.”* [12]. To deal with these problems, a technical infrastructure should be conceptualized with this

wicked structure of workbench preparation and appropriation processes in mind.

To address this issue, our approach is based on the perception that a providing infrastructure should not only deal with the technical aspects of updating components, providing access to a repository of components and profiles or managing this repository. It should also provide communication channels between the different actors. Figure 1 illustrates this model of a shared communication (post-it notes) and appropriation oriented provisioning (pieces of puzzle) infrastructure.

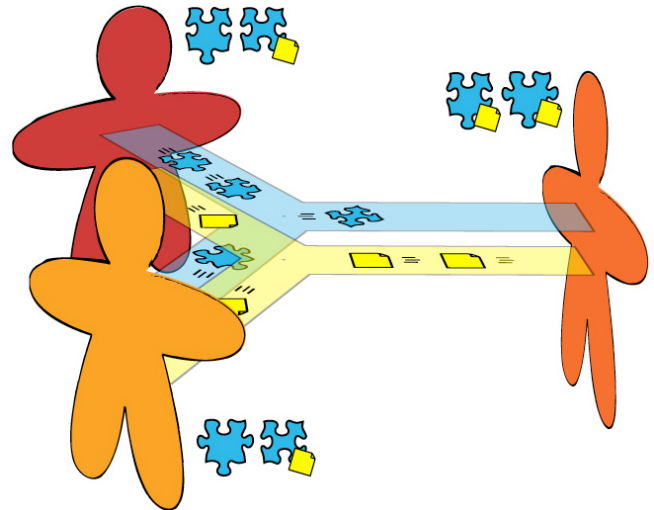


Figure 1 Conceptual perspective on an infrastructure for appropriation support.

The communication channels should address two issues:

- (1) Appropriation processes require knowledge sharing among users. Therefore, communication channels should support communities of users to reflect upon their software usage.
- (2) Support for appropriation processes needs to bridge the formal and the informal level of software provisioning and adaptation. Therefore it is necessary to provide communication channels between the different roles, allowing users to express appropriation requirements towards others e.g. administrators or power users or even domain developers.

Both types of communication channels have to be designed in a specific form, since they bridge among different actors in different situations. The diversity increases in companies with a specialized or outsourced IT department, where the different actors will have rather distinct tasks, interests, and cultures. In particular, this issue will be critical, if we want to think about a solution that can be scaled up to the level of common infrastructure for the whole Eclipse community.

But even when the diversity of perspectives and discourses increases they all deal with the artefact. Several authors suggest that ‘physical’ artefacts can be used as boundary objects [13-15] to mediate and shape the discourse.

Taking a user perspective into account, we will argue that the user interface of an application itself can be seen as a boundary object which can foster knowledge sharing among the different actors involved in the appropriation process.

To act as a boundary object among users, the functions of the application and the modification opportunities need to be understandable for users with different backgrounds of practice and levels of expertise. This means that we have to improve visualisation of the modularisation and dependency structure and tightly connect this visualisation of the software with the related information and discussion spaces. At least the appropriation infrastructure should be better connected with the daily user experience. This implies, that the communication channels among users should be integrated into the user interface and refer to the modularized structure of the functionality and vice versa so that the content of the functionality-related communication among the actors should become visible and easily accessible. The direct activation [16] should support the change from use to the related meta-use functionality of discussing and tailoring the software artefact. Therefore the components, the tailoring artefacts as well as the related discussions should be co-referential [17]. Especially in the case of Eclipse RCP scenarios, where we could not expect that the users are software specialists, this issue becomes a critical demand.

A. An architecture for an appropriation infrastructure

Figure 2 shows an architectural outline to realize the described infrastructure, which provides sharing of knowledge and artefacts between users and administrators.

In the following, we want to take a closer look at the architecture supporting the appropriation work. The architecture is split in two parts (see Figure 2). The middle part describes the shared or backend part of the infrastructure, while the left part describes the integration into the local application context. The right side represents other actors, connected to the appropriation infrastructure. This architecture describes a logical view of the

appropriation infrastructure and does not yet constitute the physical/technical design.

The shared part establishes a common ground that allows the participants to share their knowledge and tailoring artefacts with each other.

1) Discourse infrastructure

The role of the discourse infrastructure is to foster the social process of sense making and negotiation around the used technology. Discourses can be related to the own application usage, intertwined with experiences of other actors or negotiations of common interpretations. It should include the reference to the related components, as well as to other additional components and configurations and patches. The other role is to integrate the discourses into the local context.

The discourse infrastructure is split into two parts: A local and a shared part. The role of the shared part is to store and manage the actors' discourses. The role of the local part is mainly to integrate discourse into the application. This means, to present discourses to the user in an adequate manner in relation to the use context. In addition, it should enable and encourage the users to actively join the discourses.

2) Cooperative Tailoring

The section of cooperative tailoring will be responsible for dealing with the technical aspects of a cooperative provisioning solution. The main role of the shared part is the management of two repositories and the access to them: One repository store and manage ready-made components (e.g. Eclipse plug-ins, but in our case study it was also important to manage third-party non Eclipse components, like Tomcat or MySQL databases). The other repository stores and manages tailoring artefacts.

Typically ready-made components are large, support version management and were usually created by professional software producers. In opposite to this, typical tailoring artefacts are small and are the result of a local

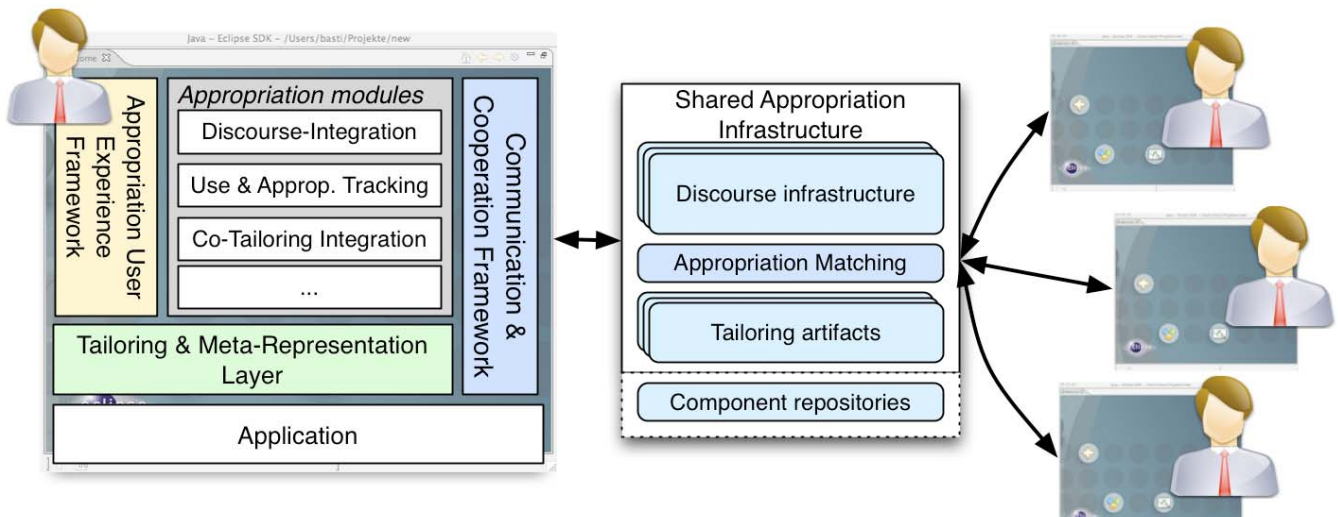


Figure 2 Local architectural perspective on an infrastructure for appropriation support.

tailoring-process. Examples of tailoring artefacts are composition of components (e.g. user defined features or profiles), scripts (e.g. EclipseMonkey scripts) or configuration of specific components (e.g. personalization or 'groupalization'). Among other things, the aim of the infrastructure is to support local tailoring processes, giving others access to the resulting artefacts. This means that the tailoring artefacts can be isolated and have to be shareable. This means that they can be made persistent and describe their dependency in a way that we know from components. Therefore tailoring artefacts can also be described as user generated components, because on one side they emerge out of a local tailoring-process, but on the other side they have to fulfil many requirements we know from the component approach.

The shared infrastructure should be able to store components itself or to reference external repositories e.g. the component suppliers' repositories. A special case is a pure peer-to-peer scenario, where a shared repository is only virtual - provided by a dynamic net of enhanced Eclipse RCPs.

The local part embeds the shared repository into the application context and should support a cooperative tailoring model. This means that it should provide a local access to the shared infrastructure, so that ready-made components and tailoring artefacts can be installed locally. But it should also provide means to share local adaptations with others using the shared part of the infrastructure.

### 3) Appropriation support

Beside the communication support, [18] describe a system that supports appropriation by automatic generated user recommendations. It underlines the importance of analysing individual and collective use histories [19]. Based on analysing these histories, the system recommends extensions and show ways for the user to use them effectively. Through the usage of a collaborative-filtering algorithm, which is chosen specifically for this context, users are supported on finding add-ons and related expertise regarding these add-ons within the cooperative context.

Although today it would be just a nice feature, in a design concept for the future this will become mandatory.

In our architecture we address recommendation features by integrating two components in our architectural concept. The first of them is the use and appropriation tracking component. It is responsible for collecting the use histories and for making them available to the shared infrastructure. In addition, it is responsible to embed use recommendation into the local application in a proper form. The second component is the shared component. Its role is matching elements on the use history, generating recommendations of system adaptations and availability of knowledge in other peers. This can be implemented by using for example collaborative-filtering algorithms.<sup>3</sup>

---

<sup>3</sup> To achieve this goal we must however address following questions: What are the suitable information in the context of Eclipse IDE, how to analyse this information, which are

Beside these different appropriation modules, the architecture contains a common communication layer, a tailoring interaction layer and end user experience framework.

The communication layer manages the communication of local and shared context. The communication data includes semi-structured data (e.g. discussions with other users and developers), as well as formal data (e.g. meta-data to explicate component dependencies). Furthermore it includes exchange of binary data (e.g. whole components) as well as textual data such as xml configuration. In our opinion, the ECF Framework is a good candidate to realize this layer.

The tailoring layer allows accessing and adapting the application's configuration. In this way, new components can be installed and uninstalled, and configurations can be externalized. The opposite direction, the integration of components and configurations by other users, should be also possible. The OSGi framework implemented in the Equinox platform is suitable for the management of the plugins. The realization of the other parts, in particular the aspect of sharing personalization and using groupalization is an open question.

The aim of the appropriation end user experience framework is to guarantee a consistent user experience regardless of the modular composition. The concrete design, in particular the smooth integration of the infrastructure into the use context, is also an open question.

In CoEUD our aim is to create a prototype as a proof-of-concept of these ideas. To achieve this, it is planned to create and evaluate two implementations of the presented logical architecture: a client-server oriented one and a peer-to-peer one.

## V. CONCLUSION

As the term 'provisioning' indicates, usually this issue is mainly addressed from a central perspective (e.g. software producer or IT administrator), neglecting the artful work of professional software developers preparing their workbench, finding, installing and appropriating the right tools in the right situation. Instead of 'provisioning technology' we propose the term 'appropriation infrastructure' to take this artful work into account.

This position paper stresses this perspective, presenting empirical findings about the practice of dealing with the issue of workbench modification. Instead of a coordination-by-centralisation approach, which lies in the core of common provision technologies – like Lotus expeditor, we found that workbench modification practices were mainly driven by a coordination-by-communication approach. Although the informal approach leads to several problems that make centralized provisioning technology attractive for the companies, a provisioning technology must respect the autonomy and the competence of the professionals, if they

---

the optimal ways to inform the user about recommendation, etc. These issues are part of the current research in the context of CoEUD.

want to avoid negative effects on the work practices of professionals. To get a more profound understanding of this work practise, we dissect two principal motives that shape the workbench modification: the antagonism stabilization versus innovation and the dialectical relationship of heteronomy versus autonomy. We believe that every new provisioning technology should prove its own worth by providing an appropriate design in respect to these two motives.

We believe that a suitable solution should take into account the different perspectives and interests involved – end user, administrator, organization, etc. To shape our position and to enlarge the scope of design/thinkable solutions, we focus on the question of how a technical infrastructure should look a like from a work practice perspective.

We have presented a general architecture, given a first tentative answer of how a provisioning infrastructure should be designed. The proposed architecture is based on the concept of boundary objects and was split into a local and a shared part including related activities of adaptation, appropriation and communication.

An unresolved challenge is to provide a seamless transition of the daily use experience to the underlying composition structure and the related communication channels.

From a technical perspective, the appropriation infrastructure and a new technology of provisioning have so many similarities, that a shared implementation effort could be the best way to bring both sides together.

## VI. ACKNOWLEDGEMENTS

The authors would like to thank Bernhard Nett, Leonardo Ramirez, Andrea Bernhards, Sebastian Deneff, Frank Hölting, Hendrik Sander, Volker Wulf, Volkmar Pipek, the Eclipse community and all the anonymous open source developers.

This work has been funded by the Bundesministerium für Bildung und Forschung as part of the project “Component based End User Development” (CoEUD) No.: 01ISF01A-E.

## VII. REFERENCES

1. Poole, M.S. and G. DeSanctis. *Use of Group Decision Support Systems as an appropriation process*. in *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*. 1989: IEEE.
2. Lieberman, H., F. Paternò, and V. Wulf, eds. *End User Development*. 2006, Springer.
3. Mørch, A., et al., *Component-based technologies for end-user development*. *Communication of the ACM*, 2004. **47**(9): p. 59-62.
4. Stiemerling, O., *Component-based Tailorability*. 2000, University of Bonn.
5. Urban, G.L. and E. von Hippel, *Lead User Analyses for the Development of New Industrial Products*. *Management Science*, 1988. **34**(5): p. 569-82.

6. Pipek, V., *From tailoring to appropriation support: Negotiating groupware usage*. 2005, University of Oulu: Oulu.
7. BMBF, ed. *Analyse und Evaluation der Softwareentwicklung in Deutschland. Endbericht an das Bundesministerium für Bildung und Forschung*. 2000, GfK Marktforschung: Nürnberg.
8. Nett, B. and G. Stevens, *Business Ethnography*. 2007.
9. Nett, B. and G. Stevens, *Technikgestaltende Aktionsforschung: Forschung als Partizipation*, in *Die Virtualisierung der Arbeit - Zur Ethnographie neuer Arbeits- und Organisationsformen*, G. Hirschfelder and B. Huber, Editors. 2004, campus. p. 445-470.
10. Strauss, A.L., *Analysis through Microscopic Examination*. *sozialer sinn: Zeitschrift für hermeneutische Sozialforschung*, 2004. **2**: p. 169-176.
11. Oevermann, U., *Die Methode der Fallrekonstruktion in der Grundlagenforschung sowie der klinischen und pädagogischen Praxis*, in *Die Fallrekonstruktion*, K. Kraimer, Editor. 2000, Suhrkamp: Frankfurt a.M. p. 58-153.
12. Pipek, V. and H. Kahler, *Tailoring together: A systematization and two cases*. *international reports on socio-informatics (IRSI)*, 2004. **1**(2): p. 3-48.
13. Star, S.L. and J.R. Griesemer, *Institutional Ecology, Translations and Boundary Objects - Amateurs and Professionals in Berkeleys-Museum-of-Vertebrate-Zoology, 1907-39*. *Social Studies of Science*, 1989. **19**: p. 387-420.
14. Subrahmanian, E., et al., *Boundary Objects and Prototypes at the Interfaces of Engineering Design*. *Computer Supported Cooperative Work*, 2003. **12** p. 185-203.
15. Mørch, A. and N.D. Mehandjiev, *Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units*. *Computer Supported Cooperative Work*, 2000. **9**(1): p. 75-100.
16. Wulf, V. and B. Golombek. *Exploration environments: concept and empirical evaluation*. in *Proc. of GROUP'01*. 2001.
17. de Souza, C.S., S.D.J. Barbosa, and S.R.P. Silva, *Semiotic engineering principles for evaluating end-user programming environments*. *Interacting with Computers*, 2001. **13**: p. 467-495.
18. Bell, M., et al. *Domino: Exploring Mobile Collaborative Software Adaptation*. in *Proc of. Pervasive 2006*. 2006.
19. Chalmers, M., *A Historical View of Context*. *Computer Supported Cooperative Work*, 2004. **13**(3): p. 223-247.