Software Architecture as Ground for Exercising Decision-making Power in Distributed Software Development: A Case of DHIS Tracker

Tiwonge Davis Manda
Department of Informatics
University of Oslo
tiwongem@ifi.uio.no

Marlen Stacey Chawani
Department of Informatics
University of Oslo
marlensg@ifi.uio.no

Caroline Ngoma
Department of Informatics
University of Oslo
carolng@ifi.uio.no

ABSTRACT

Extant literature suggests that the exercise of decision-making power within software projects is influenced by software architecture and control over: source code; implementation of consensus-based decisions; scope definition, and enrolment of participants. From this list, we argue that software architecture is the most dominant factor. A software artefact's openness to extension, outside the involvement of core software development teams, has a bearing on the extent to which design decision-making power is concentrated within software development teams. To avoid such an occurrence, we argue for a progression towards platform-based software ecosystems, to enable stakeholders take charge of the development of software capabilities that suit their needs.

Author Keywords

Decision-making; distributed software development; platform-based software ecosystems; software architecture

INTRODUCTION

Advancements in communication technologies, coupled with the need to leverage expertise that is not locally available, provide an imperative for distributed software development (DSD). Among other things, DSD allows for a spread in software development best practices across contexts (Braa et al., 2004; Titlestad et al., 2009; Braa and Sahay, 2012: pp. 240). DSS also affords cross-context collaboration, in the development of generic software products that can support common use cases across contexts (Staring and Titlestad, 2008; Titlestad et al., 2009). Despite such benefits, participation in DSD can be quite challenging. For example, the size of software development projects, such as the one we reflect upon herein, significantly constrains possibilities for participation. Even where there are active efforts to promote participation; the sharing of power in PD still requires negotiation of competing values of utility and efficiency (Asaro, 2000; Bratteteig and Wagner, 2012).

Literature suggests that participation in decision-making is in part influenced by software architecture (Benkler, 2002; Staring and Titlestad, 2008) and organizational structures such as the distribution of roles. Roles have a bearing on activities one is involved in, as well as their proximity to the centre of decision making (Barcellini et al., 2009). Members of core software development teams are likely to retain control over key aspects to the exercise of decision-making power: *source code* (Mahendran, 2002; Sack et al., 2006); *implementation of consensus-based decisions* (Barcellini et al., 2009; Bratteteig and Wagner, 2012); *definition of what problems and solutions are deemed relevant (scope definition),* and *enrolment of participants* (Bratteteig and Wagner, 2012). A closer look at these factors suggests the centrality of software artefacts in the decision-making process. Decisions related to the aforementioned factors are made with the software artefact in mind. In addition, the nature of a software artefact has a bearing on the democratization of design-related decision-making. Restrictive software designs are likely to consolidate the role of core team members, whereas more open and modular software architectures promote decentralised development of extensions by different people, to suit their needs (Benkler, 2002; Staring and Titlestad, 2008; Lawson et al., 2009; Tiwana et al., 2010; Wubishet et al., 2013).

In this paper, we reflect on how software architecture evokes or precludes design choices, for stakeholders outside core software development teams, regarding the uptake of context-specific requirements. Empirically, we draw upon our involvement within the district health information software (DHIS) project, which is present in at least 46 countries, across four continents. The size of the DHIS project space means a majority of stakeholders, outside the core software development team, have little influence on design decision-making. At the same time, use cases supported by DHIS software are diverse; context-specific software requirements are sometimes in conflict; and locally critical requirements might not be supported at global level. Negotiation of these dynamics requires significant participation from users, who are domain-experts, as well as their representatives, in order to develop locally relevant solutions.

After noting difficulties with relying on one central node to support such a diverse community, DHIS software development has taken a turn towards platform-based software ecosystems, in the likeness of Android operating system and Apple's IOs, as a way of enhancing local participation, whilst retaining the benefits associated with DSD. Platform-based software ecosystems leverage the expertise of a diverse developer community, with better appreciation of user needs, thereby affording development of new capabilities unforeseeable by the platform's original designers (Tiwana et al., 2010).

Based on noted challenges and recent developments within DHIS software development, we argue that PD needs to seriously consider the progression towards software platforms offerings and its influence on the exercise of design decision-making power, within DSD.

METHODOLOGY

The paper provides a reflective analysis of our involvement, as action researchers, software implementers, as well as mediators between local users and global teams of software developers. We were involved in development and customization efforts for DHIS Tracker solutions, to support antenatal care services delivery in Malawi and Tanzania. DHIS Tracker is a generic software solution for collecting, managing and analysing transactional, case-based data records. The software allows health personnel to monitor longitudinal access to health services. DHSI Tracker is a module within the DHIS 2 application suite, a web-based solution for collecting, managing and analysing health data. DHIS 2 development is coordinated by the Health Information Systems Programme at the University of Oslo (HISP Oslo), with some development work done in Vietnam.

Reflective analysis affords questioning and articulating tacit assumptions associated with playing particular roles, as well as formulation of strategies for handling uncertainties faced (Bjørn and Boulus, 2011). The experiences presented herein extend over the period July 2010 to July 2014. The first two authors led DHIS Tracker customization efforts in Malawi, whereas the third author was responsible for efforts in Tanzania. Our personal experiences are supplemented with email communication on DHIS software development and an interview with a core DHIS software developer.

EMPIRICAL INSIGHTS

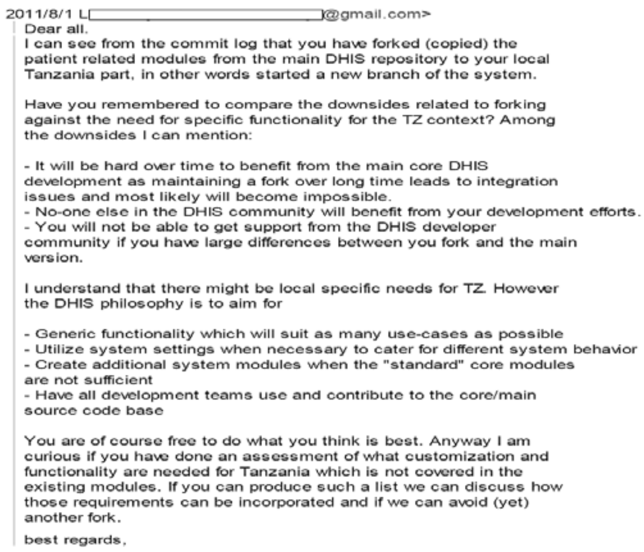
In the empirical setup within which we operated, end-users and developers never had direct contact, meaning realization of some local requirements was dependent on our effectiveness in pushing through such, at global level.

Software requirements from Malawi were collected by the first two authors, first through a situation analysis conducted in July 2010, and follow up consultations with target end-users (health workers). During software customization, the software was demonstrated to, and tested by, target end-users, to evaluate interface designs. After customization, end-users were given training on basic computer literacy and DHIS Tracker. During the trainings, input was solicited regarding interface designs. Further software customization followed thereafter. After implementation end-users were provided with support, followed by an evaluation of the system in use.

Requirements for Tanzania were gathered through a situation analysis done by the third author. Meetings with target end-users, as well as training on DHIS Tracker provided a platform for deliberating and concretizing requirements. Use of the software after implementation resulted in further design requirements. Although users in both Malawi and Tanzania were consulted, the realization of some of their requirements was dependent on processes at global level. Such was the case where actualization of requirements required modifications to the DHIS Tracker software core.

At global level, the DHIS project space is organized around roles such as: software developers, implementers, documenters, and coordinators. Although individuals can play different roles, certain roles are personalised, thereby segregating the activities people may get involved in. While adaptation of DHIS software to fit local requirements is often done by implementers, coordination of software requirements gathering, prioritization, and assignment of tasks to developers, is the responsibility of coordinators. Consequently, in assuming the role of implementers, we lacked necessary decision-making power to push through critical context specific requirements, for inclusion in the global software core. In both Malawi and Tanzania we implemented partially developed DHIS Tracker solutions, which lacked required functionality for generating aggregate reports.

The consequence of such inability to push through context-specific requirements was quite profound in that DHIS Tracker software architecture severely restricted possibilities for extending software, outside the global core, controlled by HISP Oslo. During the development of DHIS Tracker solutions for the two countries, which lasted until March 2012, for Tanzania, and November 2013, for Malawi, DHIS Tracker had a weak application programming interface. There were also compatibility issues across software versions, making it difficult for implementers to change software versions. Such underdevelopment of DHIS tracker API meant a majority of extensions to the software core, needed to be made under the guidance of HISP Oslo. In the end it was both challenging to be part of the global software development community, as well as attend to locally critical software requirements. The email extracts below depict a conversation by a team in Tanzania and a developers from HISP Oslo, after Tanzania attempted to create a parallel DHIS repository, also termed forking (Stalder and Hirsh, 2002), so they could have increased influence in the implementation of locally critical functionality.



2011/8/1 L[redacted]@gmail.com>

Dear all.
I can see from the commit log that you have forked (copied) the patient related modules from the main DHIS repository to your local Tanzania part, in other words started a new branch of the system.

Have you remembered to compare the downsides related to forking against the need for specific functionality for the TZ context? Among the downsides I can mention:

- It will be hard over time to benefit from the main core DHIS development as maintaining a fork over long time leads to integration issues and most likely will become impossible.
- No-one else in the DHIS community will benefit from your development efforts.
- You will not be able to get support from the DHIS developer community if you have large differences between you fork and the main version.

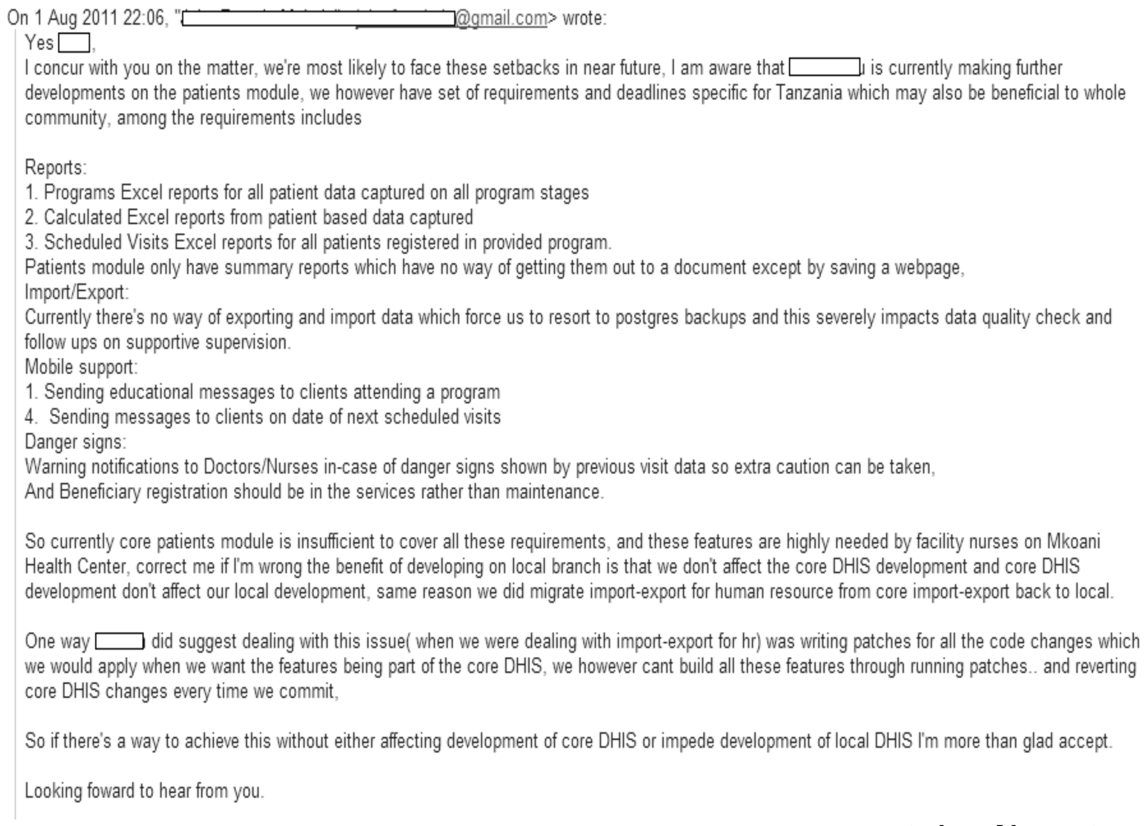
I understand that there might be local specific needs for TZ. However the DHIS philosophy is to aim for

- Generic functionality which will suit as many use-cases as possible
- Utilize system settings when necessary to cater for different system behavior
- Create additional system modules when the "standard" core modules are not sufficient
- Have all development teams use and contribute to the core/main source code base

You are of course free to do what you think is best. Anyway I am curious if you have done an assessment of what customization and functionality are needed for Tanzania which is not covered in the existing modules. If you can produce such a list we can discuss how those requirements can be incorporated and if we can avoid (yet) another fork.

best regards,

Figure 1: Forking of DHIS repository



On 1 Aug 2011 22:06, "[redacted]@gmail.com> wrote:

Yes [redacted],

I concur with you on the matter, we're most likely to face these setbacks in near future, I am aware that [redacted] is currently making further developments on the patients module, we however have set of requirements and deadlines specific for Tanzania which may also be beneficial to whole community, among the requirements includes

Reports:
1. Programs Excel reports for all patient data captured on all program stages
2. Calculated Excel reports from patient based data captured
3. Scheduled Visits Excel reports for all patients registered in provided program.
Patients module only have summary reports which have no way of getting them out to a document except by saving a webpage,
Import/Export:
Currently there's no way of exporting and import data which force us to resort to postgres backups and this severely impacts data quality check and follow ups on supportive supervision.
Mobile support:
1. Sending educational messages to clients attending a program
4. Sending messages to clients on date of next scheduled visits
Danger signs:
Warning notifications to Doctors/Nurses in-case of danger signs shown by previous visit data so extra caution can be taken,
And Beneficiary registration should be in the services rather than maintenance.

So currently core patients module is insufficient to cover all these requirements, and these features are highly needed by facility nurses on Mkoani Health Center, correct me if I'm wrong the benefit of developing on local branch is that we don't affect the core DHIS development and core DHIS development don't affect our local development, same reason we did migrate import-export for human resource from core import-export back to local.

One way [redacted] did suggest dealing with this issue(when we were dealing with import-export for hr) was writing patches for all the code changes which we would apply when we want the features being part of the core DHIS, we however cant build all these features through running patches.. and reverting core DHIS changes every time we commit,

So if there's a way to achieve this without either affecting development of core DHIS or impede development of local DHIS I'm more than glad accept.

Looking foward to hear from you.

Figure 2: Explaining forking of DHIS repository

To open up the DHIS Tracker development space, there has been a shift towards a platform-based software ecosystem. For example, DHIS version 2.13, released in September 2013, introduced an app store, which affords third party software developed using HTML and JavaScript to be installed inside DHIS2 software. Software developers have also been actively working towards application programming interfaces (APIs) for extending all modules of the DHIS tool suite, including DHIS Tracker, to allow for third party software development. Following is an interview the first author (labelled A1) had with a core software developer (labelled D), regarding this shift:

A1: Am interested in finding out the reason behind the shift from the more generic development, where people can just customize [the software you provide] to the app space; if I may call it that

D: By going to the app space we are not losing the generic aspect of it, so that part is still there. Because what we do with the app is...we are in a way exposing what the user has done. In DHIS there are two parts, one is the maintenance part and the other is the services part. So normally what you do in the maintenance part [is]: you define everything (all the meta-data, all the programmes) and then you access it in the services part. So we are doing the app on the services part

A1 Ok

D: Whatever you define here, on the maintenance part, comes in here [the service part]. That's not to say [that] by going the app [way] we are not using the generic functionality. But, then, at the same time, of course, the idea for going the app way is: doing everything with the small community [of core developers] we have in here [referring to DHIS offices in Oslo] is too much

A1: So what necessarily does the app space add in this case?

D: It's two things. One is: technically, the system we had is not sustainable, because the technology we are using is quite heavy [regarding programming competence] and, then, of course, it's being phased out and new trends are coming; the app. The other [dis]advantage is that it is too tightly coupled. Everything is bundled together. If you want to have other parts being involved in that development space, there is little space [to accommodate others]. By externalizing, going the app way, we are in a way giving the others the space so that they can develop their own apps…

A1: So what do you anticipate, going forward, in terms of involvement from elsewhere?

D: By going the app [way] the advantage is now we are dividing the [development/design] space into two, if you like. Somebody takes care of the core, which will be the team in here... we are going to handle whatever the users define, still maintain all the flexibility we have, but then how you interpret, how you display, how you do recording [of data] (in what form, in what layout) we are not going to be bothered by that. That's too [context] specific. And, then, at the same time, we can't say: this is the layout we have [and] this is the data entry form we support. We can't say, for

> some users, stick with that. They might have something different. Then, if that's the case, we provide them the web API, so that they can design themselves. So in a way we are dividing the whole [development/design] space into two, and then we take care of the first part and then the remaining part they are going to fill in. That's where the app comes [in].

The transcript depicts a realization of the diversity of use cases supported by DHIS software and the need to afford user communities an opportunity to take charge in defining context-specific solutions. The developer seems to suggest that moving towards a platform-based software ecosystem decouples software development at local implementation level, from that at global level. Similar sentiments have been suggested regarding DHIS Tracker:

"During the next three months we will focus a log on creating an SDK [software development kit] for Tracker on Android, allowing anyone to create custom apps without having to implement the hard parts" (Software development coordinator, July, 2014)

CONCLUSIONS

The empirical case presented herein suggests that members of core software development teams retain significant control over: source code; implementation of consensus-based decisions; scope definition, and enrolment of participants. However, the extent to which decision-making power is concentrated within core software development teams is significantly shaped by the extent to which the architecture of the software under development affords extensions, outside the involvement of core development teams. For example, where software has poorly developed application programming interfaces, significant extensions will need to be made to the software core under the stewardship of core software developers. Such an occurrence is bound to negatively influence engagement with users at local use level, especially where their requirements are not taken on board. Beyond this, our findings suggest that a progression towards platform-base software ecosystems may open up the design space, allowing local software communities to have greater influence in the uptake of local requirement, whilst benefiting from distributed software development efforts.

REFERENCES

Asaro, P. M. (2000). Transforming Society by Transforming Technology: The Science and Politics of Participatory Design. Accounting, Management and Information Technologies, 10, 257-290.

Barcellini, F., Détienne, F. & Burkhardt, J. M. (2009). Participation in Online Interaction Spaces: Design-Use Mediation in an Open Source Software Community. International Journal of Industrial Ergonomics, 39, 533-540.

Benkler, Y. (2002). Coase's Penguin, or, Linux and "the Nature of the Firm". The Yale Law Journal, 112, 369-446.

Bjørn, P. & Boulus, N. (2011). Dissenting in Reflective Conversations: Critical Components of Doing Action Research. Action Research, 9, 282-302.

Braa, J. & Sahay, S. (2012). Health Information Systems Programme: Participatory Design within the Hisp Network. In: SIMONSEN, J. & ROBERTSON, T. (eds.) Routledge International Handbook of Participatory Design. Routledge

Braa, J., Monteiro, E. & Sahay, S. (2004). Networks of Action: Sustainable Health Information Systems across Developing Countries. MIS Q., 28, 337-362.

Bratteteig, T. & Wagner, I. (2012). Disentangling Power and Decision-Making in Participatory Design. Proceedings of the 12th Participatory Design Conference: Research Papers - Volume 1. Roskilde, Denmark: ACM.

Lawson, J.-Y. L., Al-Akkad, A.-A., Vanderdonckt, J. & Macq, B. (2009). An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-the-Shelf Heterogeneous Components. Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, 2009. ACM, 245-254.

Mahendran, D. (2002). Serpents and Primitives: An Ethnographic Excursion into an Open Source Community. Unpublished Masters thesis, University of California, Berkeley, Berkeley, CA.

Sack, W., Détienne, F., Ducheneaut, N., Burkhardt, J.-M., Mahendran, D. & Barcellini, F. (2006). A Methodological Framework for Socio-Cognitive Analyses of Collaborative Design of Open Source Software. Computer Supported Cooperative Work (CSCW), 15, 229-250.

Stalder, F. & Hirsh, J. (2002). Open Source Intelligence. First Monday, 7.

Staring, K. & Titlestad, O. H. (2008). Development as a Free Software: Extending Commons Based Peer Production to the South. International Conference on Information Systems (ICIS), 2008 Paris, France.

Titlestad, O. H., Staring, K. & Braa, J. (2009). Distributed Development to Enable User Participation: Multilevel Design in the Hisp Network. The Scandinavian Journal of Information Systems, 21, 27-50.

Tiwana, A., Konsynski, B. & Bush, A. A. (2010). Research Commentary---Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Info. Sys. Research, 21, 675-687.

Wubishet, Z. S., Bygstad, B. & Tsiavos, P. (2013). A Participation Paradox: Seeking the Missing Link between Free/Open Source Software and Participatory Design. Journal of Advances in Information Technology, 4, 181-193.