Evolution of Blaise Survey Development in Statistics Finland

Joonas Salmi, Pyry Keinonen and Petri Godenhjelm Statistics Finland

1. Abstract

This paper discusses how Blaise Survey Development is organized in Statistics Finland and how the Survey Development process has evolved during Blaise5 implementation from 2017 to 2019. Statistics Finland has implemented Data Collection Management System which has influenced and set requirements for Survey Development. Data Collection Management System has been in production since January 2019.

This transition included change in work methods and culture. The renewed ways to work have caused a need to recreate the in-house survey development process and recognize the roles needed in daily basis work. Instead of starting Survey Development from scratch and creating customized surveys on demand, the focus was to find new ways to reduce the amount of effort by standardizing. The aim was to streamline Blaise Survey Development.

Standardization has helped both the transition process from Blaise4 to Blaise5 and creating new surveys from scratch. This meant in practice the implementation of the use of version control, standardized survey-independent-layout, and standardized survey-independent base code as a part of survey development. This way we have been able to harmonize the development phases of Blaise5 questionnaires and reduce the effects caused by person dependent practices.

The next step is to implement Agile methods in survey development starting from 2020. This step includes the processes and implementation how to lead and organize our Survey Development and work resources as a whole. The goal is to reduce wasted effort and make resources available efficiently when needed.

2. Background

Transition from Blaise 4 to Blaise 5 led to rethink the survey development process and practices. At the same time there was a transition to new generation of Blaise survey developers. With new generation of developers and new software, it was a convenient time to reform the whole survey development process.

The need for the reform of the survey development process was due to the fact that Blaise survey development got too person specific and the new data collection management system set technical requirements for Blaise surveys. On the old production environment Blaise 4 surveys were prepared to a network drive and surveys were run on local Blaise installations, whereas on the new production environment Blaise 5 survey packages are deployed to our new data collection management system called Ruuti.

Previously Blaise developers had their own Blaise surveys to develop and they were mostly stored on local or network drives. These practices showed as weak source code collaboration and created a lot of individual customized solutions. In the worst-case scenario everyone developed with their own survey-specific Blaise Resource Database file (.blrd), which meant that they weren't compatible with each other. This also fragmented the layout development and created major problems to managing it. Different surveys might have had functionalities, which weren't available in other surveys.

Developing without a uniform mode of work made Blaise survey development fragmented and project deadlines harder to reach. The experiences gained was the basis for the reforming of the survey development process. The aim was to streamline survey development process and make it easier to manage. Utilizing Git version control system to better handle source code collaboration and versioning was the backbone for the success of the survey development reform.

2.1 Process' progression & roles

Work roles and work management has stayed the same, while Blaise survey development has changed. This prompted to rethink the way how surveys should be developed to best fulfill the needs of the Statistics responsible for the data collection.

Prior survey development process helped to form core guidelines for renewed Blaise survey development process and different stages of work. These guidelines define all the different roles and their responsibilities associated with Blaise survey development. Different roles and their responsibilities aren't specified based on organizational titles but rather by real work tasks. These guidelines aim to make Blaise survey development processes clear to the whole organization.

Blaise survey development is starting to resemble more software development. Some of the projects have started to adapt more agile way of work. It started by working through different cycles or iterations. These iterations consist of updating the survey with the changes and then testing the survey. Defining clear iteration steps makes survey development more efficient by concentrating the work focus to one iteration step at a time with all relevant experts included. Changing the work culture and way of work has been a long process and it's still ongoing.

3. In practice

Improving Blaise survey development meant that not only the ways to work was changed but also the standardization in programming was implemented. In practice, the organization started the reform with the rational decision to standardize Blaise Resource Database file and layout. Using only one resource file across all surveys ensures that every survey has the same functionality available. Introducing version control and Ruuti pushed for standardizing base solution structure.

The base solution structure is the same in every survey. It includes always the main survey data model project as well as pre-fill data model project and a manipula project for pre-filling a Blaise Database with test cases. Also, the standardized projects that contains classification tables for lookups are included if needed.

Ruuti handles multi-mode surveys and all the related tasks. It utilizes Blaise as a data collection instrument. After Survey packages are deployed to Ruuti, they are automatically downloaded and installed to interviewers' personal workstations. Ruuti system sets also requirements for the survey project. These requirements are unified data model with standardized Blaise Settings and ontology variables for passing data from Ruuti System to Blaise and vice versa. In practice the survey is divided into sections by question themes and every section is programmed in their corresponding blocks in separate Blaise Include files alias .incx. This hierarchical structure makes the survey data model file alias .blax easier to maintain and the code more comprehensible.

Survey development is divided to three categories based on the complexity level: basic, advanced and master. Developer on basic level can only fix question texts. This is usually done on a text editor instead of using the combination of Blaise and Git version control. Advanced developer can program the whole survey, if there aren't complicated structures. Master level developer is needed only for the most complex survey structures.

Developers in IT-department have the core competence to run Blaise in our architecture. Anyone, who has the skills in our organization, can program surveys in basic and advanced levels. This is encouraged to keep resources available for more demanding tasks. Blaise team handles also the survey installation on test and production environment.

3.1 Survey development process

Developing a survey starts with a new assignment. If it is a completely new survey, a new repository is created in version control with core survey solution. This is master level responsibility in Statistics Finland. Advanced or master level programmers then may program the survey data model and rules and commit their work to version control.

After the survey or a part of the survey is programmed it can be deployed to the test environment when needed. Deployment starts a test cycle where the programmed sections or the whole survey is tested and returned to programmers, if changes are needed. Any project member at the basic Blaise programming level can fix question text errors, when they are spotted while testing. All changes that require programming are distributed to more advanced programmers according to the challenge.

Thus, survey development is done in continuous cycles. After a cycle the changes are programmed, and the next cycle begins and new assignments such as new sections are programmed and issues from previous cycle are fixed. Survey development process is visualized in figure 1.

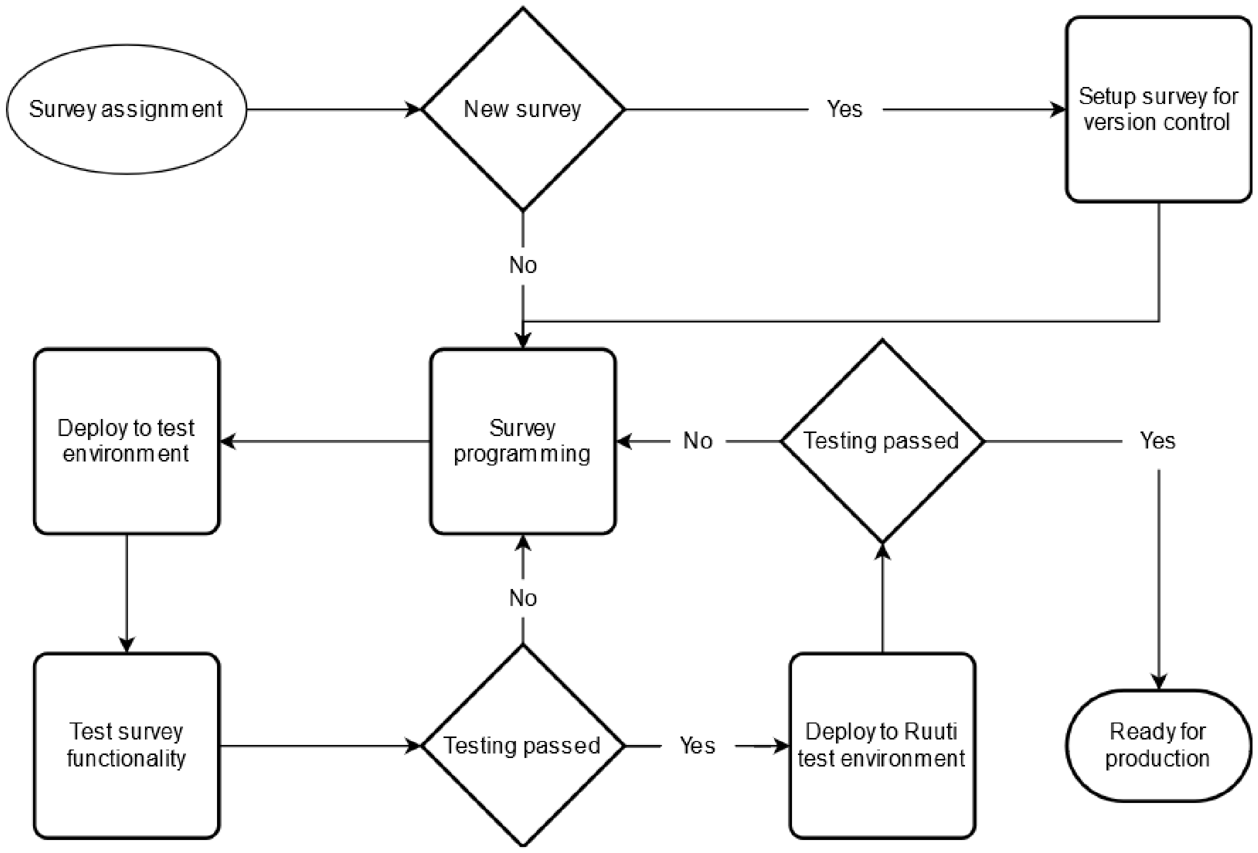
Survey assignment
New survey
Yes
Setup survey for version control
No
Deploy to test environment
Survey programming
No
Testing passed
Yes
No
Test survey functionality
Testing passed
Yes
Deploy to Ruuti test environment
Ready for production

Figure 1: Survey development process.

When the survey development is in its final stages, the survey is deployed to Ruuti test environment. There it undergoes technical testing to make sure Ruuti system handles the survey correctly. Content and functionality are tested outside Ruuti environment. Ruuti tests are done to ensure survey compatibility with Ruuti.

After all the testing is done, the survey is deployed to Ruuti production environment. From there the survey is available to be downloaded to interviewer's workstation for installation. Respondent samples, timeframes and other specifications are done separately from the survey package. If the survey is in production, and there is an issue to fix the survey can be updated. Ruuti handles incompatible data model in the middle of a data collecting period, if all the previous fields still exist in the new survey version. Field numbers can be increased, and checks changed. However, between the data collecting period, the data model can be changed entirely. New published survey version is automatically downloaded and installed to interviewers' workstations.

3.2 Implementation of version control

After implementation of version control, collaborating on Blaise survey source code has become effortless. Instead of taking turns to work on different parts of the source code, it can be worked on simultaneously. Version control takes care of all the changes in the same place and version history readily accessible. The only exception to this is the Blaise resource file. Version control can't keep track of changes of a binary file.

Implementation of version control started with Microsoft Team Foundation Server using Git version control system. Statistics Finland has recently switched to a cloud-based Microsoft Azure DevOps. Because transition to new way of work is still ongoing, the new Azure DevOps features aren't yet fully implemented to daily work. At the moment the use of backlogs and management features is only at the design stage.

Git version control system keeps track of different survey versions. There can be more than one different version of the same survey running at the same time with different data models. The right version is easy to check out and work on. This makes implementing bug fixes easy to different versions of Blaise surveys, even in the middle of data collecting period.

3.3 Standardized practices

Blaise survey solution is standardized to allow straightforward integration with data collection management system (Ruuti). All Blaise surveys use a similar (.blax) source file. The source file defines modes, languages, roles, special answers, attributes and the necessary fields to fill out respondent and survey info. Ruuti sets requirements to fields that must be defined to ensure compatibility.

With the technical advances, the survey layout wasn't up to date anymore and it needed an upgrade. For a better usability experience, the mobile first principle was chosen while designing the new layout for self-administered surveys. The layout design was developed in collaboration with the cognitive laboratory, which is our questionnaire design and pre-testing team, to ensure great user experience. The new layout is standardized, and all survey specific texts are assigned to variables on the contrary to the old layout. This leads to better and easier maintenance when every survey uses the same layout. This ensures that all the Blaise surveys have similar structure that the interviewers are familiar with.

4. Conclusions & Future

The current way of work is a great improvement on the old way of work. Still, there are improvements to be made. There are multiple projects running simultaneously and allocating resources to the right projects at the right time has proven to be difficult. Last minute minor tweaks are still being done right until the beginning of the data collection period, although developer resources would be better used of on another survey. Survey development would benefit from system to allocate the resources where they are needed the most.

4.1 Agile practices

Programming in most survey development projects started to follow continuous cycle programming style. This makes it possible to adapt to a more agile survey development style. The implementation of agile survey development has started and is already in effect on some survey development projects. The agile way aims to resolve difficulties with developer resource allocation and with simultaneous work assignments. Instead of working on simultaneous surveys, the survey developer can focus on one survey at a time.

The switch to agile methods leads to defining new roles and responsibilities. Every survey has a product owner, who has the best knowledge of the substances of the statistics responsible for the survey. Product owners had to be trained to the new role and their responsibilities. Survey development is organized to one-week sprints. Product owners define, with core Blaise survey developers, what are worked on during the sprint and then propose a time when the sprint is held. Ultimately the steering group organizes the sprints and gives them a priority. This ensures that less important tweaks aren’t prioritized to the same level as more major updates anymore.

Transitioning Blaise survey development to agile survey development is a bigger change to way of work than what was done recently. The process is developing, and it is more than likely to differ in the future from the one that is being implemented now.