Lähde: information system for maintaining up-to-date information on forest resources in Finland

A Jolma

Smart Forestry, AFRY Management Consulting, Vantaa, Finland
Email: ari.jolma@afry.com

Abstract:Lähde is a core component in the information system for managing forest and related environmental information in the Finnish Forest Centre. Its development commenced on 2020, the initial product went into production on 2023 and it is currently actively being developed further to handle new kinds of forest use notifications and new forestry practices such as continuous cover silviculture. Lähde is the go-to source of information about the current and future state of forests and forest operation plans and suggestions in Finland. Forests are very important to Finland and to the Finnish people providing livelihood and recreational value besides having an intrinsic environmental value. Lähde replaces an older system and it is a complete rewrite. Lähde is implemented in the Amazon Cloud using free and open source software; especially the GDAL and GEOS based geospatial software stack has an important role due to the nature of the data. Key features of Lähde are its high degree of automation to keep the data up-to-date, and the removal of hard linking between environmental information and forest resource information, which allows better ways to model the interactions between them.

An overview of the developed system is depicted in Figure 1. From the point of view of the client, the Forest Centre, the system implements or aids in workflows, which are fully automatic, actions carried out with the application, or long-running tasks requiring more complex activities. The functionality required by the workflows is based on processes, which manage the forest and environmental features based on inputs. An example of a fully automatic workflow is the import of changes in land parcel network to the data. Import of new forest inventory data, which can cover hundreds of square-km, into the system is an example of complex workflow. In both cases the processes need to be orchestrated to maintain the integrity of the data, and to prevent system congestion. Developing a system that manages the great variety in the sizes of the features and processes using the available cloud components optimally was the greatest challenge in the development.

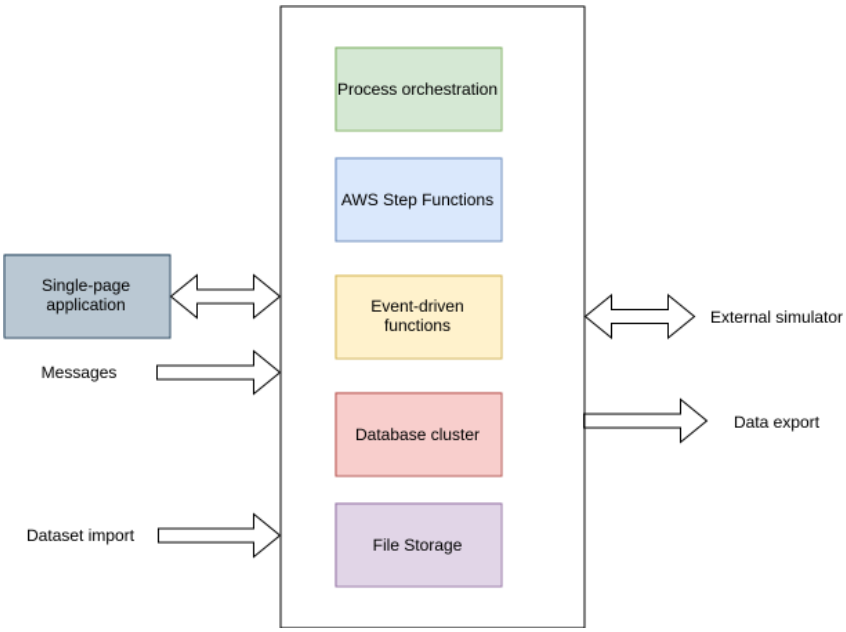

Figure 1: Overview of the Lähde system.

Keywords:*Forest resources, cloud computing, middleware*

1. INTRODUCTION

In 2019 the then new director of development of the Finnish Forest Centre commenced a project to renew the information systems of the Centre. A key component in the network of the systems in the Centre is the forest resource and environmental information system, which hosts the forest inventory, forest operations, nowcasted and projected state of the forest, and related environmental information of whole mainland Finland. The bid to build a new such system was won in 2020 by a consortium of two small companies: Bitcomp and Simosol (Ahokas 2024). Those two small companies no longer exist as they have been merged into two larger companies: Sitowise and AFRY, respectively. The system went into production in 2023 under the name Lähde (”natural well” or ”source” in English) and it is still actively developed further (Lähde is at version 1.2.4 at the time of writing).

Forests are important to people for multiple reasons: they have a spiritual value, they are a source of recreation, and they provide livelihood in the form of food, jobs and money. In addition forests are important habitats and have an intrinsic environmental value. All these are reflected in the legislation, which in the case of Finland, a densely forested country, is extensive. Forest resources change constantly due to operations, growth, and various weather and environmental factors, for example pest conditions. In Finland forest ownership is diversified and private ownership is important: the population of the country is 5.6 million and there are 0.6 million forest owners. The average size of a forest property is 31 hectares. Forests are owned by state, companies, parishes, individuals, groups of individuals, and estates of individuals (Forest Centre, 2025).

The Finnish Forest Centre[1] is a state-funded organization (formally a limited company) tasked to advise forest owners in forest and related ecosystem management and maintain information about forests in Finland. It has also some duties related to forestry legislation enforcement. In its roles the Centre receives a large number and variety of messages from forest owners, agents, and authorities. These are often XML messages (Sitowise, 2025) and are either legally required or voluntary. The Centre also distributes forest information as open data. Another means offered by the Centre to forest owners is a website (metsaan.fi), which can be used to view, upload, and download forest data, information, and messages. The Centre also acquires forest inventory data whose production is outsourced to external companies which produce it from remote sensing, mainly lidar, data.

Cloud computing has become popular throughout the 2000s as a platform for developing information systems (IS). ’Serverless’ is a term marketed by Amazon Web Services (AWS) to denote a way to think about and develop IS components not as programs running in servers but as functionalities implemented in the cloud. The AWS Lambda is a pivotal serverless concept: a small program or a unit of software logic defined in program code running in a machine provided by the cloud as instances. While the general promise of cloud is almost limitless computing and storage for data, the limits of a single lambda are very real, most notably their limited running time and limited memory.

Software programming and information system development should aim for working, valuable, understandable, flexible, and maintainable results. Design principles and paradigms, as employed both in programming and development methodology, should help to reach the aims. In recent years agile methods have become dominant in information technology (IT) industry. Agile methods focus on customer value, human factors and fast, cyclic development. The fundamental setting in any IS development is that there is a customer and his/her requirements and wishes, and the developer. Agile methods aim to improve the development by enhancing collaboration between customer and developer, and by having a trust in individuals.

This paper describes the development of the Lähde system: the requirements, the data model, the technology, and the development methodology. The initial impetus for the development of the Lähde system in the Forest Centre was to introduce automated workflows into processing the various external changes and information into the forest and environmental data base. Another driver for the new development was to separate the environmental, habitat data from the forest data as an independent set of features. The Centre was also in the process of introducing agile development methods into its IS development projects.

2. MATERIALS AND METHODS

Finland is a densely forested country, with 57% of the total land area of 304 thousand km^2, made up of forest stands[2]. A forest stand is a basic unit of forest management and as such has a sufficiently uniform composition

[1] https://www.metsakeskus.fi/en

[2] Statistics are based on Lähde databases and compiled by the author in June 2025.

of soil and structure. A stand is also bound within land parcels. In Lähde there are 12.6 million stands; thus the average size of a stand is 1.37 ha but it may vary widely: there are stands in the database that are over 10 km^2. The other core feature in Lähde is a habitat. A habitat is also a delineation of forest but unlike stands habitats may overlap. The Centre has defined eight habitat classes but for the purpose of forestry practices three categories are important. The most stringent category, *prohibiting,* forbids all forestry operations, while *limiting* and *affecting* allow. The boundaries of habitats in the prohibiting category dictate the boundaries of stands. At the time of writing there are 428 thousand habitats in Lähde covering the area of 4.6 thousand km^2.

A key requirement for Lähde was to maintain an up-to-date land property data and keep the forest stand and habitat boundaries delimited into parcels. A land property consists of one or more parcels, which are polygon geographic features. There are close to one million parcels with forest stands in Finland. Parcels and their borders are managed by the National Land Survey of Finland and on the average the borders of approximately 200 forested parcels change every day. The change may be a very small one due to a small correction to a location of a boundary marker or a larger one due to splitting of a parcel.

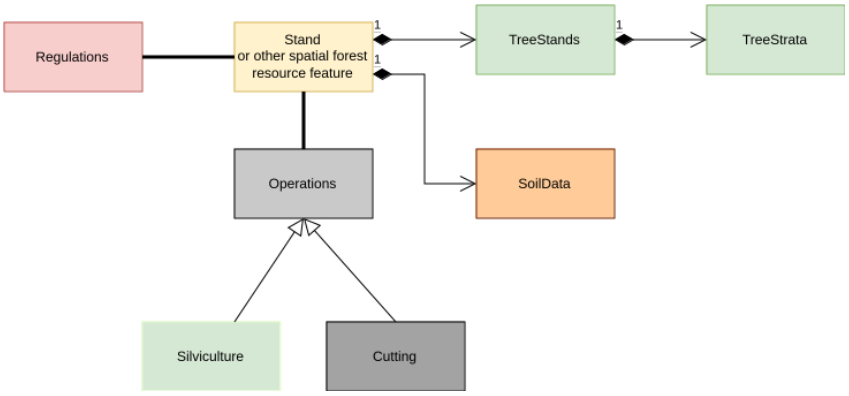

Figure 2: Data model of a forest resource feature.

Figure 2. depicts the internal structure of a forest stand. It is based on the forest data XML standard used in Finland (Sitowise, 2025). The structure is shared, with some omissions, also by other spatial features in Lähde, such as grid cells, tree crown based cells, micro stands, habitats, notifications on forest use, and conserved features. The core component has the geometry and in the case of habitats, the class denoting its legal status and relevance for forestry practices on the underlying forest stands.

Regulations are features that overlap forest stands and thus imply restrictions on forestry practices. They are either other forest resource features (habitats or conserved features) or national parks, protected areas, or other – there are over 30 different types of spatial features having a legal status in Finnish legislation and restricting forestry in addition to habitats, whose legal status is specifically on the Finnish Forest Act.

Operations are either proposed, planned, or completed. Operation proposals are made by forest use advisers (field proposal) or by the external forest simulator used by Lähde (Rasinmäki *et al*, 2009) (simulator proposal). Forest owners are required by the Forest Act to notify the Centre about planned and completed operations. The notifications eventually enter into Lähde in various forms, the most accurate is a cutting geometry from a harvester.

A tree stand describes the state of the forest at a point in time, which can be the time of inventory, current (nowcasted from inventory), or future projected (current year + 10 years in Lähde). At the time of this writing the projected state is being extended to include the followed method of silviculture. Tree stand data may originate from remote sensing, forest owner, or other authority. Tree strata describe the various layers (stories) of trees in the delineation: the species, the role of the layer, amount and quality of timber, carbon stock, and so on.

Soil data describes the quality and class of the site in terms of forest growth. The master information regarding soil data is in the grid cells (the cells form a regular 16 m x 16 m grid over Finland), which may be updated from field surveys of forest stands.

Information about forestry operations arrives at Forest Centre as documents about cuttings and silviculture. These documents usually contain spatial data, for example cutting data may have actual harvester location data. The amount of data varies by the season and can be 10–1,500 per day for cuttings, and 100–5,000 forest use notifications per day. The spatial and other data may be of somewhat varying quality and when processed into stands require business logic defined by experts. Changes on new regulation features are on the order of zero to 50 per day.

The bulk of changes to the data in Lähde comes from new forest inventories. Up to 20 forest inventories are carried out yearly on rectangular areas of 2 to 5 thousand km^2. The data is prepared by contractors and include coverages of 16 m x 16 m regular grid cells, cellular units delineated by tree crowns (size is ~250 m^2), micro stands (size is ~2500 m^2), and delineations of forest/non-forest. Grid cells and other units contain tree stand structure data; grid cells and micro stands contain also soil data. The data is imported into Lähde and used to create new stand network or update existing stands. Grid and tree crown unit data are by far the largest datasets in Lähde taking both the approximate size of 6–7 terabytes of disk space in relational databases.

The core requirements of Forest Centre for Lähde in the call for tenders were the following (Forest Centre, 2020):

- separate data models for habitats and stands
- automatic processes for importing day-to-day, field, and inventory data
- automatic updating of boundaries and structures of stands and habitats due to changes in parcels, prohibiting regulations, and harvester cutting geometries
- user interface for browsing and manipulating stands and habitats.

Field data originates from hand-held devices with offline software for browsing and manipulating stands and habitats.

The main user interface is desktop software that connects to the main system via Internet. The daily use of the main user interface varies between 40–100 edits per day.

The outline of the structure of the forest in stands and habitats (and to some extent also grid cells, tree crown units, and micro stands) is three tree stands (inventory, nowcasted, projected next year and ten years into future) with one or more strata (dominant, undergrowth, retention, etc). The nowcasted and projected tree stands are computed using a stand-level simulation model (Rasinmäki et al 2009, Salminen et al 2005) which is implemented and offered by an external provider as a web service, which the Lähde uses. The following year projected tree stand is computed beforehand to facilitate fast updating of current tree stand in August each year.

The winning offer selected AWS as the platform for the development. Development on AWS typically proceeds by writing the needed infrastructure (databases, machines to run the code, IS components like lambdas, networks, and file storage, etc) as code (IaC), which is executed in an integration pipeline to create and modify the components in the cloud. The main computing components that are used in Lähde are lambdas, containers, and step functions. Only one virtual server (AWS EC2) is used; it is for downloading the land parcel network. AWS step functions is a workflow service that executes actions (calls lambdas, sends events, and performs simple logical functions) based on a JSON data structure that enters and exits the steps the developer has defined. AWS offers a large selection of tools and components of which mainly the above mentioned computing components were used together with databases (PostgreSQL relational database and DynamoDB NoSQL database), file storage (AWS S3), web services (AWS API Gateway, CloudFront), event delivery (AWS EventBridge), and some other tools (AWS Parameter Store, CloudWatch, etc).

The programming languages that were used included TypeScript (IaC and some lambdas), Java (some lambdas), and Python (majority of lambdas and code run in containers and in EC2). Due to the importance of spatial data, libraries such as GDAL, GEOS, and PostGIS, and Python libraries such as Shapely and Fiona were used in most of the code.

A requirement set by the Forest Centre was that the development should be carried out using a modified scaled agile framework[3] as a collaboration between the Centre and the developers. The methodology implied a three-level structure for the work.

1. Epics, which were defined by the Centre for itself, defined the main goals.
2. Features, which were also defined by the Centre, but were discussed with the developers, represented major tasks determined to be needed to achieve the epics.
3. Stories, defined in collaboration with the Centre and the developers, which represented requirements deduced from the features, and which could be implemented within a few days of work.

Stories were the main focus of the work in the development. In the first phase they were identified and described based on the features, their work load was estimated and thus ready-for-work stories were collected into a backlog. In quarterly planning (two days of meetings) the coming quarter was considered: how much

[3] SAFe(tm) by Scaled Agile, Inc

development resources the team has, and what features can be taken under work counting the amount of work their stories need. The stories were then worked on in two-week cycles, which began with reviewing the work done in the preceding cycle, and planning for the upcoming two weeks.

The hierarchy in the development work was very low once the features were defined, prepared, and selected for work. The day-to-day work was carried out by developers and client representatives in collaboration. The Scrum Manager was responsible for facilitating the overall flow of development; and some developers had more responsibility on the architectural design of the system. Most of the developers were initially not very familiar with AWS. Client representatives had various types of responsibilities according to the agile framework or acted as domain experts. At certain stages of the development there were extra meetings needed to solve some pressing problems. Such problems arose from needs to define better the data processing requirements, or the lack of knowledge on how to implement/handle certain things on AWS. People from outside the development team were invited into these meetings.

The development resource was approximately 11 full-time developers from inception until first production release. From the Centre there were 4– 6 people mostly involved in the development as product owners, testers, and experts. After the system went into production, the development resource was much less: 2 full-time developers and 2–3 people working on the Centre side.

3. RESULTS

3.1. System architecture

The development began August 2020 and the first production version was launched on 13 January 2023. Support for inventory data import and processing was added to the production version in August 2023, after which the initial product was considered fully delivered. After August 2023 support for importing field data and importing new types of notifications have been added.

The architectural design of Lähde (Figure 3) is a modular monolith. The modules are conceptually a set of services with a common theme. The theme may be a feature class, functionality, or workflow management. Feature class modules own a database within the relational database management system (RDBMS) cluster. Services are web services or services offered via the event bus. Currently Lähde has 24 modules.

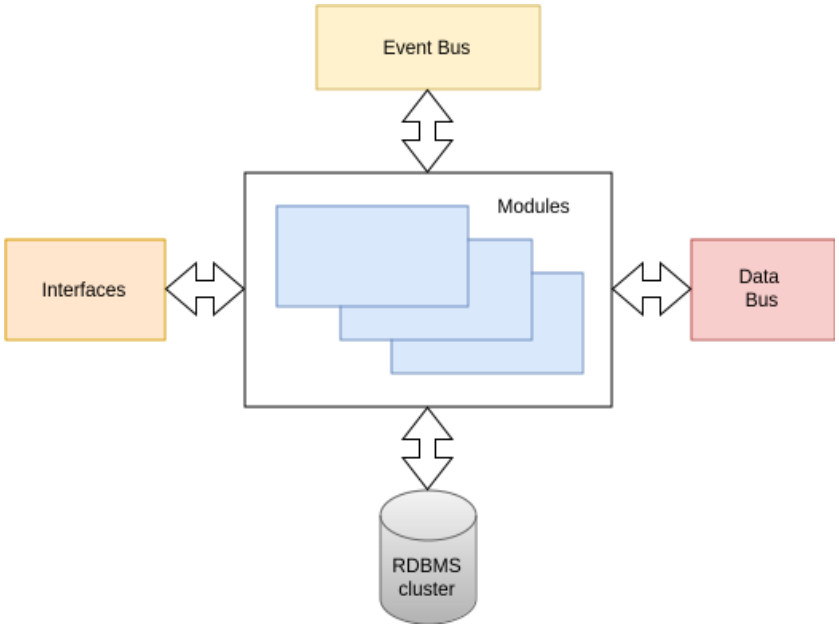

Figure 2: The architecture of Lähde

3.2. Workflows and processes

Workflows import new data and information about forest operations in Lähde. They are initiated from the user interface, by arrival of new data either to the interface or to the data storage for incoming datasets, or as they are scheduled. The processes power the workflows and the services power the processes. During a process execution the modules communicate with each other via the event and data buses.

At any given time there are multiple processes running in Lähde. A process that handles for example stands typically contains a step in which they are loaded from the database into the databus, a step in which some data, for example tree stands, of those stands is modified, and a step in which the stand database is updated with the new data. To prevent processes overwriting each other's changes, the processes are orchestrated based on the parcels they operate on. This is achieved in the process orchestration module by running processes as a

subprocess of a specific orchestration process, which contains a step to determine the parcels the process will work on, and logic to make processes wait for parcel locks to be removed.

Furthermore, the process orchestration contains a mechanism for holding processes to start if there are already so many processes running that the system resources are at risk of becoming overloaded.

In Lähde the process orchestration module runs the processes with Step Functions, Lambdas, and an AWS DynamoDB table (Figure 4). The 4 lambdas are for handling incoming and outgoing events, for maintaining the process document, and for creating a web service for the processes. In the most complex situation, importing new inventory data, the workflow consists of expert users importing various datasets and running various tools separately to achieve a desired outcome. However, even a single, fully automated workflow the process consists of multiple step functions. A basic premise in Lähde is that a process should never fail due to a step function failing due to a service failing or taking too long. That is ensured by wrapping each service request in a process with a timeout check and logic which converts a failure response to a request into storing the error message into the process document and graceful ending of the process.

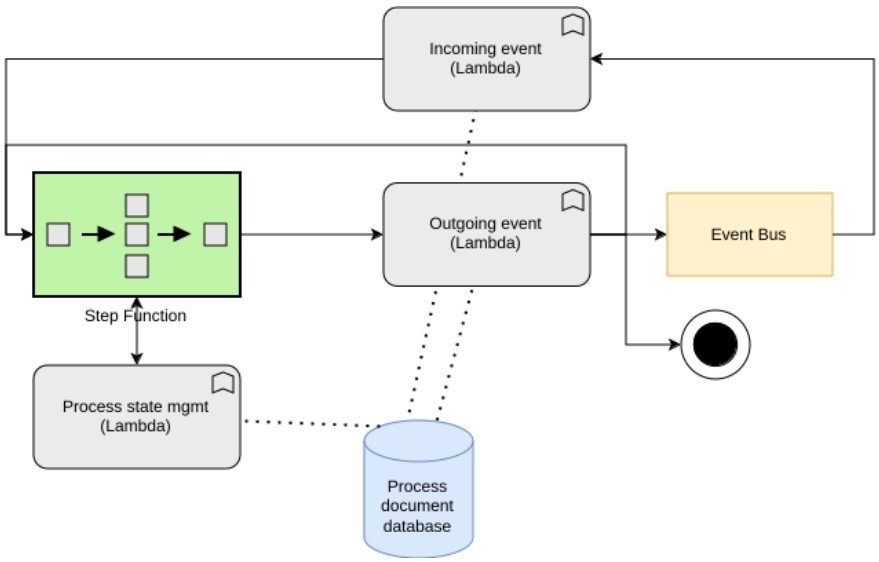

Figure 3: Executing a process in Lähde.

A process is initiated by sending an event to the event bus. The lambda for incoming events receives the event, creates a document for the process, and either initiates the process by starting an appropriate step function or by putting the process on hold to prevent congestion. The steps of the step function are executed according to its logic leading to process state changes and requests to the outgoing lambda. Outgoing requests are either service requests, request to execute another step function as a subroutine, or to end the process. Typically the processes are orchestrated by the parcels, whose stands they modify. The orchestration happens by using a service request to compute the affected parcels, and based on that either to allow the process either to start or sets it to wait until the parcels in question are not anymore reserved for any other process.

In the case of parcels changing the parcel-locking and orchestration becomes interesting. First of all, the task of determining the parcels on which the parcels-changing process will work on, becomes, within the module that maintains the parcel network, a task of downloading a new parcel network, computing the changes, and determining what are the old parcels that need to be locked. Then the parcels-changing process itself needs to compute the new boundaries of the stands, and their new tree stands based on those, before it can finally switch to the new parcel network.

When boundaries of parcels or prohibiting habitats change or stand borders change due to some reason, the stands and stand network need to be adjusted. The adjusting is done in a processing pipeline, which comprises collecting the clips from which to build the new network, splitting the clips further to ensure stands have “good” shape, computing the forest and soil parameters of the clips, running an algorithm to combine the clips into stands, and, finally, to assess the results.

Several methods for handling the variability in the scale of data processing were devised during the development. The two main methods were 1) running processes, steps of a process, or computing within modules in parallel; and 2) using containers instead of lambdas, or bootstrapping computing into a container from a lambda. The last method, container bootstrapping is a method where lambda first examines the task it has got, and if it seems large, starts a container for the task and handles the responsibility to respond to the request to the container.

4. DISCUSSION AND CONCLUSION

The Lähde system is a central component in the information processing at the Forest Centre. It is the source of current (nowcasted) and ten years into future projected forest resource data and it holds up-to-date environmental information that is relevant to forestry. The data changes continuously due to (i) inputs from forest advisors, experts, and field workers; (ii) external information from forest owners, authorities, and contractors; (iii) changes in land parcels and environmental protection; and (iv) new forest inventory data.

The Lähde system was successfully delivered and put into production after four issues (i) the data processing system was developed, (ii) the workflows and data management were defined in sufficient detail in customer—developer collaboration, (iii) the problem of spatial processing of stands and stand network was solved, and (iv) the problem of processing large and large sets of features was solved. The key idea in the first issue was to use land parcel based locking of processes, which meant using an orchestration process (step function) as a wrapper for the actual workflow process. Also the idea of implementing the processes with middleware (step functions and the process orchestration module) using the services from the modules had to be developed. The second issue required that the customer was more specific about the requirements and algorithms, after which the developers had to translate those into process middleware and services. This remains an issue, it is surprisingly difficult to divide the processing on the one hand to the middleware and on the other hand to the services. The third issue is partly about splitting and constructing polygons (stands) so that the result is "natural", and partly the problem of maintaining stands as irregular tessellations of forested area of land parcels. For the fourth issue several solutions were devised. The service in the module can in some cases split the task into a set of smaller tasks, suitable for a single lambda, and employ a step function to execute those in parallel. In some cases the service receives the task in lambda but determines that it is too large for it and bootstraps itself into a container run, which then executes the task,

The architectural design clearly separates the physical data model and the conceptual data model. The physical data model is how the module internally stores its features into the database. The conceptual data model is how other modules see and possibly modify the feature. In addition to these two models, the interface may present the features using one or more specific models. In practice separating these three or more models turned out to be very difficult. Reasons include the way work was organized and lack of communication. Data modeling typically requires long-term commitment, which was difficult to achieve in a fast moving, agile project.

Development is continuous and cyclic execution of planning, designing, developing, reviewing, reflecting, and adjusting. The trick is to advance at each cycle, but even understanding what 'advance' means may differ between the client and the developer. For a client it usually means new features but for a developer advancement may mean better understanding of what the client wants, the technology that is used, and how some things actually should (have) be(en) done.

ACKNOWLEDGMENTS

The author wants to thank the Finnish Forest Centre and its employees who have worked in the Lähde project for the collaboration and the permission to use the development as an object of a study. The author wants also to thank the colleagues, present and past, in AFRY and Sitowise for collaboration in the development.

REFERENCES

Ahokas, K. 2024. Over 20 million euro IT project renewal was successful with the help of an agile method – "hardly any developer was with it familiar with it". News article in tivi.fi dated 22.5.2024. https://www.-tivi.fi/uutiset/a/207fe9ea-ab29-4c42-8f40-6fe7a00d11b8 (In Finnish, the full article is behind a paywall)

Pynnönen, S., Haltia, E., and Hujala, T. 2021. Digital forest information platform as service innovation: Finnish Metsaan.fi service use, users and utilisation. Forest Policy and Economics 125. https://doi.org/10.1016/j.forpol.2021.102404

Rasinmäki, J., Mäkinen, A., and Kalliovirta, J. 2009. SIMO: An adaptable simulation framework for multi-scale forest resource data. 66(1) 76-84. https://doi.org/10.1016/j.compag.2008.12.007

Salminen, H., Lehtonen, M., and Hynynen, J. 2005. Reusing legacy FORTRAN in the MOTTI growth and yield simulator. Computers and Electronics in Agriculture. 49(1) 103-113. https://doi.org/10.1016/j.com-pag.2005.02.005

Sitowise, 2025 https://metsatietostandardit.sitowise.com/ accessed 1.7.2025

