ENVISAGE

ENhance **VI**rtual learning **S**paces using **A**pplied **G**aming in **E**ducation

H2020-ICT-24-2016

D2.1

Analytics infrastructure installation and data aggregation

Dissemination level:	Confidential (CO)
Contractual date of delivery:	Month 04, January 31th, 2017
Actual date of delivery:	Month 06, March 31th, 2017
Workpackage:	WP2 - User data aggregation, shallow game analytics, and visualizing learning
Task:	T2.1 - Tracking infrastructure for user data collection and aggregation
Type:	Other
Approval Status:	final
Version:	4
Number of pages:	42
Filename:	D2.1_GIO_v4.docx

Abstract

The ENVISAGE project aims at supporting and improving virtual labs through a structured and data-driven process. The document at hand gives a detailed description of the tracking infrastructure built for and used in the ENVISAGE project to collect the necessary data. Based on the pedagogical requirements that were defined in one of the previous deliverables, i.e., D1.1, the tracking infrastructure will gather and pre-process all necessary data. This data is then used to visualize user behavior and also serves as input for deep analytics. Such an infrastructure can be at least divided into four different layers. The first layer tracks raw user events being sent from a virtual lab to GIO servers. Typically, a virtual lab is a website or mobile app, and GIO provides the necessary piece of

software to be integrated into the virtual lab for tracking. The format of the raw data is typically neither sufficient to allow direct visualization nor is it proper input for more sophisticated algorithms. Therefore, there exists a second layer that transforms and aggregates the raw data into a more manageable format. During this process the data is also enriched with additional (meta)data, such as geo-information, and raw events are grouped into user sessions. In an advanced setting, we have a third layer where user data can also be augmented with predictions about future users' behavior or automatically inferred traits. After this step, the data needs to be made accessible again. To accomplish this in a flexible way, the fourth layer consists of an API that makes all data available in different forms and formats. While raw data access is allowed, the data can also be accessed in an aggregated form or on a user level with all its metadata.

co-funded by the European Union

Copyright

History

Version	Date	Reason	Revised by
1 (alpha)	8.2.2017	Initial version of the Table of Contents	Fabian Hadiji, Marc Müller, Spiros Nikolopoulos
2 (beta)	17.3.2017	Beta version incorporating most of the envisioned content	Giannis Chantas, Fabian Hadiji, Christoffer Holmgård, Marc Müller
3 (final)	27.3.2017	Final version with content fixed and last request for last comments	Giannis Chantas , Fabian Hadiji, Marc Müller
4 (final)	31.3.2017	Fixed typos and improved language. Extended list of abbreviations. Minor edits.	Fabian Hadiji, Marc Müller

Author list

Organization	Name	Contact Information
GIO	Fabian Hadiji	fabian@goedle.io
GIO	Marc Müller	marc@goedle.io

Executive Summary

The ENVISAGE project aims at supporting and improving virtual labs through a structured and data-driven process. The document at hand gives a detailed description of the tracking infrastructure build for and used in the ENVISAGE project to collect all necessary data. We begin by reviewing the pedagogical requirements that were already defined in the previous deliverable D1.1. We then discuss how these requirements can be satisfied from a technical perspective. We then describe how the tracking infrastructure gathers and preprocesses the necessary data so that user behavior can be visualized. At the same time, the data also needs to be converted so that it can be used for deep analytics. Such an infrastructure can be divided at least into four different layers and we describe each layer in detail.

The first layer persistently tracks raw user events which are being sent from a virtual lab to GIO servers. For this purpose, multiple servers run in parallel to ensure availability and zero downtime deployments. Typically, a virtual lab is a website or mobile app, and GIO provides the necessary software to be integrated into such a virtual lab for tracking. We will exemplify this process based on the "Wind Energy Lab" which is currently in use for the first ENVISAGE feasibility case study.

The format of the raw data is typically neither sufficient to allow direct visualization, nor is it proper input for more sophisticated algorithms. Therefore, there exists a second layer that transforms and aggregates the raw data into a more manageable format. As the current infrastructure does not support real-time tracking, all events from the different tracking instances need to be merged regularly. Secondly, the aggregated data is imported into a database. During this process the data is also enriched with additional (meta)data, such as geo-information, and raw events are grouped into user sessions.

In an advanced setting, we have a third layer where user data is augmented with predictions about future users' behavior. In such settings, the imported data is used to construct feature vectors that serve as input to (machine learning) algorithms. In supervised settings, historic feature vectors are used to train a model. Such models are then used to make predictions for the current users. All output from the algorithms and their predictions are stored in the database as well, to ensure fast and easy access. In unsupervised settings, patterns in the data are discovered and the data is enriched with indicators of these patterns, e.g., individuals' memberships in various groups.

In a last step, the data needs to be made accessible again. To accomplish this in a flexible way, the fourth layer consists of an Application Programming Interface (API) that makes all data available in different formats. Besides allowing raw data access, the data can also be accessed on a user level with all its corresponding metadata. To close the entire loop including all components of the ENVISAGE project, the data needs to be made available to the virtual labs and the authoring tool (details on the virtual labs were given in D1.1 and details on the authoring tool will be given in deliverable D4.1). We will also describe in detail how the existing infrastructure has been extended to the needs of the ENVISAGE project and lastly, we give some initial statistics on already tracked user data. These statistics are also based on the "Wind Energy Lab".

Abbreviations and Acronyms

API	Application Programming Interface	JSON	JavaScript Object Notation
AWS	Amazon Web Service	KMS	Key Management System
CDN	Content Delivery Network	NoSQL	Not only SQL
CPU	Central Processing Unit	RAM	Random Access Memory
EBS	Elastic Block Store	RDS	Relational Database Service
EC2	Elastic Compute Cloud	S3	Simple Storage Service
ELB	Elastic Load Balancer	SHA-1	Secure Hash Algorithm 1
FTP	File Transfer Protocol	SLA	Service Level Agreements
GA	Google Analytics	SQL	Structured Query Language
GTM	Google Tag Manager	SSL	Secure Sockets Layer
HTTP	Hypertext Transfer Protocol	TLS	Transport Layer Security
I/O	Input/Output		

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 The ENVISAGE Goals

The overall goal of the ENVISAGE project is to improve virtual labs through a structured and data-driven process. This requires data from the users, i.e., the students, of different virtual labs, and an authoring tool that is capable of adapting existing virtual labs based on the data analysis. Initially, existing labs are extended in such a way that they integrate tracking software to allow gathering of behavioral user data. The software to extend a lab is typically provided by third-party services and in the case of the ENVISAGE project, GIO develops and provides this software. This software needs to be integrated into the source code of the virtual lab by its developer or maintainer. The collected data is typically in the form of events with additional metadata attached to it (see Section 4.1 for more details). Tracking these events on a very fine level does not only allow shallow analytics but also provides the foundation for a more advanced analysis using machine learning, i.e., deep analytics. Using this data, one tries to understand the student's behavior and to adapt the structure or content of the virtual lab in order to support the learning process.

Shallow analytics provides the first insights into the behavior of students solving different tasks. However, as the name indicates, it mainly monitors their activity and this is often done on an aggregated level without looking at individual users or particular segments of students. Examples of such shallow analytics are the number of active users on a given day or the average duration of a session. Here, the focus is to provide information on the lab level and to aggregate information across all users. However, in order to gain actionable insights or to even adapt the content dynamically, more sophisticated approaches are necessary. Additionally, the focus should shift to the individual user or prominent groups of users to provide a more personalized experience. Such personalized education has been identified by different stakeholders as an area that can revolutionize the state of education.

The ENVISAGE project aims at utilizing machine learning technologies to describe decision making, to segment users automatically, or to predict their future behavior or performance. Often, the algorithms in use cannot directly work on the events but instead need a numerical or categorical representation of the information. An event as such is just a message or dictionary of semi-structured information as described below. Machine learning algorithms, however, tend to work on data structures such as vectors for one user or matrices for several users. Therefore, all events of a single user must be transformed in some kind of vector representation. When setting up the tracking, one should make sure that the tracking within the virtual lab collects all necessary data from the students and their actions. This data is then transformed and features are constructed from the raw event stream.

1.2 The ENVISAGE Requirements

We can differentiate the requirements of the various parts of the project in different groups. On the one hand, there are general requirements that need to be satisfied to solve the problem on a high level and to support the greater vision. Then we have specific requirements that result from particular problem definitions or metrics that place certain constraints on the tracked events in order to solve subproblems.

General requirements neither take technical restrictions into account, nor discuss in detail the pedagogical point of view. Instead they summarize on a high level how an infrastructure must look like, in order to deliver value to the project. On the other hand, we have developed during the course of the first months a set of requirements that stem from teachers and people involved in learning analytics. These requirements specify certain information in detail and describe different scenarios with schools, teachers, and students involved. For example, we need to understand the sequence of steps a student is taking to understand their behavior and this information needs to be grouped based on teachers, schools, or regions. Let us now start with the general requirements first.

General requirements

The general requirements describe on a high level the different needs so that the entire vision of the project can be realized:

- **Provide data for shallow analytics**: Often simple counts provide a first insight into the data. While some of this information can be accessed via existing analytics tools (see Section 2), learning analytics also imposes specific requirements that are typically not satisfied by existing tools. Here, a common example is the time to solve a (sub)task and we touch upon other examples further below (see Section 8 as well).
- **Aggregate, enrich, and augment data**: As described in the next bullet, deep analytics and machine learning require data in a flexible form. All kinds of features are generated based on the raw events and this processing requires a lot of creative thinking. To support this process, the raw data is aggregated and grouped as necessary. Additionally, the data can be enriched with third-party information. For example, geo-information, weather information, or other demographic information that is available while respecting privacy restrictions. When this data is used in algorithms to predict future user behavior, the results can be used to further augment the existing data.
- **Provide data for deep analytics**: Once we move beyond shallow analytics, the setting advances and gets more challenging. Here, the existing analytics platforms typically do not satisfy the requirements at all. Either, the data is aggregated too much or data is only provided in its raw form. Here, the ENVISAGE project is providing data to the deep analytics interface in such a form that it can easily be used in advanced algorithms. The input to deep analytics is manifold. It does not only take data into account that is generated akin to the shallow analytics but also requires entirely new input. For example, to model a student's decision making, a state model of the student should be captured continuously throughout the entire usage of the virtual lab. This often tracks various boolean features which indicate behavior or traits on a very fine level. E.g., has clicked button of type A or has changed the setting of parameter B. Now, depending on the algorithm, various interactions between such models will go into the machine learning model as well. If we stay in the boolean

case, we could add pairwise interactions such as XOR[1] operations between two simple features. This creates additional, more complex, features that the algorithm is not able to construct itself. On top of that, interaction between different, non-boolean, datapoints can be added as well. For example, the sum or the quotient of two simple numerical features. In summary, deep analytics is typically user-centric and very detailed, while shallow analytics is lab-centric and makes use of higher level statistics.

- **Making data available again**: Once all data has been collected, transformed, enriched, and augmented, the data needs to be made available again to the virtual lab or its authoring tool. Otherwise, the data is locked inside a data warehouse and the project faces the same limitations as if a standard analytics tool was used.

Specific requirements

We will now try to make these general requirements more tangible and give a few examples what data should look like for specific learning analytics use cases. The following metrics were originally introduced in deliverable D1.2. While D1.2 identified the requirements from the perspective of a teacher, we are now going to describe the technical requirements that result from these metrics. In the deliverable at hand, we will also sketch how we will solve and implement these metrics. This includes a discussion of technical requirements that must be put in place. The metrics presented in D1.2 give rise to a more concrete set of requirements. We will here review those metrics again and explain how they can be implemented based on GIO’s tracking infrastructure.

- **Time-on-task**: This measures the time spent to solve a (sub)task. Looking at our setting and the event-based tracking, this is measured as the time between two specific events. This requires that the beginning and the end of a task is clearly identified by a named event.
- **Time-to-completion**: Similar to the previous metric, we need to measure a “completion” event. This event should clearly mark if the goal of an activity has been reached. In contrast to the previous metric, the overall time spent from start to finish of the entire activity is measured.
- **Class categorization profile**: A group of students, e.g., a class, can be classified into different categories. In order to perform such a categorization, the performance of each student must be determined. Here, we can either use a proxy, e.g., time-to-completion, or an alternative score needs to be tracked additionally. This grouping does not necessarily have to be on the class level but can also be a more flexible grouping. Therefore, the tracking also needs to support a general grouping or clustering of students based on the tracked data.
- **Perceived, expected and actual general class or group profiles**: This metric requires more than the standard tracking of events. In particular, it requires specific input from the users of the lab, e.g., input from students asking about their own

[1] exclusive or

perception. This can be extended by also obtaining corresponding data from teachers and classmates.

- **Levels of proficiency**: Here, it is envisioned that teachers can compare the performance of their own classes with other classes and country averages. Additionally, teachers should be given the option to add a level to each task. As the tracking is user focused and does not model the learning tasks themselves, the analytics dashboard should query a teacher for the input of task levels. As mentioned above, a comparison on a class level or similar grouping requires the virtual lab to transmit the group information.
- **Mastery index**: This metric compares a particular student's performance with the average, minimum, or maximum of a class or different group of students. I.e., it essentially grades a student based on a specific normalization. When the tracking of groups of students is in place, this is more an analytical challenge in the dashboard than a tracking issue, and it needs to respect the student's privacy.
- **Travel-path related metrics**: This requires a precise sequence of events which have been taken by the students. Fortunately, this is naturally supported by an event-based tracking. Nevertheless, generating value from this data is still a major challenge and does not only require capable algorithms, but also a proper visualization.

Although most of the metrics above can be realized, there are some restrictions and limitations that should be pointed out at this point. It should be kept in mind that the designed system should be generally applicable beyond specific virtual labs or schools. Otherwise, it is difficult to build a product as a (self-)service and instead each new lab would require a large amount of customization and the platform develops more towards a consulting service or agency business. Clearly, some of the requirements above will always require input from teachers or virtual lab designers, and this is generally not avoidable. For example, the concepts like *proficiency* and *mastery index* need a custom definition for each lab. Unfortunately, it is not straightforward to envision a product or system that models such a scoring automatically for every task and setting. However, with our current approach we are aiming at a solution that requests the definitions from a teacher and we aim at providing a design and interface so that this can be done in a highly automated fashion. For example, we can think of a simple interface that asks for specific occurrences of events and some parameter values. These can then be used to implement a scoring function for a virtual lab. Besides this issue, here a some other limitations that need to be kept in mind:

- The metrics above induce a certain structure of the virtual lab users. For example, each class is composed of students or a group of students. Additionally, there exist teachers who are not lab users but may also provide additional data. This structure requires a group-call functionality, so that users can be grouped into these different (sub-)groups. As opposed to an event which typically tracks an action, a group-call adds metadata to a user in form of a cluster affiliation. However, it is not possible to build a model for each school in every city and country with all of its specifics.

Therefore, we will potentially have to find some workarounds for school specific issues, in order to guarantee a general system that is applicable on a large scale.

- As mentioned in the previous bullet, a teacher may add information on students. This can be, for example, a record of progress for each student. This is generally not a problem, however, the information will have to be transmitted in a general form. It will not be possible to build a database system for each school for their own record keeping. Additionally, the virtual labs will have to be implemented in such a form that they request a user authentication. Otherwise, records cannot be aligned with the tracked data during the usage of the lab. This is particularly challenging in school settings where computer pools are used, i.e., many students use the same machine.
- Although many interesting scenarios could make use of a real-time analytics system, e.g., teacher observing the student's progress during a class, this will most likely not be manageable in the timeframe of the project with the limited resources. A real-time system is more complex in various aspects and its maintenance requires many resources. Instead, the tracking will be in real-time but its aggregation and augmentation will be done in a batchwise fashion. This can result in lags of a few hours when data is accessed.

1.3 Scope of the Document

We start by summarizing the state of the art on analytics, including learning analytics. We then describe the entire GIO infrastructure in detail. This includes the different layers for data tracking, data aggregation, data augmentation, and data access. Before giving concrete examples of already tracked data, we also describe how we extended the existing GIO infrastructure to meet the requirements of the ENVISAGE project.

2 Analytics State of the Art

The current analytics landscape can initially be quite confusing. Typically, every platform puts barriers in place when it comes to accessing the raw data in order to maintain a competitive advantage. Nowadays, it is expected that the data, which is generated by the usage of website or apps, has a certain value if presented in a meaningful way. To monetize the data, it is necessary to have information which nobody else owns. Such a competitive advantage can be reached through data presentation, aggregation or evaluation. This applies in particular for the processing of raw data and then enriching it with other data sources or computations. Analytics provides a service to analysts, product owners and marketing teams at the same time. Every individual niche has a different focus and analytics products have different target groups as well. Monetization is possible if a certain niche can be satisfied with the provided results. On the other side, monetization is also possible by presenting meaningful insights. Meaningful insights can be, for example, behavioral patterns that result out of an algorithm which potentially works on multiple data sources at the same time. This may also span across data of different customers. As an example, when two e-commerce websites are tracked, the analytics provider gets data from both websites. At the same time, the analysts from each website only see their own website and only have access to their own data. However, the analytics provider is able to combine the learnings from both websites and by doing so, it can provide more value to both customers. Every analytics provider wants to have such kind of proprietary data. On the one hand, this can be data which is gained through an exclusive tracking, e.g., very detailed user behavior in certain verticals or relationships in (social) networks. On the other hand, this can be achieved via access to additional data sources and aligning this input with the original data, e.g., geo location and demographic information for all tracked users.

As in the example above, an analytics provider is not only able to see users of one website or app. Every analytics provider has numerous customers, and therefore, multifaceted information about different users per website or app, and often across different customers. Once an analytics provider has this information, it tries to secure and protect it as much as possible. This often results in limited access to the tracked raw data. Each company is protecting their own position and their competitive advantage as much as they can.

Another reason is the preprocessing of the data. Many dashboard driven analytics platforms are using aggregated data to simplify and enable fast and easy access to information. One prominent example is Google Analytics (GA). The geo distribution of all users can be accessed with only one click. Often, GA’s customers do not need information on a single user level but aggregated data is sufficient. The objective is mainly to show cohorts of users. This also has simple technical reasons: saving bandwidth and building human readable visualizations.

Depending on the complexity of an app or website, there is a vast number of event combinations a user can do. For example, the free language-learning platform duolingo[2], that includes a language-learning website and a corresponding app, needs a sophisticated tracking strategy. Assuming every hit of the keyboard buttons during a translation lesson is

[2] https://www.duolingo.com/

tracked, a translation of “Yes, please.” to the German “Ja, bitte.” needs ten events. This vast amount of events would not improve any analytics but instead present an enormous overload of information and introduce a lot noise. Instead, it is more important to know, if the task was solved correctly, and how long it took to solve it. This would lead to two events only, first starting the task and then ending the task. The latter event would contain an additional value of “true” or “false” to provide information about the user's performance. Nevertheless, this still creates a flood of events for millions of users. Therefore, high bandwidth is needed and a large data storage. Of course, simply the amount of people who are using duolingo (they had 110 million[3] users worldwide in 2015) presents challenges for any kind of infrastructure. Especially, for analytics provider which receive all the tracking data. duolingo is known to use Mixpanel[4] among other services. Mixpanel calls itself an advanced analytics provider. While using Mixpanel, one will recognize certain constraints when it comes to accessing data and receiving data. This is because any analytics provider has to pay to store and to process tracking events within its infrastructure. Costs are scaling with the number of users. Therefore, downloading data, querying databases, or enriching data will be charged by the service. Or it is only available in higher priced tiers. Mixpanel as an example, limits its access to tracked raw data and also to the amount of data that can be tracked. This can be seen in Mixpanels pricing[5], the number of custom events which can be tracked is limited to 15 custom events. Also, the export of raw data will take up to five days[6]. To build the metrics in D1.2, and to be able to enrich the data as the project envisions, using a provider like Mixpanel is not a feasible approach.

Even if raw data access was granted, and there was no limitation to custom events, an infrastructure is required to handle the tracked data. It is unlikely that the operator of an education learning website or a provider of such an app has the resources, the technical skills and the capacity to handle raw analytics data and to process it.

2.1 Advanced Analytics

There were two major developments in the analytics space in the past years. The first significant change was the advancement from analytics[7] to predictive analytics. Predictive analytics allows making predictions based on past data which. Today, predictive analytics is considered to be more or less a subset of analytics. It mainly supports the decision-making process for an analyst. Many companies, such as Mixpanel, Localytics[8] or Amplitude[9], try to extend their classic analytics service with the ability to make predictions of user behavior. This kind of applied machine learning is limited in accessibility and reconfigurability.

[3] https://techcrunch.com/2015/06/10/duolingo-raises-45-million-series-d-round-led-by-google-ventures-now-valued-at-470m/

[4] http://mixpanel.com/

[5] https://mixpanel.com/pricing/

[6] https://mixpanel.com/help/reference/exporting-raw-data

[7] This contains also the access to shallow analytics

[8] https://www.localytics.com/

[9]https://amplitude.com/

Preconfigured interfaces allow users of these services to make predictions on their tracked data easily. On the one hand, such applications are realized with only a few clicks. But on the other hand, the usage of the output for automating marketing campaigns or just viewing the used features is limited. Additionally, the enhancement or editing of the models, or the learning process is not possible.

After predictive analytics, the term prescriptive analytics[10] has become popular recently. The naming of prescriptive analytics stems from the combination of predictive and descriptive analytics. But it is supposed to mean much more. Prescriptive analytics is the union of all applied scientific disciplines that support decision making, allow a pro-active action and at the end get feedback to improve decisions and re-enforcement actions. It closes the loop between observations, decisions, and actions.

2.2 Learning Analytics

With respect to the required metrics presented in deliverable D1.2, there is no known analytics platform that is able to provide all this information. General analytics providers are often featuring a certain number of metrics[11][12], like the average session time, lifetime value, attribution source, or retention. Some of these metrics like average session time are also interesting in the educational learning context. But metrics such as lifetime value or attribution source do not play an important role in analyzing student's learning behavior. The main problem is that the pedagogical aspects are not taken into account by common analytics platforms. On the one hand, the tools are focused on e-commerce and apps in general. This is because the metrics in the learning analytics space are not as standardized and generally applicable as in other industry sectors. On the other hand, using results gained from analytics in a pedagogical environment also requires a service that can handle and return results that are interpretable by educationists and give theses persons the ability to adjust the educational learning application.

[10] Deep analytics could be placed in the prescriptive analytics space

[11] The 8 Mobile App Metrics That Matter http://info.localytics.com/blog/tracking-lifetime-value-for-apps

[12] How can I use Mixpanel to achieve my business goals? - https://mixpanel.com/help/questions/articles/what-should-i-track

3 Infrastructure

In this section, we begin by explaining how we designed our general infrastructure and how it works technically. In the following sections, we describe different aspects of each layer in greater detail. Looking at the entire infrastructure, we can distinguish at least four different layers.

- **Layer 1**: Data tracking (Section 4)
- **Layer 2**: Data aggregation (Section 5)
- **Layer 3**: Data augmentation (Section 6)
- **Layer 4**: Data access (Section 7)

Figure 3.1 shows how all of those four layers interact among each other, as well as with the virtual labs and the authoring tool. In a nutshell, the pipeline is initiated by a virtual lab sending data to the infrastructure. This data is then stored persistently, enriched with additional information and stored in an aggregated form in a database. Then we have a data augmentation layer that can make use of raw data as well as of preprocessed data to employ deep analytics. The results from this layer are made accessible in same the way as the aggregated data by means of an API. This API can be queried by the other components such as the virtual labs or the authoring tool. However, it also provides raw access to all partners, to develop algorithms and visualizations based on all tracked data.

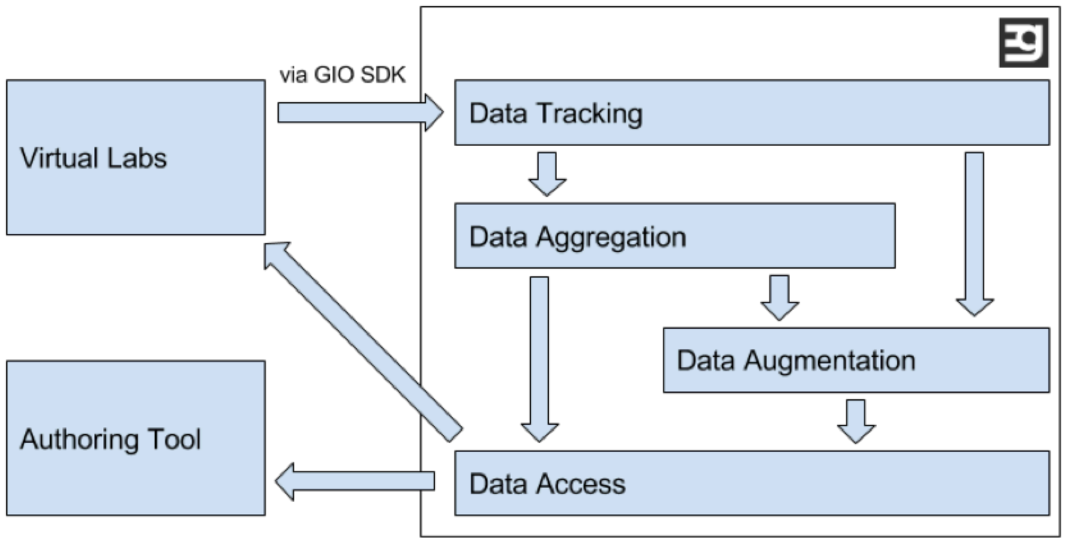


Figure 3.1 *The diagram gives an overview how the GIO infrastructure is built and how it interacts with the other components of the ENVISAGE project. The different parts of the GIO infrastructure such as data tracking, data aggregation, data augmentation, and data access are described in the following sections.*

We now describe each layer in greater detail. While initially Layer 3 is not fundamentally necessary, we still describe it as central part of the infrastructure, as it influences the way deep analytics will be incorporated in the future. As all four layers are run in the cloud based on Amazon Web Service (AWS), we will also give reference to different AWS products that we are making use of to operate the infrastructure. Using a cloud based service has various advantages but it comes naturally with some disadvantages as well. Scaling an infrastructure like the one we envision, reduces the work for various use cases to a few mouse clicks. The reliability is guaranteed by the cloud service based on various Service Level Agreements (SLA). And in the setting of platform-as-a-service, one always benefits from continuous improvements of the products. On the other hand, one quickly starts to become dependent on a single service provider as moving parts of the infrastructure becomes unfeasible without serious efforts. Additionally, many of the benefits come at their costs. Certain matters of expense are often neglected but become apparent in the cloud setting, for example data transfer costs.

4 Data Tracking

To improve educational learning, we have to represent the behavior of students. This behavior is tracked through their usage patterns of educational learning tasks. In a first experiment, we tracked various events which students triggered while solving different tasks in a virtual lab.

We now describe the tracking layer and common definitions are introduced. We differentiate between metadata and telemetric data. Furthermore, we introduce the concept of event-based tracking.

- **Metadata**: There are different types of metadata. At this point, the technical, ecological, environmental, and demographic metadata should be mentioned. Every branch and technology provides different definitions and types of metadata. We are focusing on:
  - Technical metadata: For example, the platform type, operating system or application version. These attributes resolve information about the used device, system, and application.
  - Ecological and environmental metadata: "Who, What, Where, When". For example the user identifier, the geo location or a timestamp. These attributes resolve information about the user.
  - Demographic metadata: Demographic data is user specific data as well. The data holds demographic information about a user. Age, gender, and language are classic examples for demographic data. The only demographic information which is collected at the moment, are the language and country information. These attributes are part of the metadata and part of every event request to the GIO tracking infrastructure.
- **Telemetric data**: Telemetric data is a continuous stream of data. It is used to represent the usage behavior and actions that manipulate the current state of the application. The view of a page could be part of telemetric data, but also the interaction with a control item. In the case of a "view"-event for a webpage, we want to know which page was viewed. In the case of interactions with the control item, we also want to know how the state of the item has changed due to the interaction. Especially for deep analytics, this provides information that can be used to adjust or improve behavior. Telemetric data can be unfolded to simple event streams or to event streams that also have additional context information, like changing a certain state or reaching a certain state.

4.1 Concept of Event-based Tracking

While GIO is receiving a continuous stream of events of the students (which are the main users in educational learning), GIO is able to get information about:

- How are they solving a lab?
- How is the distribution of the events which were triggered?

- How is the behavior during one session?
- How often does a student return?
- How does a student change a certain state?
- When does a student reach a certain state?
- How do different usage sequences look like?

To answer these questions, GIO needs data that is tied to every event. GIO typically uses two main variables to track an event. The first is the action, e.g., “click”, “view”, “start”. These variables are the specific actions which are done by a user. The second variable can specify an object, e.g., “lesson”. These two variables are concatenated and delimited by a ".". This approach can be used to build hierarchies with more than two levels. Furthermore, GIO can make use of an identifier field “event_id”, to identify the event.

As an example, if one wants to track the “start”-event of a lesson, the event would be "start", the specifier would be "lesson" and the event identifier could be "12" if lessons were simply enumerated.

If an event has a certain value, like a duration or a reached level, GIO provides the custom field “event_value”. This allows GIO to track the duration of an event. This would result for the event "finish.lesson", with the event id " 12", to an event value of “65” if the time to complete the lesson would have been 65 seconds.

To handle an event and to track a request, GIO has defined three mandatory fields:

- user_id (string): A unique user identifier which is unique per virtual lab.
- ts (integer): The Unix timestamp when the event was triggered.
- event (string): The name of the tracked event.
- app_key (string): This key is used to identify the application and it is unique for the entire platform.

Furthermore, GIO has the capability to handle additional attributes in form of metadata:

- event_id (string): The event identifier which specifies the event, e.g., a lesson identifier
- event_value (string): The event value, e.g., the duration of a lesson.
- timezone (integer): The UTC timezone offset in milliseconds. For UTC+1, we would receive 3600000
- locale (string): The locale represents the language as ISO-639 and the country as ISO-3166 based on the locale from the device. As an example, for a system and browser with a German locale, we would receive “de_DE”
- app_version (string): The application version which is used. This information can be used to distinguish between different application versions.
- device_type (string): The name of the device type. For now, we hardcoded “desktop“ as device type in the experimental application because the browser does not always allow to retrieve a particular device.
- screen (string): A combination of width and height of the devices screen that is used.
- In the future:

- group_id[13] (string): An unique identifier for a group, class, category, or company. A user can have more than one “group_id”.

The following data object shows a tracking call of an event that GIO’s backend would typically receive. This example is taken from the “Wind Energy Lab”[14]. The corresponding integration is described in Section 4.6 :

```
{
     "user_id": "1523318114.1486661404",
     "remote_addr": "xxx.xxx.xxx.xxx",
     "locale": "de_DE",
     "screen": "1366x768",
     "ts": 1486665339,
     "app_key": "90e167a8ba993ab18f52ec7fdb44fdea",
     "user-agent": "Opera/9.63 (Macintosh; Intel Mac OS X; U;
en) Presto/2.1.1",
     "server_ts": 1486665338,
     "app_version": 1.0,
     "device_type": "desktop",
     "timezone": 3600000,
     "event": "finish.lesson",
     "event_id": " 12".
     "event_value": "65"
}
```

4.2 System Architecture for Tracking

The infrastructure needs to suffice different technical requirements. We describe these requirements in the following and also how we plan to satisfy these requirements:

- **Scalability**: With an increasing number of virtual labs and an increasing number of active students using the labs on a regular basis, the tracking infrastructure needs to scale accordingly.
- **Reliability and Availability**: If the tracking infrastructure is offline, data will be lost irrecoverably. To prevent this from happening, the infrastructure always operates several tracking instances at the same time. At any point in time, there should be sufficiently many machines running that a failure of one machine does not result in any problems. On top of that, data should be stored redundant that a data loss is only possible with low probability.
- **Data Protection**: It is critical that data is protected from unauthorized access. This typically does not only hold for the data stored persistently but also during transfer.

4.2.1 Scalability

There are different aspects of the infrastructure that need to scale based on the number of daily active users and events sent per virtual lab. The infrastructure also needs to be able to

[13] This attribute is necessary to satisfy the metrics raised in D1.2
[14] http://windenergy.ea.gr/

handle peak times, e.g., weekday mornings in Europe. In first place, the tracking layer needs to handle all incoming requests from all virtual labs. This needs to be guaranteed without any downtimes because these would cause unrecoverable loss of raw data. As soon as more virtual labs are tracked, a single machine may not be sufficient anyhow, so that the load should be distributed among several machines in a uniform way.

To handle these aspects, GIO uses an AWS Elastic Load Balancer (ELB). An ELB allows providing one central access point on the Internet, e.g., https://stream.goedle.io, where all data is sent to but then forwarded to different machines to handle the incoming data stream. One can easily attach additional machines to an ELB as necessary. With AWS Auto Scaling, this can even be adjusted during different hours of the day. In the context of AWS, we also refer to Elastic Compute Cloud (EC2) instances when talking about machines behind an ELB. AWS offers different EC2 instance types for different purposes. Each EC2 instance typically has an Elastic Block Store (EBS) attached to it where the instances store data persistently.

Having a load balancer in place, multiple tracking instances receive all incoming requests in form of events as described above. These events are then stored persistently on an EBS device of the instance. These instances are operated by means of EC2 instances of small computational power but capable of high Input/Output (I/O) performance. The sole purpose of the tracking instances is to track the raw events and other machines are responsible for aggregating the data and storing the result in databases. Since each tracking instance has only storage for a limited time range, the data is copied everyday to a second storage without such data size limitations. When data is accessed for aggregation and further processing, the raw data is always accessed on this storage to avoid interference with the basic tracking. Obviously, this storage for all historic user data needs to scale as well. Here, we make use of AWS Simple Storage Service (S3) which acts essentially like an endless File Transfer Protocol (FTP) storage with additional encryption capabilities.

4.2.2 Reliability and Availability

Two additional reasons to use AWS are their reliability and availability. Amazon grants with their EC2 SLA[15] a monthly uptime percentage of at least 99.95%. With the given resources, it is impossible to build a comparable system that has the same reliability and availability guaranteed based on a bare metal solution. Especially when it comes to scalability, AWS supports building a system that easily can be scaled. In the case of increasing traffic, we have the ability to add more EC2 instances behind a ELB with the same availability guarantees. Furthermore, AWS provides high bandwidth and it is not necessary to maintain any additional, non-virtual, hardware. This means we do not have to care about any failure of hardware. S3 falls under the SLA as well and has additional guarantees when it comes to data persistence and data loss protection. So far, we have not experienced any data loss due to AWS failures and only rare cases of EC2 outages.

[15] https://aws.amazon.com/ec2/sla/?nc1=h_ls

4.2.3 Data Protection

Another important part of the tracking infrastructure is to protect data from unauthorized access. For this reason, we are not only using Secure Sockets Layer (SSL)/Transport Layer Security (TLS) to encrypt data at transfer, but also encrypt all raw data at rest, too.

Using SSL/TLS connections for all data transfer requires the purchase of different certificates. Right now, GIO has purchased certificates for different subdomains such as https://stream.goedle.io for secure tracking or https://api.goedle.io for secure data access. These certificates can easily be added to an ELB so that all traffic between a virtual lab and AWS is encrypted. Only key employees at GIO have access to the keys associated with the certificates to ensure that none of these keys are accessed by unauthorized persons.

When data is copied from the tracking instances and stored on S3 in a daily routine, we encrypt the data. Therefore, the data is useless if the necessary encryption key is not provided. To further increase the level of security, different keys are used. To implement such a system, we make use of AWS Key Management System (KMS). KMS helps to generate and manage encryption keys. Again, only key employees of GIO have access to the KMS.

4.3 Available Integrations

We wanted to keep all options open and provide a maximum flexibility for the tracking. Therefore, we decided to connect our tracking infrastructure with Google Tag Manager[16] (GTM). GTM is a client-side tracking aggregation service.

There are additional advantages of GTM. On the one hand, the tracking can be used for websites and mobile application at the same time without additional integration effort. On the other hand, we can change the tracking script without updates on the client-side. This also improves the maintenance abilities of the data collection process. GTM has become a very popular and widespread tool in particular for web-based applications.

4.3.1 Google Tag Manager

The Google Tag Manager structures a website or app as a container. This container has defined tags. A tag is a functional component which can be configured with rules. These Tags are connected to actions which take place on the website or app.

An action is initialized by a user or the execution of a script. Such an action is a predefined point in the source code. As an example, when the user triggers a "click"-event, GTM listens to the “click”-events and forwards the information about the executed status at the moment of the interaction. The user does not notice anything and instead, the website or app is working as before. In the background, an Hypertext Transfer Protocol request is fired with the attributes defined in the tag.

The process of collecting data from a user's click on a button to make it accessible is structured in three phases.

1. **Implementation of the GTM script into a website or a mobile app**: Google provides a code snippet which can be used to add the GTM functionality to a website or

[16] https://tagmanager.google.com/

mobile app. This snippet also downloads the GIO JavaScript which is used to forward the tracking information to the GIO infrastructure. To fire a custom event, which is not predefined by GTM, one has to add a tracking call at every position in the code. This tracking call corresponds to an event. Furthermore, a data layer is defined for the GTM. By default, it has no information but it can be enriched with information on the client-side. As an example, a timezone offset can be added.

2. **Building a tag and defining events which should be forwarded**: By default, the GTM can forward every action on a page. To have a preselection and to filter out useless information, we only track the events that we really want to use later for learning analytics. This filter can be implemented in the GTM dashboard. This can be achieved with a regular expression that contains the name of the valid events.
3. **JavaScript execution**: The data layer and the GTM information are forwarded to the GIO JavaScript, which provides a track method. This track method needs the data layer that contains all mandatory attributes. To have a reliable access to the GIO JavaScript, GIO decided to store this script with help of Amazon CloudFront Content Delivery Network (CDN)[17].

The implementation guide for using GIO with GTM for the ENVISAGE project was shared with all partners and has already been used successfully for the first implementation.

Since the JavaScript is accessible via CloudFront CDN, we can change the script and adjust it. Therefore, GIO can control the tracking and the output of the third tracking phase. Furthermore, the availability follows the Amazon SLA. In this respect, GIO is able to make this script accessible under common industrial availability standards.

4.3.2 HTTP-API

Although the GTM allows tracking in many different scenarios, we sometimes have the need for submitting user data without the GTM framework. One example of such a use case can be the backend of a virtual lab that also provides information on the performance of students. For such cases, GIO also provides a pure HTTP-API. This HTTP-API is able to receive all information defined in Section 4.1 . To send a request to the HTTP-API, it is necessary to authenticate at the GIO backend. An example HTTP-request could look like the following curl[18]-command:

```
curl
     -X POST -i
     -H "Content-Type: application/json"
     -H "Authorization: <HashValue>" -d '{
     "app_key":"1234",
     "user_id":"123451243",
     "event":"finished.lesson",
     "event_id": "12",
     "event_value": "65",
```

[17] Amazon CloudFront Content Delivery Network - https://aws.amazon.com/de/cloudfront/
[18] https://curl.haxx.se/

```
    "ts":123545612,
    "device_type":"desktop",
    "local":"de_DE",
    "geohash":"wp123234",
    "user_agent":"Opera/9.63 (Macintosh; Intel Mac OS X; U;
    en) Presto/2.1.1",
    "screen":"640x480",
    "timezone":3600000,
    "app_version":"1.1.1"}' https://stream.goedle.io/track
```

The body of the POST-request simply contains a JavaScript Object Notation (JSON)-object with the tracking data. Besides the body, the request needs two header fields, namely "Content-Type" and "Authorization". The former indicates that the request submits data in the JSON format and the latter field provides the signature for the authorization. Building the signature works as follows:

1. The JSON-object needs to be transformed to a string.
2. This JSON-string is concatenated with an API-key that we provide.
3. The concatenation of the JSON-string and the API key is then hashed with the Secure Hash Algorithm 1 (SHA-1).
4. The signature can now be used for authorization.

4.4 Historical Data Import

We do not only want to collect data from the tracking present in the virtual labs which we already started. Instead, there is sometimes another opportunity to enrich the data with historical information. The main reason to use historical data is to extend the observation period. Also with respect to WP3, the system should have the capability to inject historical data which was collected before the start of the real-time tracking.

A historical import requires data in our format. Before we can inject historical data, the data has to be transformed to the format described in Section 4.1 . Otherwise, the data cannot be used. This leads to mandatory information that historical data must have:

- Unique user identifier
- Timestamp when the event was triggered
- Event name of the action that was tracked

There are some pitfalls that we have to be aware of. In Section 5 we will see how the raw data is aggregated. Once this is done, there is an attribute that indicates the first occurrence of a user. This attribute is important to identify the user journey and provides the information at which time the user journey began. If we now want to use historical data, we have to take into account that the first seen event could be earlier in time. E.g., we saw a user on December 30th, 2016, but now we get historical data from November 11th, 2016. The user which was first seen by our system on December 30th, 2016, now shows up in this historical data set as well. This impacts two parts of the data process. The first seen attribute and the estimated sessions as described in Section 5.3.1 . In this simple case, we only get another session but this session is now the first session in the order and must be inserted before the already grouped sessions. Therefore, we have to start a recalculation of the user's sessions and set a new first seen attribute. This process has to be robust because there are different scenarios that can arise.

Table 4.1 *Possible historical import scenarios*

Historical Import Scenarios		
Event Sequence	**Old Sessions**	**New Sessions**
E1 - 20min - IE1 - 15min - E2	|E1|E2|	|E1-IE-1E2|
E1 - 5min - IE1 - 20min - E2	|E1-E2|	|E1-IE-1E2|
IE1 - 31min - E1	|E1|	|IE1|E1|
IE1 - 10min - IE2	-	|IE1-IE2|
E1 - 31min - IE1 - 31min E2	|E1|E2|	|E1-IE1-E2-IE2|

Historical data is not always data that is older than the first seen attribute. It can also be data that has been collected after the start of the tracking. This is the second pitfall where we not only have to be aware of the first seen attribute and the ordering of the sessions. We also have to be aware of the last seen attribute and sessions, which can be extended, due to interim events that affect the session grouping and connect two sessions for example. Table 4.1 describes some of the potential scenarios. The following information is necessary to read Table 4.1:

- EX: Old event X
- IEX: Injected event X
- E1 - 20min - IE1 - 15min - E2: This shows the event sequence by a user that has to be represented after injecting historical data. Where 20min/15mins is the time in between two events.
- |:Represents the beginning or the end of a session

4.5 On-Boarding Process

As a first step, we clarified with the consortium which actions are important and which platform will be used to implement the tracking. Therefore, we used the first project meeting to make an auditing for the tracking concepts and started collecting information which actions could be used with respect to an event-based tracking. Additionally, the event identifiers and event values were discussed.

One of the overall objectives[19] of ENVISAGE will be the future usage of the developed in an economic environment. It is important that the project results can scale in an economic context. A consulting-like approach with auditing and discussing each event in detail is a time-consuming process. To shorten the required time of such a procedure in the future, we started forging a process out of the experience we made during the first implementation. The result is a seven steps procedure[20]:

[19] This aligns with WP6.

[20] In later iterations, we should reduce and simplify this process even further. For a real world scenario, a one-click solution is a desirable goal.

- **Step 1**: Definition of the learning objective
- **Step 2**: Finding events that enable this objective
- **Step 3**: Definition of event identifiers and event values
- **Step 4**: Check access to these events
- **Step 5**: Create an unique lab identifier
- **Step 6**: Integrate and activate the GTM
- **Step 7**: Check data after the first day

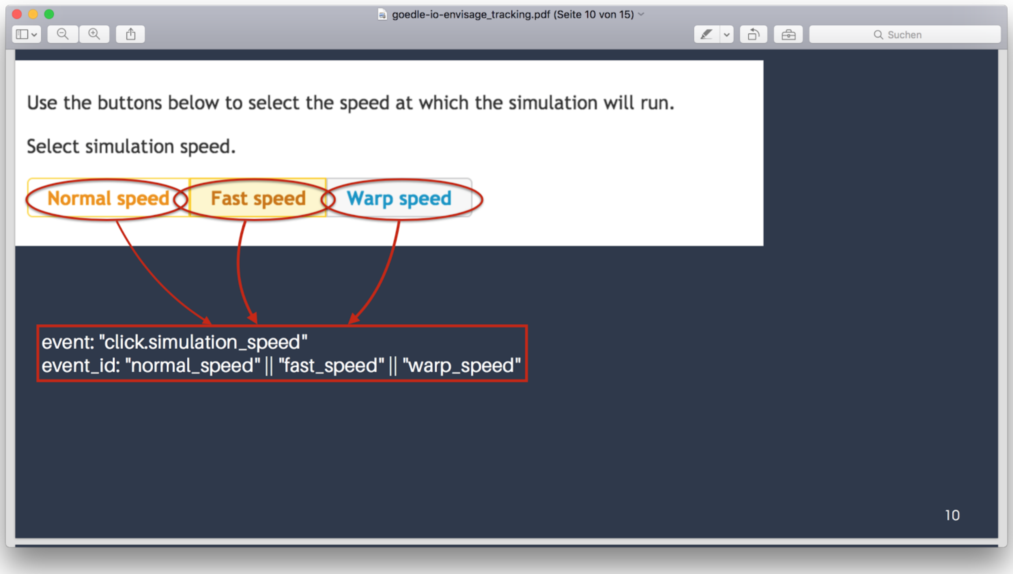
These steps will be used for the onboarding of labs in the future. Additionally, the tracking should directly use all events, which are created without additional configuration. To get custom events, the lab maintainer should implement the tracking calls at the appropriate positions in the source code.

4.6 Example Integration: Wind Energy Lab

The "Wind Energy Lab" from EA is an approved pedagogical tool which is actively used in school classes. EA was able to provide the consortium with the source code of this lab. A close cooperation among all partners of the consortium enabled an event-based-tracking tailored towards learning analytics. In detailed discussions, every event was defined and further specified as necessary.

The development of the final virtual lab is not finished at this point in time. To get an idea for the tracking and the data, we decided to use a lab, which is already used by EA in lessons. To accelerate the data acquisition, instead of the lab maintainer, GIO implemented the GTM tracking into the lab and modified it accordingly.

As one of the results, GIO provided the consortium with a list of proposed events for the initial tracking. Figure 4.1 exemplifies how this was done in the case of a tracking point for changing the simulation speed in the "Wind Energy Lab".


goedle-io-envisage_tracking.pdf (Seite 10 von 15)

Use the buttons below to select the speed at which the simulation will run.

Select simulation speed.

```
event: "click.simulation_speed"
event_id: "normal_speed" || "fast_speed" || "warp_speed"
```

Figure 4.1 *Tracking proposal for changing the speed of the simulation.*

The whole “Wind Energy Lab” was analyzed in this way and tracking points were defined. Afterwards, all events were summarized and the full list of events for the “Wind Energy Lab” can be seen in Table 4.2. The color code shows which events are implemented in the code already.

- **green**: is implemented and data was received
- **red**: is implemented but no data was received
- **orange**: is not implemented yet

Table 4.2 *Wind Energy Lab event definition*

Definition of Tracking Points			
Event	**Event Id**	**Event Value**	**Description**
view.instructions	<identifier_instruction_tab>		Event when user views the instruction tab
view.configuration			Event when user views the configuration tab
view.simulation			Event when user views the simulation tab
view.report_charts			Event when user views the report charts
increase.parameter	<identifier_configuration>	<|difference_old-new|>	Event when user increases a parameter configuration of the lab
decrease.parameter	<identifier_configuration>	<|difference_old-new|>	Event when user decreases a parameter configuration of the lab
click.simulation_speed	<identifier_speed>		Event when user adjusts the simulation speed
start.simulation			Event when user starts the simulation
pause.simulation			Event when user pauses the simulation
reset.simulation			Event when user resets the simulation

Definition of Tracking Points			
			configuration
add.turbine			Event when user adds a turbine
remove.turbine			Event when user removes a turbine
view.show_power_output_report			Event when user views the power output report
view.show_values_for_current_hour			Event when user views the report for the current hour
view.show_pie_charts			Event when user views the pie charts

For these events, the GTM snippet was added to the source code of the “Wind Enegery Lab” with the following data layer:

```
dataLayer.push({
     'event': < 'event.specifier' >,
     'goedle': {
          'timezone': < utc offset >,
          'locale': < language >,
          'device_type':  <  DeviceTypeIdentifier  e.g  mobile,
     desktop >,
}});
```

The data layer is necessary because we added the time zone offset to be prepared for exact time analysis in the future. We also added the locale to distinguish the information where people come from and which language they speak. Additionally, the device type is needed to distinguish between desktop and mobile users. We get the locale and timezone via a JavaScript method and set the device type to “desktop” because the lab is not mobile optimized and runs on desktop machines at EA. For future purposes, we will dynamically distinguish the device type programmatically.

This helped us to start gaining practical experience and to fulfill the requirements in the case of the defined metrics. The metrics are depending on the tracked information. For example, to calculate "time-on-task" metric, the starting and ending events have to be tracked based on a user action. We also have to take care of providing unique identifiers and mapping information so that categorization is possible. Every event has to be mapped to a single user.

5 Data Aggregation

The previous section described how all raw data is tracked and stored persistently on S3. The next steps merge all data from different tracking instances, store the intermediate result again on S3, and then aggregate the data. In a final step, this aggregated data is stored in a database.

5.1 Transformation of Raw Data

The tracking infrastructure receives all raw events in a continuous stream. As the current infrastructure is not a complete real-time system, the data needs to be processed in a batchwise fashion. GIO currently processes the data on a daily basis but can also imagine shorter intervals in the future to downsize the gap to a full real-time system. As described in the Section 4.2.1 , the data is stored across different instances and hence needs to be merged first. Then the raw events of all virtual labs need to be grouped in first place based on the particular lab and individual user. Both, the lab and the users, have unique identifiers for this purpose. Afterwards, the data is available for each lab and can be stored in a database so that further processing becomes easier.

At this point, different choices for the most appropriate database can be made. GIO has decided to first store the data in a Not only SQL (NoSQL) database that is highly scalable. In particular with a small team, easy scalability had highest priority as different labs and learning apps can quickly grow. For this purpose, DynamoDB[21] within AWS has been picked as the database of choice. Each user is stored with their metadata and their events in different DynamoDB tables.

While this format may not be adequate for every machine learning algorithm yet, it satisfies a number of requirements so that further processing is possible and generation of features for supervised and unsupervised learning is made easier.

Multiple EC2 instances are required for merging the data from different collectors and importing the data in a parallelized fashion into DynamoDB. The reliability and availability is of great importance, especially because of the complex interaction between different services. Therefore, the data which is stored in DynamoDB is automatically and synchronously replicated across three facilities in our AWS region. This protects us against failures of individual machines and even failures on facility level.

5.2 Enriching with external data

Often, additional data is freely available that makes it possible to further enrich the data. A simple example are geo-locations that can be inferred from the IP address. This often goes down to a latitude and longitude level. However, one should also mention that the quality of such a resolution is often limited. Such data will at least allow to compare performance on a country or city level.

In the case of learning analytics, one can further imagine enriching the data additionally by, for example, obtaining demographic information or social status. For many regions in Europe

[21] https://aws.amazon.com/dynamodb

there are statistics available on creditworthiness and wealth. Other statistics have also shown a correlation between education and social status. For example, it is often reported that income and education of parents correlate with the educational level of children. When now comparing results from virtual labs of students in different regions, census data can be integrated into the analysis. With deep analytics, this correlation and its impact can be further analyzed.

5.3 Aggregation and Categorization of User Data

Often certain aggregated statistics are of interest on a lab or user level. For example, it is often useful to obtain retention information for a lab. Retention matrices or graphs visualize how often students return to a particular lab. On the user level, it is often more helpful to have user events grouped by sessions or to obtain a set of days on which a student worked with a lab. As mentioned above, pedagogical metrics also ask for a different aggregations or categorization of the users. Labeling the students as high, medium, or low performers can also be done during this step.

5.3.1 Calculate sessions

While it might seem trivial on the first look to group user events into sessions, a more detailed look at the problem shows that many approaches can be taken. Typically, a web-based lab does not recognize if a user makes a break or leaves the lab. Therefore, the most simple form of session calculation is using a fixed duration which is assumed to be a break. As a starting point, we currently assume that 30 minutes without an event mark the break between two sessions.

In a more sophisticated and data-driven setting, one would look at all inter-session times for events and determine a 95-percentile or similar, to obtain a lab-specific value. For example, some virtual labs require more reading in between solving tasks. Therefore it might look like that a user has left the virtual lab already. If the student then becomes more active again, it looks like a separate session. But in fact, the time between two consecutive events was just not long enough. On the other hand, if we pick that value to generous and the same machine is used by different classes in short period of time, the sessions of different students (or groups of students) are considered to be same, again introducing noise into the data.

6 Data Augmentation

The goal of the ENVISAGE project is to provide insights to virtual lab designers, creators, and maintainers that go beyond shallow analytics and today's state-of-the-art, for example, a travel-path-analysis of students or an automatic segmentation based on performance and/or behavior. This information cannot be read off from the raw events but instead requires prior processing and computations on the data. The insights expected from this augmentation also go beyond the simple methods that were described in the section on data aggregation.

In many cases, a first step is the generation of features and corresponding feature vectors per user. The process of converting the existing data into features that are understood by machine learning algorithms is often referred to as feature engineering. Feature engineering is an area within machine learning that is very time consuming, requires expert knowledge, and is often impossible to automate.

Once an extensive set of features has been developed, an intensive preprocessing is typically required. This can range from simple binning, normalization, and scaling, to more complex approaches such as dimensionality reduction. Many of those tasks depend on the algorithm that is used later on. For example, if an algorithm is used with a Euclidean-distance function, one needs to transform a categorical into a numerical representation. Additionally, values often need to be scaled so that the distance function returns the desired results.

Once data is in adequate form, the corresponding algorithms can be applied to the user data. However, in a supervised learning setting, a model needs to be learned first before predictions can be made. Typically, a learned model is first verified on a small test set to assess its performance.

If we want to provide an end-to-end solution to virtual lab managers, all the steps above have to be run in the cloud on a regular basis as well. Once a model is learned, the model is applied to all new users that qualify for a certain prediction. The results of the predictions are stored in the database as well. Here, a NoSQL database is not as well suited because we often want to query only predictions for users satisfying certain conditions. GIO therefore runs a Structured Query Language (SQL) database in parallel where particularly the results of the machine learning algorithms can be stored and queried are more flexible compared to a NoSQL database. For the same reasons as in the NoSQL case, an AWS solution is used for the relational database. To be more precise, a PostgreSQL instance is run within the AWS Relational Database Service[22] (RDS). The next section will describe in greater detail how predictions for users can be accessed.

Example: Let us assume that we want to automatically cluster students depending of their problem solving capabilities. For this purpose, we only take into account students that have successfully solved a particular task. We can then preprocess the data of those users and run a clustering algorithm such as k-means on the data. We might identify two different groups of users that we label with two different names. We then add this information to the users in the database in order to store this information persistently.

[22] https://aws.amazon.com/rds/

Such computations are typically more computationally challenging, and hence more Central Processing Unit (CPU) and Random Access Memory (RAM) intensive so that a different layer is used in the cloud. We would not want to burden an instance that is running for a different purpose with this load, e.g., a tracking instance. We therefore pick for such operations instances with multiple cores and higher amount of RAM but potentially lower I/O performance.

The process of feature engineering can become quite time consuming. Sometimes features, or entire sets of feature vectors, can be cached and used for different algorithms or for hyper-parameter optimization. In those cases, we can save time by not computing all features repeatedly but instead use a cached version of the data. For this purpose we make a copy of each dataset and store the encrypted data on S3 as well. If such a cached dataset exists for a specific deep analytics pipeline, it can easily be downloaded in a fraction of the time that is needed to generate the data from scratch.

7 Data Access

For different purposes, data needs to be accessed in different forms. The most basic form is raw data access. Such an access returns the data in the very same form as it has been tracked on the original device. I.e., it does not contain any additional information and is not aggregated. Without further processing, this form of data has limited utility - in particular for teachers.

Following the processing layers that were described so far, the next option is to return data in aggregated form. Shallow analytics dashboards typically show such information on an app level. For example, the daily active users of the past days or the average time for solving a particular task. This data can either be shown in a separate visualization tool, e.g., a web-based dashboard, or, as proposed in section 3.4.5 in deliverable D1.2, the analytics data can be shown directly in a distinct frame in a virtual lab. This would require specific endpoints in the GIO infrastructure that return aggregated information providing value to different groups of users, e.g., the teachers or the students.

Lastly, it can also be interesting to access data on a user level with all its metadata and inferred data. I.e., once particular users of interested have been identified, the infrastructure should allow returning all of their data. Among other things, this includes all event data in raw form of that particular user, its calculated sessions, metadata, and predictions from the deep analytics part.

To realize those different forms of data access, the final component of the GIO infrastructure is an HTTP-endpoint that provides a JSON-API. As it will be described in the examples below, various URLs return data in JSON-format that satisfy different needs. To realize this API, the GIO infrastructure again makes use of an ELB with several EC2 instances processing the different requests. Depending on the nature of the request, the API-instances query the different databases (DynamoDB and PostgreSQL) or the object storage S3. The API uses different authentication mechanisms so that only authorized users can request data. The data itself is returned decrypted, however, by using an SSL/TLS connection, GIO makes sure that nobody has access to the data during the transport to the end user. We will now describe the different data access option in greater detail.

7.1 Raw Data Access

A first requirement for data access was in form of a raw data access. The raw data helps each consortium member to get a better understanding of the data and provides test data for initial development. This data can also be used to start with an initial visualization.

An example event was already given in Section 4. The raw data access makes all events available per virtual lab per day.

The API needs to guarantee that only authorized users have access to the data. Therefore, a user has to authenticate at the API with their credentials. The credentials are in form of the identifier of the lab and a master key. The master key is shared in a securely manner among the consortium members via a password manager.

Example: the simplest approach of downloading raw data is by using a system tool such as “curl”. Having a lab identifier and a master key at hand, the data can be downloaded for a particular day with a command of the following form:

```
curl
     -H 'content-type: application/json'
     -H 'X-goedle-app-key: <LAB_ID>'
     -H 'X-goedle-master-key: <MASTER_KEY>'
     https://api.goedle.io/apps/<LAB_ID>/data/<YYYY>/<MM>/<DD>
```

7.2 Aggregated Data Access

Depending on the requirements of the data visualization, one can imagine to also return aggregated data on the virtual lab level. Shallow analytics may require direct access to some aggregated information such as daily active users or retention information. However, these endpoints will be developed as requested. Often this information can be accessed from elsewhere and the focus here is on data that helps designing and adapting the virtual lab. When these requirements arise, the development can be discussed and realized if manageable.

7.3 User Level Access

When accessing the raw data, the data has to be further processed in order to obtain meaningful information for a teacher or a virtual lab designer. As discussed in the previous sections, the data aggregation, enrichment, and augmentation provides additional information. For this purpose, one can also query all available data that exists for an individual user. This includes, but is not limited to, session information, locale data such as language and country, device type, and of course, the event history as well. Once a full data augmentation layer is in place that augments the data with deep analytics, this information can also be accessed per user.

Example: Akin to the previous example, one can again make use of system utilities such as “curl” to access this data. It is again assumed that the user has a lab identifier and the corresponding master key at hand. Additionally, the user identifier must be specified.

```
curl
     -H 'content-type: application/json'
     -H 'X-goedle-app-key: <LAB_ID>'
     -H 'X-goedle-master-key: <MASTER_KEY>'
     https://api.goedle.io/apps/<LAB_ID>/users/<USER_ID>
```

Depending on the scenario, we will also add query parameters that let you return the entire history of events as well.

8 Extending the existing infrastructure for ENVISAGE

While some parts of the infrastructure and data collection code already existed, we started to extend the infrastructure and its tracking capabilities explicitly for the ENVISAGE project.

As seen in the Section 2, many of the existing analytics and data tracking tools do not provide the necessary functionalities and flexibilities to effectively support virtual labs as we envision it. Based on these observations, we are currently extending all aspects of the platform to fit well into the ENVISAGE scenario.

Tracking

We are extending the tracking code to provide specific functionalities to track groups of students or classes. This includes, for example, asking students for an identifier, so that a student returning to virtual lab several days later again be identified. Similarly, an identifier for classes or schools can be incorporated.

However, we have to be aware of more strict privacy regulations when tracking students and underage persons. It is important, that we do not start collecting any personally identifiable data. Looking at the identifier above, it should be a unique identifier, however, the identifier should not immediately allow a mapping to the student's real name. Under certain circumstances, this can be desirable for a teacher. Nevertheless, this mapping should not exist on third-party infrastructure services, nor be used in any other way or connected to commercial applications. This will become an issue as soon as we start categorizing and grouping users with respect to the metrics raised in D1.2. This was one request brought up by the teachers from EA and needs to be handled in accordance with privacy laws.

Many of the virtual lab scenarios are focused on a specific goal, i.e., are compared to arbitrary apps much more goal oriented. We are currently extending our tracking infrastructure to provide specific support for such events and traits attached to it. For example, it should be possible to distinguish intermediate goals and final goals.

Aggregation

When adding the input and tracking of student identifiers, the data received by the backend potentially requires deduplication. If students will be given the possibility to choose a username, it is often desirable to align the events tracked before the login with the events after the actual login, so that a complete picture is available. I.e., when students first visit the virtual lab, they have a different identifier, as they have not entered a username yet. Once they login, a more persistent identifier can be attached to their events although the previous events belong to the same users and sessions.

This also connects to the privacy issues mentioned in the previous tracking paragraph. We have to find an anonymous identifier, that is capable to be used over the whole targeted educational scenario, which potentially lasts over several weeks, and still preserves privacy rights.

Augmentation

Only very few analytics tools have started to augment user data with predictions. Typically, those tools do not grant access to the predictions directly but allow using the information in subsequent tasks. For example, in marketing suites for e-commerce applications, marketers can send an email to all users who are predicted to be more likely to have a purchase in the near future. Here, the augmentation of the users is in form of a probability or score that indicates the purchase likelihood. In the scope of the ENVISAGE project, we will create a framework that generally allows to augment student data with predictions and other automatically generated traits that are useful for learning analytics use cases.

The algorithms that augment data can either run directly on the GIO infrastructure, or the API can be used to augment individual users based on external calculations. In the first setting, the source code of the algorithms will be deployed in the GIO infrastructure, i.e., the data augmentation layer, and the algorithms run in a regular fashion. For example, an algorithm clusters all students every day based on all available behavioral data and performance indicators. Here, one can for example analyze how certain task solving skills correlate with the performance.

In the latter case where the algorithms do not directly run on the GIO infrastructure, API endpoints will be created in such a way that external information is attached to existing users. For example, by providing data access to raw data, other project partners may apply algorithms of this data themselves and feed the results back into the GIO infrastructure so that other algorithms can make use of these predictions as well. To give a more concrete example, one can think of a setting where supervised learning algorithms run directly on the GIO infrastructure that predict which users will solve a particular task. Additionally, an external algorithm calculates a clustering of the students based on their behavior and feeds the information back into the GIO infrastructure. The supervised learning algorithms can then use the cluster label as an additional feature. For the visualization, all this information can then be returned with help of the GIO API and summarized in either the virtual lab itself or in the authoring tool.

Access

So far, the GIO API does not provide a lot of endpoints for data access and the API is primarily used to feed GIO's own products and data analysis. Most of the endpoints are not yet tailored to work with virtual labs or the authoring tool but they will be adapted accordingly, so that the desired information is made available to those parts of the workflow. This will in particular focus on supporting the needs of learning analytics specific use cases. When looking at many other available tools on today's analytics market, such data access in different forms is not be possible. This holds in particular if the raw data is enriched and augmented as described in the previous sections.

To start with the first analysis and development of algorithms, we have added an API-endpoint for raw data access as described in the previous section. This API-endpoint is made available exclusively to the ENVISAGE partners and GIO does currently not plan to make such functionality available to all customers due to the reasons described above. This policy might change in the future if the demand increases from tech-savvy customers.

We are currently working on API-endpoints that return user data and their events as sketched in the previous section. For the moment, this will omit the augmentation based on deep analytics as this is still work in progress. However, once the consortium has developed the algorithms and technology, the endpoints will be adapted and extended accordingly.

This can also lead to additional endpoints that return data of entire groups of students. For example, all students having a particular cluster label.

Depending on the needs of the shallow analytics visualization, we will also provide additional endpoints that return aggregated data. This can include but is not limited to simple counts and statistics such as daily active users or monthly active users, retention matrices, or similar metrics that can also be found in today's state-of-the-art analytics products.

9 Statistics on Tracked Data

For a first overview of the tracked data we provide descriptive information about the data. From December 30th to March 13th we observed 92 users. These users had about 178 sessions.

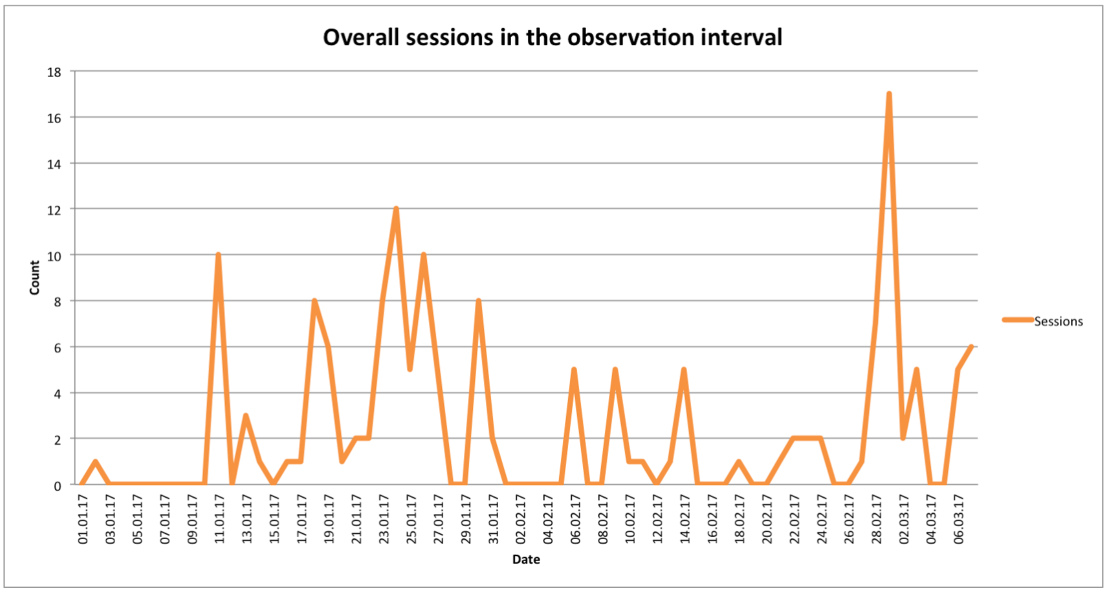


Figure 9.1 *Overall sessions for the tracked Wind Energy Lab users*

To dive deeper into the tracked data, we will focus on days where more than five sessions were measured. The following evaluations were done on the dates in Table 9.1. To get a first understanding how the lab is used, the Table 9.1 shows the days which had more than five sessions.

Table 9.1 *Days with high usage*

Date	Session Count
2017-03-01	17
2017-01-24	12
2017-01-11	10
2017-01-26	10
2017-01-18	8
2017-01-23	8
2017-02-28	7
2017-01-19	6

We found eight users which were seen on one of these particular days and also returned on another of these days. Only two users returned more than two times. At this point, we assume that the returning users are specific devices which are used for the “Wind Energy Lab”.

A first hint about the usage is the occurrences of the used events in Table 9.2.

Table 9.2 *Event occurrences*

Event Count	Event
393	view.configuration
289	view.show_power_output_report
148	view.show_values_for_current_hour
137	add.turbine
116	start.simulation
89	click.simulation_speed
81	launch
65	view.instructions
60	view.show_pie_charts
57	remove.turbine
24	view.home
23	reset.simulation
19	pause.simulation

One can see that there are only a few events which are used frequently. If we take into account that the launch event occurs only once per session, the events with a higher count than launch are events of the "Wind Energy Lab" which are needed to solve the lab.

In order to extract information for additional deep analytics and showing the practical implementation of the metrics in D1.2, we introduce the graph in Figure 9.2. This graph is depending on user sessions and the overall connection of events fired by users. The graph shows the connections for subsequent events in time grouped by users. The result is the connection between all events for all users in the "Wind Energy Lab". The nodes of the graph are events, while the numbers on the edges count the frequency of transitions. The thickness of the edges is relative to these counts.

To visualize this information, we used the force atlas 2 algorithm to arrange the nodes in the graph. A very interesting side effect is the closeness of nodes that represents the interactions of the users. It can be seen that the "simulation event" nodes are well connected and arranged nearby. The same can be seen for the "report event" nodes. This graphs shows that travel-path related metrics are possible even with simple transitions. Although not yet shown here, the tracked data can also be used for time-related metrics. However, metrics such as time-on-task or time-to-completion require additional tracking points.

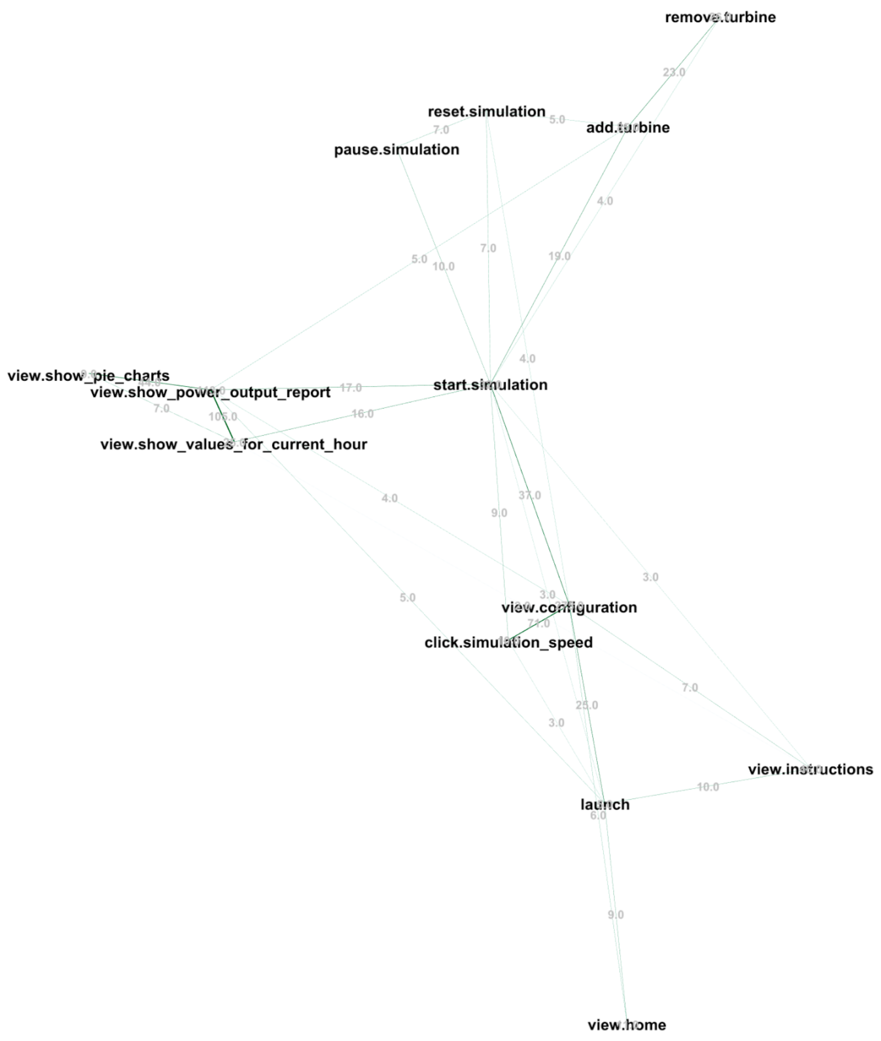


Figure 9.2 *Transition graph*

Other metrics also suggested a grouping or categorization of users. For this purpose, we have had a closer look the IP addresses of the users. We were able to identify four central IP addresses whose source was in Greece. The main traffic came from EA in Athens. Another source of traffic was CERTH (Thessaloniki), where the lab was tested by other members of the consortium. This information was actually confirmed during the second project meeting in Athens (March 9th-10th). We were also able to resolve the following cities in Table 9.3. This table shows the top five producers of events:

Table 9.3 *Geo locations of tracked events*

Event Count	Country	Region	City
924	Greece	Attiki	Athens
168	Greece	Attiki	Marousi
94	Greece	Attiki	Marousi
39	Greece	Attiki	Athens
39	Greece	Thessaloniki	Thessaloniki

This data showed that the tracking and the integration is working properly. The data was accessed via the JSON-API which is described in Section 7.