Design Of Specialized Storage for Heterogeneous Project Data

Olga Kalyonova, Nurbek Akparaliev, Ivan Perl
ITMO University
Saint-Petersburg, Russia
ovkalyonova@corp.ifmo.ru, n.akparaliev@gmail.com, ivan.perl@corp.ifmo.ru

Abstract—**This article describes an approach for designing a storage system that will allow persistence of large number of heterogeneous entities, used to describe software development process in various process supporting tools (bug trackers, repositories, wiki engines, etc). Goal of this work is to extract core semantics essences from entities like issues, code commits, builds in continuous integration systems and others and persist them together in one indexable, integrated and searchable systems disregarding from which tool each of them came from. Taking into high level of variety in data shapes and forms, requirement to perform complicated cross-entities queries and potentially high volumes of data - designing a storage for such systems turns into a challenging task.**

I. Introduction

Nowadays the software development takes control of new positions in various fields of activity ranging from a cash register system development for stores to high-tech fault-tolerant solutions for aerospace equipment. In terms of modern solutions, reasons for data storage are growing rapidly every day, because it is important and necessary[1]. Data is the important information such as the records of the clients,records of the company's finances, particulars of the project and much more. All of the current records trend to place a lot of prominence on the IT infrastructure that serves and store sit. There are several causes why to store data[4]:

1) Optimize a cooperation. For example, cloud storage is a perfect instrument for immediate data exchange.The option to give access to multiple people makes this service a perfect tool for both distant and in-house work.
2) Create a backup for files to prevent accidental deletion.
3) Storing an information in a specific form, for instance, storing daily regime depending on the season or marks on the map.

A. The problems of information glut in the software development industry

However, storing data in the database itself does not give all the advantages that the modern software development industry would like to have. In the life of many software projects, there may be crisis situations due to too much poorly organized data. The growth and evolution of any software project inevitably leads to an increase in the number of artifacts that accompany the development process. In the event that no preventive measures are taken to organize the design data, the following problems may arise:

1) Difficulties in project management, clumsiness.
2) Unpredictability of terms and increase in the number and cost of risks.
3) Staff flow due to a project that is too difficult to understand.
4) Overstating the threshold of entry into the project, which aggravates the hiring of new staff after their outflow.
5) Increase the overhead of tasks.

As you can see, simply storing project data in a normal database will solve these problems quite difficult. According to IBM [5], at the moment the business is experiencing explosive growth in the volume of information, which is approximately 45% per year. It is important to note that only 20% and the notion of structuring does not mean that it is somehow used to analyze and control the process.

In the software development industry, it is most important to focus on the following cases when analyzing design data for qualitative improvement of software development processes and reducing the risks associated with information singularity:

1) Forecasting and an auxiliary estimation of time for the decision of a problem at planning. It is generally believed that the work takes all the time allotted to it and even more. Nevertheless, effective planning of both short-term and long-term tasks is extremely important for the comfortable and effective development and stable growth of project quality. Effective planning also helps to prevent economic risks.
2) Prioritizing tasks and tracking their status. As a rule, modern software projects usually specify a certain amount of tasks, which must be polled for a particular period of time. However, there are frequent situations where the importance, and hence the sequence of tasks is not known: insufficiently mature project management or the customer can not distinguish the main thing and indicates to all tasks the maximum priority. Unfortunately, this approach only irritates the developers and does not help to cope with their tasks, and therefore - increases the risks.
3) Building a transparent and manageable process. Frequent are the situations when the project is developing too quickly and the practices of project management do not have time to evolve with it. This leads to a decrease in the effectiveness of the project. At the same time, if management makes timely notice, it will be natural to improve the process. The projected process based on the chosen methodology should

be as transparent and natural as possible for both the team and the leadership in order to reduce the overhead of team time and improve the understanding of the development vector and the current state for each participant in the program project. This aspect becomes especially relevant when the number of project team members increase 7 people and more.

4) Tracking the process between related projects. As follows from the previous point, the growth of the project is related to the increase in project tasks, the number of people involved in the project, and the appearance of related projects. In a modern software project, starting from the smallest one, from the very beginning, there are links and dependencies on external projects, and therefore development teams: whether internal or external. In this regard, one of the areas that require special attention are the processes of interaction between teams and projects.

B. Problems in designing a project data store

When designing a project data warehouse designed to improve the understanding of the processes of developing software projects, the following problems may arise:

1) A large volume of hard-to-analyze heterogeneous data.
2) The data can be located on a large number of sources (task trackers, repositories, wiki and project documentation).
3) Some data for analysis can not be stored in the database (due to size, format or corporate policies).
4) Necessity for each project to complete the logic of data analysis from scratch as a separate application to the database.
5) Overhead costs for administration of the project data base.
6) To significantly improve the process due to data analysis, a highly skilled analyst (data scientist) is needed to identify hidden dependencies and correlations.

II. Requirements for the storage of heterogeneous data

A. Requirements based on problems

Based on the described problems, we formulate qualitative and functional requirements for the storage of heterogeneous data:

1) Quality:
   a) It is designed for increasing and potentially large amount of data.
   b) Has the basic re-used logic.
   c) Lets you access resources that are stored outside the database.
   d) Minimize the configuration parameters.
   e) Retrieves data from external sources, which by themselves do not have the functions of connecting to the projected storage.
   f) The data can be not full and fusion.
2) Functional:
   a) Tracking process

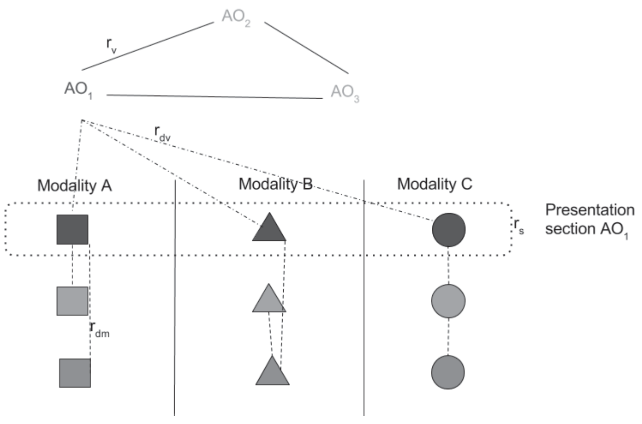


Fig. 1. The structure of multimodal data

   b) Help in prioritizing tasks by identifying strongly and loosely coupled tasks.
   c) The visibility of results calculated for end users.
   d) Tracking a given process by creating practices in the context of the selected software development methodology.
   e) The ability to analyze complex structures, such as several related projects.

B. Multimodal data approach

Based on the nature of heterogeneous project data, for their presentation and analysis we use the approach of multimodal data.

This approach can be applied due to the fact that heterogeneous data includes the notion of heterogeneous data.[2] Under multimodal object in this article we will understand an object or process, which is described with various characteristics of different modalities. Under modality we will on assume one of more attributes which describe a specific characteristic of an object. Set off characteristics of the same modality will be called essential presentation. It is not mandatory that each multimodal object exists in all modalities.

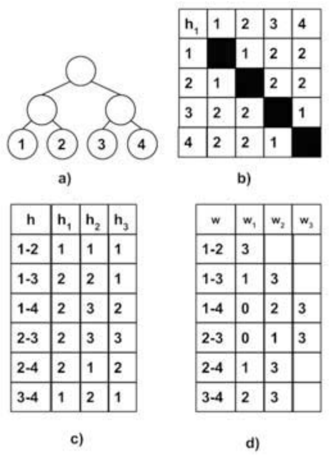
By their nature, heterogeneous objects are described with number of attributes which are not compatible with each other, for example colour, weight, price, and these characteristics can be treated as modalities of this particular object. When object has multiple modalities describing it, it can be called a multimodal object.

Previously, a method for representing multimodal data was developed [3]. The use of the multimodal data approach is aimed at the first stage of the implementation of functional requirements and allows:

1) Obtaining samples of complex objects, for example, tasks. In the near approximation, this will allow you to set an approximate estimate of time or complexity for new tasks. This can also be used to construct statistical samples or calculate the required resources.
2) Identify the cyclical dependencies in tasks. This property can be used to prioritize tasks and track interaction processes between teams. In some cases,

it can be applied to solve the problem of multiple dependencies in assemblies.

3) Check the correctness of the state of the object. For example, if an object is in a certain state, then it must have one or another artifact (build log if the assembly was run or a report after running test scenarios if the task is in the "Tested" status). This ability will allow validating the process of developing and adapting new project practices and software development methodologies. In addition, this ability can indirectly simplify the interaction between development teams. In the example with test scenarios: if the task is returned to the development, it will be useful to immediately see the logs of the failed test scenarios.
4) Organize, present and analyze complex task structures. A common practice is to use Jira's auxiliary plug-ins (such as Structure) to organize complex task hierarchies. Another example is the sequence of assemblies of the branch project intended for solving the problem.



h_1	1	2	3	4
1	■	1	2	2
2	1	■	2	2
3	2	2	■	1
4	2	2	1	■

h	h_1	h_2	h_3
1-2	1	1	1
1-3	2	2	1
1-4	2	3	2
2-3	2	3	3
2-4	2	1	2
3-4	1	2	1

w	w_1	w_2	w_3
1-2	3		
1-3	1	3	
1-4	0	2	3
2-3	0	1	3
2-4	1	3	
3-4	2	3	

Fig. 2. Meta-data of multimodal data: a) - hierarchical graph of modality, b) - first strategy of storing data of common parent height, c) - second strategy of storing data of common parent height, d - table of coherence power.

C. Description of multimodal data approach

As already described above, the concept of multimodal data assumes the existence of a set of objects, generalized by specific characteristics. A set of such objects is shown in Fig. 1. One of the main goals of the multimodal data representation method is to find the correlation in the data set and consists of 4 main stages.

1) Data input. At this stage, data entry and initial data cleaning are assumed. An example of initial data cleaning is the removal of duplicate objects.
2) Classification of the essential presentations within the modalities. As a result of this stage it is supposed to obtain hierarchical graphs for all modalities, as shown in Fig. 2 (a). Particular attention should be paid to the fact that the classifier must be specific for each modality. Hierarchical modality graphs are essentially directed graphs that can be reused when adding new data. Each vertex of such a graph is an essential representation of one or several multimodal objects.
3) Calculation of height of a common parent. At the entrance to this stage, hierarchical graphs of modalities are transmitted. For each pair of vertices of the graph, the altitude of the common parent is calculated by the max-minimal strategy (maximal from minimal). Additionally, at this stage, the parameter h can be specified, which limits the search height of the common parent: if a value is reached and the common parent is not found, the search stops and it is considered that the common parent does not exist. As a result of this stage, a table is created for each modality, which reflects the values of the heights of common ancestors for pairs of elements. Regarding the storage of the values of pairs, one of the strategies can be chosen:
    a) The first strategy involves obtaining a table of heights of essential presentations for each modality separately. An example of obtaining a height table for any modality Fig. 2(a) is shown in Fig. 2 (b). A distinctive feature of this table is that the number of rows and columns is equal to the number of essential presentations, and this number can be quite large. The second feature is the symmetry with respect to the main diagonal. The table constructed in this way, as we see, is analogous to the adjacency matrix. In the event that we do not store duplicates, the table can take the form of a triangular matrix.
    b) The second approach is to build one table for all modalities. This approach is shown in Fig. 2(c). In this case, the number of rows of the table is equal to the number of pairs of objects, and the number of columns is equal to the number of modalities. This approach can provide an easier way to organize data, so that adding new elements does not result in the appearance of new columns. With this approach, we do not store duplicates. An additional advantage is the possibility of buffering or paging the parts of a complete table in contrast to a huge table of adjacency.
4) Calculation of coherence power. This step is performed in order to identify strongly coupled multimodal objects according to a given formula. The result of the work of the stage is a table, similar in structure to that shown in the Fig. 2(d). You can determine the most related objects from this table by minimizing the column number and maximizing the value in the column. For example, the most strongly related objects in Fig. 2(d) between objects 1 and 2. The maximum value in a column in the simplest case is equal to the number of modalities. As the column number is increased from left to right, the binding conditions are weakened and the associated elements become larger: in this case, first a connection is added between objects 1-3, 2-4 and 3-4, and then all the others. After the second column, a graph can be constructed as an Fig. 3.

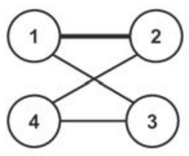


Fig. 3. Resulted graph after second iteration

D. Features of the projected storage using the multimodal data approach

Due to the fact that the multimodal data approach was chosen for design of the heterogeneous data storage, we can specify and detail the qualitative requirements for the designing storage.

1) You should choose strategies for implementing the method of multimodal data representation, which will be designed for:
   a) Not a static, but a dynamically increasing set of data;
   b) The number of objects can be significant. Even if there are only 100 multimodal objects in the set, for example, tasks, and approaches and formulas will not be chosen correctly, it can significantly increase the overhead of both time and environment in which the projected storage will unfold.
2) The implementation of the multimodal data representation method should be implemented separately and independently of the data being processed. Exceptions can be made only by the classifier, which are connected separately depending on the modalities being processed.
3) When developing the basic logic, it is necessary to take into account that the multimodal object may contain not the most essential presentation, but a reference to it (depending on the modality settings).
4) Setting parameters such as h_{term} and w_{min}, as well as formulas for calculating w, should be as transparent as possible from the user, and ideally hidden from him.
5) An auxiliary module is needed to extract data from external sources and transmit them to the first stage of the multimodal data representation method.
6) Modalities should be configured in such a way as to take into account the lack of data. The problem of fusion data is partially solved in the first stage and is completely solved by using the multimodal data approach.

Functional requirements with a multimodal approach were considered in section II-C.

III. Designing a storage based on requirements

A. Choosing a general idea

Let us now try to formalize the data essence with which we are dealing:

1) First of all, these are the objects of the project data. This is the most important component for the data storage. These data are heterogeneous by nature. Cases when the project data is homogeneous will be considered as a special case of heterogeneous data. Project data will be collected from various project tools using a data collection module (pumps) and will be distributed to separate entities as independent elements, that is, there will be no external relationships between entities. For communication between entities there will be a higher abstraction like the work item.
2) Multimodal project data objects. Such objects are the union of a certain number of essential presentations. As multimodal objects for working with project data, we introduce the concept of a work item, which is a meta-object that includes the essential presentations for such modalities as repositories of various kinds, task trackers, continuous integration servers, test reports, etc. In addition, such an object may not be characterized by artifacts of modality, such as, for example, name and status. Essential presentations in such modalities can be given externally or obtained from other modalities. As mentioned earlier, it is important to have a setting that allows missing values of essential representations, or vice versa, to always require them.
3) Meta-data of multimodal data approach. There are three types of entities:
   a) Hierarchical graphs of modalities, the number of which is equal to the number of modalities. Depending on the implementation, in some cases, when adding a new element, it may be necessary to partially rebuild the tree, but this will help to maintain the correctness of data and calculations when adding a significant part of new objects.
   b) Table of heights of common parents. The number of such tables can be equal to the number of modalities, or 1, but with the number of columns equal to the number of modalities - this depends on the approach that was chosen in Section II-C.
   c) Table of coherence powers. Like the table of heights of common parent choosing the second strategy, the coherence power table stores meaningful information in columns, not in rows. The number of such tables can vary depending on the chosen strategy:
      i) Store one table and update the values in it when applying the formula.
      ii) Prohibit changes in the calculation formula for binding forces and store one table.
      iii) Store all tables created when the formula is changed.

In addition, it is important to remember the auxiliary modules, which were declared in the requirements of functional in Chapter I-B (1) and quality (2).

1) Implementation of the logic of the method of representation of multimodal data. A module with this functionality should also support the ability to connect classifiers that have a modality-specific imple-

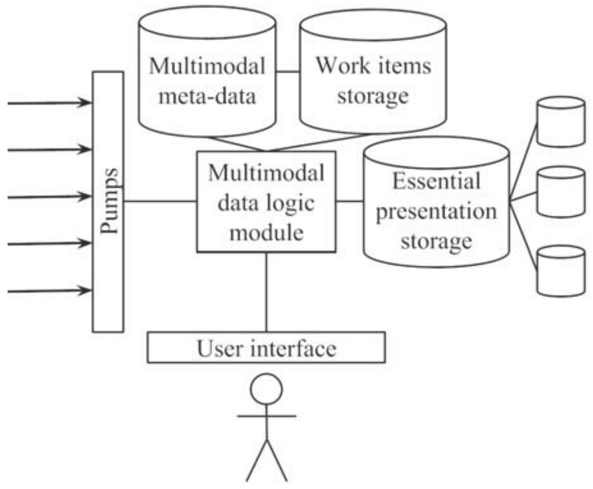


Fig. 4. The general structure of designing architecture

mentation.

2) Module for obtaining data from external sources (pumps). This module will provide the ability to obtain data for the processing of design data by connecting to their sources, among which can be:
   a) Bug trackers and task trackers (Mantis, Jira, Trello, etc.);
   b) Code repositories (SVN server, GitLab, GitHub, Bitbucket);
   c) Continuous integration servers (Jenkins, Travis);
   d) Documentation (documentation servers, wiki, confluence).
3) User interface module. This module is extremely important, because the storage and the results of its analysis are aimed at reducing the threshold of entry into the processing of design data and solving problems with the need for a qualified scientist in data analysis. In addition, the interface should be as clear as possible in order to reduce the overhead of user training and maximize team productivity through the tool. The toolkit should simplify their work, rather than create a new one. As the Agile Manifesto states, the fundamental document of the most popular methodology for Agile software development today is ”People are more important than tools”, and therefore the tool being developed should be people-centered.
4) Under the hood of the user interface, we will ask tricky queries to the database, that is, we will have our own DSL query language. It allows to retrieve entities in many different combinations.

Considering the data types described above and the dedicated modules, we will compile the architecture shown in Fig.III-A.

Next to the data storage of essential presentations, auxiliary storages are depicted, since some data cannot be stored in the database, and possibly only where they already exist. In this case, the database stores only pointers to where you can find path with index to where you can find the required data.

B. Architecture overview

Based on the previously obtained data-based architecture, it can be concluded that a system similar to the data warehouse was developed. Consider this similarity in more detail. As a rule, the data warehouse has the following properties:

1) Subject-Oriented. In this case, there is indeed a focus on the given specificity, since the storage is projected for a clearly described range of tasks and solutions to these problems. However, the data storage can be used in any software project without significant changes in the architecture. This provision was stated in paragraph II-D.
2) Integration. In the context of the data warehouse, speech generally refers to the consistency of storing heterogeneous data. In our case, this condition is fulfilled by using the multimodal data approach.
3) Time-variant and Nonvolatile. In the context of the task being solved, it was decided not to comply with these provisions due to the lack of storage of the historical sequence of data - it should be the same for the selected development process - and even vice versa: there is a great need for up-to-date data. Refusal of these requirements will also help to reduce the overhead of maintaining the environment of the developed storage.
4) Summarized. In this case, the development store will provide significantly more functionality than the summable sections of the stored data.

C. Selecting the types of specific data storage

When the general architecture of the data storage was designed, and it became clear that independent storage of various parts in separate stores under the control of the meta-platform is supposed to be independent, it is necessary to define specific types of data storage for each kind of entities defined in Chapter III-A. When selecting the storage type, we will pay extra attention to the following parameters:

- Model of work with data;
- Target data type;
- Possibilities for profiling and not targeted storage of data, if it does not correspond to data entities directly;
- Scaling options;
- Availability of implemented databases of a given type and specialization.

To date, there are quite several different types of databases. Among them we distinguish the following:

1) Relational databases (SQL)[13]
   - Advantages: commonly used database, a lot of capabilities(ex. for loop and functions), easy to maintain
   - Disadvantages: very slow database, requires a lot of effort to adjust a database, consumes a lot of memory and CPU time
2) NoSQL[14]

- Advantages: elastic scaling, sizable data, flexible data models, cost effective(allows to process and store more data at a much lower cost), requires less management on entities, very fast database
- Disadvantages: no advanced expertise

3) The graph storage[15]
   - Advantages: flexibility, deep fast search, indexing
   - Disadvantages: processing high volumes of transactions, not good at handling queries that span the entire database, not optimized to store and retrieve business entities
4) Column Databases[16]
   - Advantages: this database is good at queries that involve only a few columns, aggregation queries against vast amounts of data, column-wise compression
   - Disadvantages: this database is not good at incremental data loading, online transaction processing usage, queries against only a few rows.

Evaluating the pros and cons, NoSQL with high speed and reliability best fits to store data in our work items storage and in essential presentation storage, and Column-based database greatly fits to place multimodal meta-datas in storage.

To store essential presentation and work items we have chosen NoSQL database for the following reasons:

- project meta-data items weigh a few kilobytes, so it is better to store them in random access memory(RAM).
- for flexible schema, as we want to aggregate information from different systems, like JIRA and Trello, entities will have dynamic properties.

Comparing RAM with solid state drive(SSD) and hard disk drive(HDD), if we need access time with the smallest milliseconds, our only choice is RAM. SSDs also match this criterion, but only in terms of seek time. However, in typical workflow, to process a single user request, not one, but several seek operations are often needed, and then, due to some additional requirements, there may be extra requests to disk. Additionally, random access to slow storage is more costly, and sequential access to slow storage is cheaper. Thus, slow storages are made for sequential access. To reiterate, if we need to sequentially access massive datasets and are satisfied with speed of 100 MB/s, the most cost-effective storage is an HDD. It's 10 times slower than RAM with respect to sequential access time, but 100 times cheaper as well. This means that in terms of bytes, we could buy 10 times more free HDD space than RAM with the same total throughout. In most cases, It's important to note that we are talking only about our situation here. If you operate on a bigger scale, the price may have a higher priority. Just the same, modern NoSQL databases store data in RAM.

Based on this, it is logical to store data in RAM and HDD or SSD will be a place where you can periodically drop the state so that you can reboot the server or upgrade it. To solve this problem, modern NoSQL databases like Redis or Tarantool provide WAL and Snapshot mechanisms to maintain the integrity of the database, first to store in a single file all queries that change database and to prevent an accumulation of this transaction log file, then make a snapshot to save a copy of database in hard disk. Server can be crash in an unexpected time, and all data can be recover from log file and last snapshot.

To store data, first we should choose a data model. There are a lot of types of models like key-value, column-based, document-oriented, graph based and so on, in terms of heterogeneous project data, when some of our entities are constant, other are flexible, key-value data model greatly fits our needs. Key-value pairs map nicely to programming language data types, so each NoSQL database creators tries to put a connector to most common high-level programming languages together with the database. Considering the nature of multimodal data and what was said in Chapter III-A, we concluded that column-based data model greatly fit to store data in module of multimodal data.

Some NoSQL databases stores key-value pairs as tuples, therefore, there may be several keys and values. Since we are dealing with tuples and lots of keys, there can also be many indices for making the necessary queries. As relational databases NoSQL also provides different types of indexes for a much quicker search in data, most of them implement hash, tree, rtree, bitset indexes, to work with data ranging from bit data to MD5 hash to store user passwords[12]. Variety of indexes gives us not only a diversity of data that we can keep, also gives more speed in storing heterogeneous data.

Imagine for a while what will happen if we choose a relational database or graph database instead of NoSQL in this case. Regarding relational database, this type of database stores data in a hard disk and uses RAM to store hot frequently used data. This way of storing data may seem better, hard disk is cheaper and necessary information in RAM, but what will happen if we want to fetch rarely used data. At first relational database will check a hot data in RAM, then go for data to hard disk. This request can be characterized as a long one, because database server checks two places including slow hard disk, then updates hot data in RAM[11]. If we compare NoSQL and Graph databases, on the one hand NoSQL stores sets of disconnected aggregates, one the other hand Graph database can not handle massive sizes like NoSQL, it just retains minimum sizing at a greater depth of data[10]. In this project case, to connect efficiently disconnected sets, we implemented WORK ITEM storage. To sum up, nowadays NoSQL databases bring together many interesting solutions offering different data models and database systems, each more suitable than traditional SQL solutions for certain use cases and shapes of data. Honestly, relational database, NoSQL and graph databases developed greatly, and can compete each other on the same level, the choice between the three will rectilinearly depend on data size, the speed you need, how much you are willing to spend on hardware.

D. Selecting specific databases

To select specific implementations of the given database types obtained in Section III-C, it is necessary to conduct a study and determine which databases are most suitable for the current architecture and tasks, based on the requirements previously obtained. In total it is required to select 3 databases for the following entities:

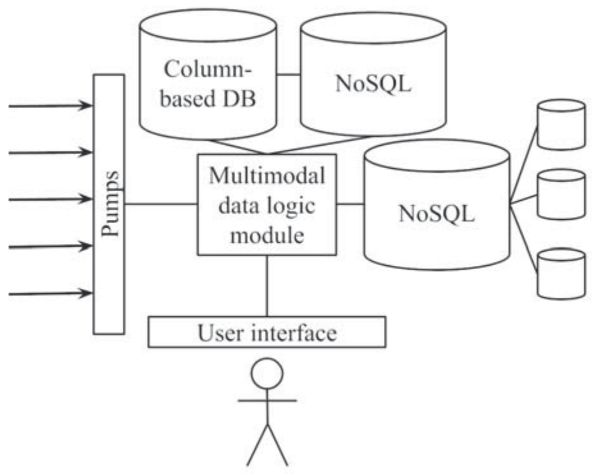


Fig. 5. The general structure of designing architecture with type markers

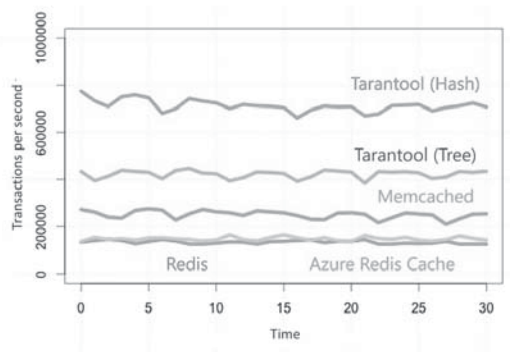
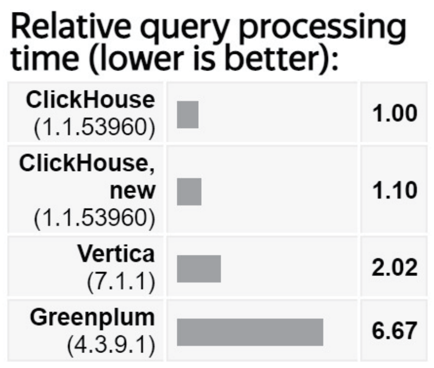


Fig. 6. The compatibility of NoSQL in-memory data storage



Relative query processing time (lower is better):

ClickHouse (1.1.53960)	▬	**1.00**
ClickHouse, new (1.1.53960)	▬	**1.10**
Vertica (7.1.1)	▬	**2.02**
Greenplum (4.3.9.1)	▬	**6.67**

Fig. 7. The compatibility of column-based data storage in set of 1 billion dataset size

1) NoSQL database for essential representations;
2) NoSQL database for multimodal objects;
3) The column database for storing the metadata of the multimodal data representation method.

When choosing the implementations of storages of the specified types, we select the following priority characteristics:

1) Speed of access to data;
2) Scalability;
3) Data backup;
4) Compatibility with high-level languages (the presence of connectors);
5) The possibility of expansion plug-ins;
6) The availability of transactions

Since all modern NoSQL and Column databases correspond to 2-6 points, we tried to compare databases at speed. Based on benchmarking and results available in various articles[17][18], Tarantool NoSQL storage and ClickHouse Column database show best results in all classes of operations under different workloads. In-memory and column-based databases have high potential and are highly optimized to reduce execution time for our case.

E. *Proof of concept*

To test the resulting architecture, the first prototypes of the auxiliary modules were developed, and the necessary copies of the databases were deployed. As a source of test data, projects with an open license were selected, which provide access to their bug trackers and repositories. For testing, the following projects were selected in increasing size and popularity:

1) IoT-Log Merger (bitbucket, trello);
2) Taskodrom (github, github issues);
3) Mozilla Firefox (github, bugzilla).

Based on the functional requirements that were described here and here, the following results were obtained:

1) The ability to predict the complexity of tasks is implemented. For example, for the first project, the task in the status ”In Progress” with the active commit to the repository is an average of 4 hours, in the second project - 10 hours, and in the third - 93 hours. Explain such terms can be the experience of teams, the size of tasks and the complexity of their implementation.
2) The tasks in the status of new were analyzed and the priorities for 20%, 5% and 60%, respectively, were corrected. Let us explain the numbers obtained. In the first case, the team is still young and badly worked, and therefore frequent mistakes are made in setting priorities. The percentage is rather low, since the tasks themselves have relatively small project size, small team size and low connectivity. The second, more experienced team can more precisely prioritize tasks and is already well-versed in the project’s links. The complexity of the third project is very high, and tasks, as can be seen from the previous paragraph, are done for a long time. In this regard, priorities are not always placed in an optimal way.

3) The results of the system work are graphical graphs that are graphical and easy to analyze in the construction of related tasks. Relying on research in the field of multimodal interfaces and visibility of the analysis of visual data by a man, this method allows increasing the speed of a person's work with data at times.
4) To verify the implementation of practices, the documentation was examined. For the first project, as practice, the practice of adding a reference to a task to commit-message was chosen, and for the second and third project - the practice of adding screenshots to bugs in the UI. The percentage of implementation of practices in the 1st project - 66% is explained by an inexperienced team with a not followed process. The percentage of compliance with practices in the second (82%) and the third (98%) is due to the increased level of team competence and clearly established practice in the process of teams.
5) To test the possibility of analyzing complex structures it was possible only on the data from the third project. In this case, the data from different components were analyzed. The results of the analysis were used to obtain samples from the 1st and 2nd points.

IV. Conclusion

As a result, a system was obtained that satisfies all the claimed functional and qualitative requirements that were formulated on the basis of problems of the software development industry.

1) Qualitative requirements were implemented as follows:
    a) In the multimodal data presentation method, the second approach for storing tables of heights of common parents was chosen, which allowed to work with projects where the number of tasks is more than 10,000 (the 3rd test project, the number of tasks in the "New" status).
    b) The selected databases have the highest scalability and still retain the highest data access speed.
    c) Special approaches to the implementation of the logic module for multimodal data.
2) The implemented logic can be re-used by replacing the minimum number of elements:
    a) Settings of the pumping unit (pumps);
    b) Setting up and finalizing the required classifiers.
3) Access to external resources is realized as follows:
    a) Implemented a module for pumping data (pumps);
    b) Implemented the possibility of remote data storage of various modalities with storage of the path to them in the developed data store.
4) To implement the connection to remote instruments, specific parts of the pump's module were implemented.
5) The configuration parameters are configured independently by the system based on the internal logic but can be changed by the user if necessary.
6) To work with incomplete and mixed data, the multimodal data approach was used. In addition, when implementing and configuring the multimodal logic module, the settings were used to consider the possibility of missing data.

Functional requirements have been fully implemented and tested on three projects of different size and popularity, the results of which are described in Section III-E.

Based on the results of the design, we can conclude that the developed system for storage and analysis of heterogeneous multimodal data of software development projects:

1) Reduces the difficulties in project management due to the visibility of the current status of both tasks and the project as a whole.
2) Increases the accuracy of the forecast for estimating the time of the task execution based on the analysis of the timing of the implementation of similar tasks.
3) Reduces the outflow of team members by increasing the visibility of their work.
4) Reduces the threshold of entry into the project, due to the linking of different systems into one work item that is easy to analyze. It helps newcomers to instill project practices through automated control of their implementation.
5) Reduces the overhead of software development teams due to the visibility and uniqueness of the manipulated object of the development process.

A. *Practically meaningful benefits*

What is the benefit? Multimodal data practices provide methods for building multidimensional correlation graph of multimodal objects. In our case – work-items are multimodal objects and we can find links between almost any two work items in the project. When you are working in a big and distributed project – this will help to easily find required related works to what are you doing, find all issues which processing was making history of files you need to update and will significantly improve project data integrity. Work flows will become much clearer and this will enhance development process in general.

All these things may appear related to SEMAT and Essence. Majority of projects which currently exist are using millions of home-brew processes with unique configuration of practices and practices implementation. All these projects are using almost endless list of supporting tools and their combinations and configurations. Main goal of our solution is to provide a unified overview on top of this unique configurations. These configurations are important because they were designed for the very specific needs of each project. Unified view on the top is required to provide easy navigation in project data. Interesting point here is that these unified entities are easy to manipulate. It is much easier than getting into each tool or its configuration and extracting data from there. We already solved this. And on the top level we may define unified practices in the project based on unified entities. So, for each already running project we can define its virtual SEMAT/Essence project representation.

As a result, our solution can be used as a tool for performing estimation and actual migration from existing process to SEMAT/Essence. Having monitored unified entities project can define what practices they want to have and what they need to implement. This is a kind of design mode. Than, they may start performing actual process transformations in their project applying selected practices in a tools level. In the same time, they will see how all these changes are reflected on the level of unified entities in our system. At the end of the transformation, implementation on the tools level should match process defined on the level of unified entities.

REFERENCES

[1] Importance of Data Storage and Backup, Web: https://www.slideshare.net/JohnWard23/importance-of-data-storage-and-backup-50883225

[2] Kalyonova O., Perl I. *Introduction to Multimodal Data Analysis Approach for Creation of Library Catalogues of Heterogeneous Objects* // Proceedings of the 21st Conference of Open Innovations Association FRUCT - 2018, pp. 327-331

[3] Kalyonova O., Perl I. *Revealing of Entities Interconnections in System Dynamics Modelling Process by Applying Multimodal Data Analysis Paradigm* // Proceedings of the 21st Conference of Open Innovations Association FRUCT - 2017, pp. 156-161

[4] *5* reasons why you should use cloud storage every day, Web: https://federalnewsradio.com/commentary/2017/03/5-reasons-use-cloud-storage-every-day

[5] *I*BM Corporation *The explosive growth of Information.* Web: https://www.ibm.com/ru/events/ox2010/pdf/3.pdf

[6] A. L. Thompson, D. Bohus., *A Framework for Multimodal Data Collection, Visualization, Annotation and Learning.* Web: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/annelo-de102-thompson.pdf

[7] *A*. Kashnikov, L. Lyadova, *Integration of heterogeneous sources of data based on recurrent decomposition*, International Journal "Information Technologies Knowledge", 2011

[8] *W*.H. Inmon, *Building the Data Warehouse.* Canada: John Wiley Sons, Inc., 2002.

[9] *G*. M. Faruk Ahmed, Md Shoriful Islam, Molla Md Rezaul Karim, *Comparison Between Inmon and Kilball Methodology for the Purpose of Designing, Constructing and Testing of a Commercial BIDW Project*, International Journal of Computer Graphics, 2017, pp. 11-20

[10] *N*eo4j, Concepts: NoSQL to Graph, Web: https://neo4j.com/developer/graph-db-vs-nosql

[11] *N*eo4j, Concepts: Relational to Graph, Web: https://neo4j.com/developer/graph-db-vs-rdbms/

[12] *T*arantool: in-memory DBMS and application server, Web: https://medium.com/@Vadim.Popov/tarantool-in-memory-dbms-and-application-server-64d60ffa1d6e

[13] *W*hat are the advantages and disadvantages of SQL? Web: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-SQL

[14] *A*dvantages and Disadvantages of NoSQL databases, Web: https://www.hadoop360.datasciencecentral.com/blog/advantages-and-disadvantages-of-nosql-databases-what-you-should-k

[15] *T*he Good, The Bad, and the Hype about Graph Databases for MDM, Web: https://tdwi.org/articles/2017/03/14/good-bad-and-hype-about-graph-databases-for-mdm.aspx

[16] *W*hat's Unique About a Columnar Database? Web: https://www.flydata.com//blog/whats-unique-about-a-columnar-database/

[17] *V*. Abramova, J. Bernardino, P. Furtado, *Experimental evaluation of NoSQL databases*, International Journal of Database Management Systems

[18] *P*erformance comparison of analytical DBMS, Web:https://clickhouse.yandex/benchmark.html