Degree project

UpTime 4 – Health Monitoring Component

Author: Dmytro Ledinov
Supervisor: Jonas Lundberg
External Supervisor: Kaj Nystrom, Combitech AB

Date: 2013-09-30

Course Code: 5DV00E, 30 credits
Level: Master

Department of Computer Science
Abstract

The problem of monitoring and analysis of performance parameters of complex program systems is really important nowadays. By monitoring a complex system during its operation lifetime it becomes possible not only to obtain a set of different performance parameters, software and hardware characteristics, but to make an analysis of them to determine potential risks, errors and reasons of their appearance.

This master thesis is aimed to solve this problem in terms of a complex document-management platform. The goal of thesis work is to develop a holistic health and performance monitoring component that could be integrated into the platform. The project was developed in collaboration with the Swedish company Combitech AB.

Keywords:
Monitoring, analysis, performance parameters, software and hardware characteristics, complex system.
Table of Content

1. Introduction .................................................................................................................. 1
   1.1 UpTime Platform ........................................................................................................ 1
   1.2 A Monitoring System for UpTime ........................................................................... 2
   1.3 Goal of Master Thesis and Requirements .............................................................. 2
   1.4 Restrictions ............................................................................................................... 2
   1.5 Report outline ........................................................................................................... 3

2. Preliminary Research on the Solutions ....................................................................... 4
   2.1 Method of Research .................................................................................................. 4
   2.2 Existing Ways of Solving the Problem ..................................................................... 4
   2.3 Initial Requirements as Criteria of Solutions Selection ......................................... 4
   2.4 Description of Existing Solutions .......................................................................... 5
       2.4.1 SiSoftware Sandra ......................................................................................... 5
       2.4.2 Everest ............................................................................................................ 5
       2.4.3 Windows Task Manager .................................................................................. 5
       2.4.4 SQL Suite ........................................................................................................ 5
   2.5 Comparative Analysis of the Selected Solutions ...................................................... 6
   2.6 Argumentation and Conclusions ............................................................................ 6

3. The Ways of Building the Monitoring Component .................................................... 7
   3.1 The Ways of Data Collection .................................................................................. 7
   3.2 The Ways of Data Aggregation ................................................................................ 7
   3.3 The Ways of Data Presentation .............................................................................. 8
   3.4 Conclusions ............................................................................................................. 8

4. Platform Architecture ................................................................................................ 9
   4.1 Platform Architecture without Monitoring Component ........................................... 9
       4.1.1 Anonymous Service ....................................................................................... 10
       4.1.2 Authoring Service ......................................................................................... 10
   4.2 Platform Architecture Including the Monitoring Component .................................. 10
       4.2.1 Interaction with Authoring Service ................................................................. 11
       3.2.2 Interaction with Database Server ..................................................................... 12
       4.2.3 Interaction with Operating System ................................................................. 14

5. Monitoring Component Architecture ......................................................................... 15
   5.1 Backend Architecture ............................................................................................. 15
   5.2 Frontend Architecture ............................................................................................. 16
   5.3 General View on Monitoring Component’s Architecture ........................................ 17

6. Detailed Internal View on the Monitoring Component .............................................. 19
   6.1 Typical Use Cases of the Monitoring Component ................................................. 19
   6.2 Static Structure of the Monitoring Component ...................................................... 21
   6.3 UI Specification ....................................................................................................... 23
   6.4 Involved Development Tools .................................................................................. 26
       6.4.1 Development Environment ............................................................................. 26
       6.4.2 Programming Languages ............................................................................... 26
       6.4.3 Frameworks and Technologies ...................................................................... 26
   6.5 Data Analysis and Priority of Results .................................................................... 27
7. Evaluating the Monitoring Component ................................................................. 29
8. Conclusions .................................................................................................................. 32
  8.1 Achieved Results ....................................................................................................... 32
    8.1.1 A Working Monitoring Component ................................................................. 32
    8.1.2 High Performance of the Monitoring Component ............................................ 32
    8.1.3 Decrease of Laboriousness ................................................................................ 32
  8.2 Future Work ............................................................................................................... 32
References ....................................................................................................................... 34
1. Introduction

Nowadays humanity has reached the stage when the most part of data is stored on computers and other electronic devices. Such way of storing gives a list of advantages against papers but still the necessity of having convenient and fast data processing tools remains.

That is why program platforms which are aimed to work with huge amounts of data are being developed. These platforms provide their customers with a list of services for storage, accumulation, extraction of data which make the work more convenient and productive.

The complexity of such systems lies in a large list of services provided by them and in a huge amount of data which is stored and processed. Such complex systems are database based. They require constant maintenance and monitoring to reach the highest level of performance and to avoid redundancy. That is why some piece of software or a program component that would allow administrators to make a special monitoring of performance parameters during the whole lifetime of the system is necessary.

Such system is developed by a Swedish company called Combitech AB. The product that is developed by this company is called UpTime.

1.1 UpTime Platform

UpTime platform is a document-management system which provides a set of services to its customers. A typical customer of UpTime is an organization that has a lot of documents or information organized in other way and wants to manage this information. Sometimes it is necessary to provide a customer with data storage, but usually all data is private, so it is stored in customer’s storage. Management services are provided by the platform. By using these services it becomes possible to create different kinds of new documents or look through existing ones, edit documents and delete them. In general, with the help of UpTime, it becomes possible to organize the data, import new information, edit or delete it. The platform includes a hierarchy of custom inner types like “Word” or “Phrase”. These types are responsible for a document formation. They contain a list of various attributes which help to build a document, determine the dependencies between the monitoring component parts inside it and organize fast mechanisms of searching. Three types of users of UpTime could be described:

1. Main user – is able to create, look through, edit and delete documents.
2. General user – is able to look through the documents but not allowed to create, edit or delete them.
3. Administrator – is a kind of super user which has the full list of rights. This user is a company-developer representative. The role of administrator is to work with the list of settings for the platform which are specified for each customer individually.

The goal of UpTime is to make document management easy and convenient for its customers.
1.2 A Monitoring System for UpTime
Combitech has many customers in different parts of the world. Such famous companies as Volvo and Bosch are among them. That is why it is really important to increase quality of the given service day by day. The company has a big customer support department which helps customers to solve problems related to the product and to handle their questions. Customers call a support service when something goes wrong or does not work or the work of platform is not satisfying. The process of searching for the probable reason of such undesired behavior requires a lot of actions which are mandatory to obtain all the information to make the analysis of the problem. This process is really time-consuming.

Nowadays the fourth version of the product is being developed. All given reasons explain necessity of having a program component. Such monitoring component would also be useful for UpTime and would avoid a time consuming searching of each required option, and provide useful information that helps to eliminate the problem in short terms.

1.3 Goal of Master Thesis and Requirements
The main goal of the Master Thesis is to design and implement a monitoring component for UpTime platform.

The solution which is considered to be used for monitoring must satisfy the following requirements:

1. Monitor and accumulate information about performance parameters of the platform’s server.
3. Monitor and accumulate information about platform’s logs.
4. Analyze accumulated information to find critical errors and warnings.
5. Provide centralized access to accumulated information, so it becomes possible to see everything “in one place”.
6. Present obtained information in a human-readable format, grading potential problems by severity.
7. Provide remote access to the services and data.

To satisfy these requirements it is necessary to perform a research on the following questions: what are the ways of collecting of performance parameters, what is the way to aggregate them, how is it possible to present a huge amount of data in a user-friendly way?

1.4 Restrictions
There is a list of restrictions that must be taken into account:

1. UpTime platform is a Windows based application which is built on .Net framework. That is why the monitoring component must be implemented by using capabilities of this framework.
2. UpTime platform operates with Microsoft SQL Server 2008 database. That is why the monitoring component is supposed to be able to interact with such type of database.

3. The monitoring component must be represented as a web-page and provide user interface to access its services.

1.5 Report outline
In Chapter 2 a preliminary research on existing solutions is described. The searching criteria is specified. The advantages and disadvantages of found solutions are discussed. This chapter contains the results of comparative analysis and conclusions about the relevance of a new development.

Chapter 3 describes UpTime platform itself and the monitoring component in details. The architecture of the platform and the interaction between its key elements is discussed.

In Chapter 4 the internal structure of the monitoring component is described. This chapter provides a couple of figures and diagrams which help to understand how the component is organized.

Chapter 5 explains which development tools and frameworks were used during the implementation.

In chapter 6 the experiment of component’s performance measurement is described.

Chapter 7 contains the conclusions and recommendations about possible future work and improvements.
2. Preliminary Research on the Solutions

Before starting development of a new program component it is necessary to look for existing solutions which are aimed to solve the same problem, to make an analysis of their advantages and disadvantages, to see if they cover all the requirements and also to make a conclusion about the relevance of development of a new solution.

2.1 Method of Research
The research and analysis steps should be based on an appropriate method. This method should take into account all singularities of the research subject. The basic method of preliminary research is the following:

1. Determine the set of initial requirements and tasks which must be satisfied.
2. Examine different ways of solving tasks. Pick out advantages and disadvantages.
3. Determine the criteria of selection.
4. Select existing solutions which can fully or partially match the determined criteria.
5. Examine an alternative approach about the usage of several solutions at the same time to fulfill the criteria. Pick out advantages and disadvantages.
6. Make a conclusion about research results.

2.2 Existing Ways of Solving the Problem
There are several ways of solving the problem of performance parameters monitoring:

- Perform manual monitoring using the capabilities provided by the operating system.
- Perform monitoring using special tools and software utilities which provide appropriate services.

Both approaches have their advantages and drawbacks. Manual monitoring of the performance parameters using capabilities provided by the operating system requires the involvement of system journals and trackers, a work with register and properties of different entities. It also requires some skills and knowledge to perform the actions about data obtainment. It is also difficult to analyze the disparate data. However there is no need to install additional software and to pay for it.

Monitoring with the usage of special tools is easier to perform. The obtained data is presented in a human-readable way and it is rather convenient to make an analysis based on this data. The main drawback of this approach lies in the potential lack of required services and as a result in the lack of some important data.

2.3 Initial Requirements as Criteria of Solutions Selection
In order to evaluate existing solutions we must have concrete list of requirements to which they should be compared. It is important that the criteria was selected after the interview with future users of the component. Such a lot was printed in Section 1.3. In practice each task in Section 1.3. comes with a number of sub-tasks. However, we have a chosen level of abstraction in tasks formulation that allows you to understand what the software solution should be able to do.
2.4 Description of Existing Solutions
According to the problem that must be solved, a number of different existing solutions that fully or practically fulfill the criteria could be found. The list of solutions was obtained by using search engines of Google with specifying the criteria as keywords. I have managed to find only four, but this list could be possibly extended with other solutions. However, obtained solutions are enough to make appropriate analysis. The description of each solution is given below.

2.4.1 SiSoftware Sandra
The abbreviation “Sandra” stands for System Analyzer Diagnostic and Reporting Assistant [1]. This program consists of 70 utilities for obtaining information about components of a personal computer and producing recommendations about the improvement of their work. Each utility is “responsible” for obtaining information about a separate device of the computer. It provides user with data about its manufacturer, version, date of manufacture, performance characteristics and many others. The information obtained with the help of this solution could be useful in measuring the platform’s performance parameters.

2.4.2 Everest
Everest [2] is a program for monitoring the information about hardware and software configuration of computers. It provides users with detailed description of motherboards, processors, video and network adapters, input-output devices, ports, externally connected devices, energy management, network connections and resources, users and their groups, software and operating system generally. Also this program gives recommendations about the actions to increase the performance of separate components.

2.4.3 Windows Task Manager
Windows Task Manager [3] could be also considered as a possible solution. Being a component of an operating system it can provide information about all processes that are currently running, about resources that are consumed by these processes: CPU time, physical, paged and virtual memory, network utilization and many others.

2.4.4 SQL Suite
SQL Suite [4] is a set of module tools for work with SQL database servers which includes: Trace Analyzer, Deadlock Detector, Ultimate Debugger and Heartbeat. Users can select tools which are the most appropriate for solving their problems. This solution could be very useful for database administrators and developers, because it is designed to eliminate database problems and make settings.

SQL Trace Analyzer identifies bottlenecks, performs diagnosis, reveals inefficient SQL queries and solves database performance issues. Using SQL Trace Analyzer it becomes possible to obtain the list of the most frequently running stored procedures and control the consumption of resources.

The goal of SQL HeartBeat is also different kinds of diagnosis and bottlenecks detection. It can determine bottlenecks of CPU, memory and input-output threads.
2.5 Comparative Analysis of the Selected Solutions
To compare selected solutions to each other, the criteria list was made. If the solution satisfies the criteria, it is marked with “Y” in the comparison table, otherwise it is marked with “N”.

<table>
<thead>
<tr>
<th></th>
<th>Sandra</th>
<th>Everest</th>
<th>Windows Task Manager</th>
<th>SQL Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>System performance monitoring</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Database server performance monitoring</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Log listening</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Detection of errors and warnings</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Grading of problems by severity</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Remote access</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 2.1 Comparison characteristics of the solutions

Each of described above solutions is highly specialized. It becomes the reason for some disadvantages that are rather critical and important. Each solution provides a couple of requested services but none is able to satisfy all requirements. Sometimes the software has to be platform specific and process some internal data like application logs in UpTime. It is a requirement to have a remote access to all information while existing solutions work as desktop applications and can be accessed only directly. In the particular case for Combitech this option is extremely important because the company has offices in different cities of the world. The other problem is financial costs. It is necessary to pay for a license to use most of these programs.

2.6 Argumentation and Conclusions
The results of the analysis showed that all existing solutions are able to cover only a part of requirements. Having a list of useful services they also have a lot of disadvantages which are inadmissible.

The usage of several solutions in combination would allow to satisfy more requirements. However such approach will not cover the full list of them. It is also important that the usage of several solutions at the same time becomes more complicated and inconvenient for the user. Also, several licenses would be required.

All these arguments suggest that it is necessary to start development a new solution that will definitely satisfy all the requirements and will be a useful tool for performance monitoring.
3. The Ways of Building the Monitoring Component

As it was mentioned in chapter 1.3 research on the ways of organizing the monitoring component is required. It is really important not only to satisfy the requirements, but to do it in the most effective and right way. Existing questions and problems will be solved with the help of research.

3.1 The Ways of Data Collection

The core responsibility of the monitoring component is to collect data. The data could be of different size and type. It could be located in different “corners” of the system and be related to a set of different instances. But at the same time all these factors should not affect the monitoring component and decrease its performance.

There are several ways to collect required data. Each of them has its advantages and drawbacks, but the compromise should be found.

1. The usage of API provided by third-party tools and libraries – such approach would provide an easy way to retrieve information which is necessary. The main advantage of third-party APIs is the simplicity of work with them. But there is also a huge drawback: you don’t know how the work is organized from the inside. The performance could be low and inadmissible. Also it is almost impossible to find such library that would allow to obtain the full list of required parameters.

2. The usage of native API provided by the operating system – such approach represents a low-level interaction with hardware with the help of operating system capabilities. The main benefit lies in the performance of such interaction. Native APIs work really fast, but they are rather complex to use and require appropriate knowledge and experience from the developer.

3. The usage of WMI - such approach would provide a rich instrumentation to work with performance parameters. WMI wraps the native API. The information provided by WMI is richer and a lot easier to get. The filtering syntax of WMI can dramatically reduce the amount of code that a developer has to write. The performance of WMI is lower than the performance of native API, but a trade-off in favor of WMI could be done because of all its advantages and benefits.

3.2 The Ways of Data Aggregation

The monitoring component is considered to operate with a huge amount of data. That is why it is necessary to aggregate it in some way. The aggregation of data would allow to separate information about different instances from each other.

There are several approaches that could be applied:

1. No aggregation – such approach is possible but is not the best one. When there is no data aggregation it becomes difficult for the developer to present the entire mess of information to the user. It also becomes difficult to maintain the system, though it works fine.

2. Aggregation by type – such approach would allow to separate data of different types from each other. As a result it would become easier to operate groups of
data dependent on their types. But still the data would remain fragmented and difficult to present.

3. Aggregation by groups of general instances – this approach is the best between the given three. Every performance parameter belongs to some instance. Every instance has a set of such parameters. It would be much easier to operate with entire instances but not with the fragmented parts of instances. It is possible to unite these instances into groups, but the appropriate level of abstraction is required to prevent creation of tiny instances or groups.

3.3 The Ways of Data Presentation

The monitoring component should present obtained data in a readable and understandable way. To satisfy this requirement it is necessary to create an appropriate user interface. There are also several approaches that could be applied.

1. Generation of reports – the main idea lies in creation of a specific document that could contain all required information. The document could be a file of .doc or .pdf format. For most of users paper documents are more convenient when it comes to reading of a large text which contains a lot of important details. However, the real-time monitoring is impossible in this case.

2. An order based on importance priority – this approach could be applied to pay user’s attention on the most important information. The priority could be chosen by the developer or specified by the user. But it becomes difficult to find a small detail with low priority when the user needs it.

3. Separation by categories – such approach is perfect for the systems where data can be separated by some rule or principle. In this case users would easily find necessary information, because they should know to which category it belongs. This approach makes the user navigation convenient and understandable. The user interface which follows the main idea of separation by categories could be developed with the help of different tools and libraries which support tabs, lists, etc.

3.4 Conclusions

In this chapter possible ways of building the monitoring component were examined. Based on the described advantages and drawbacks of each approach it is possible to make a choice and apply it during the development process.

The monitoring component is considered to monitor information about a set of instances of the system. That is why all required data will be aggregated by the groups of instances. The data will be collected with the help of WMI, which is followed by a list of its advantages and is able to cope with the given tasks.

Obtained data could be presented according to each of the three approaches. The best way is to use them in combination. Special reports will satisfy users who prefer paper reading, while categories will allow to perform real time monitoring and show information related to each group or instance. The information will be prioritized, so the most important data will be observable and noticeable.
4. Platform Architecture

When designing a software component it is necessary to take into account the environmental conditions in which it will operate. Hardware parameters have a great importance as well as software parameters. Developed solution must be combined with software frameworks, which are used in the environment. At the same time the characteristics of the hardware must fulfill the requirements of software to create suitable conditions for the correct and efficient operation. This problem becomes particularly crucial when developing software components or parts of an existing system. In order to build-in a monitoring component in a proper and seamless way, it is extremely important to understand all the architectural features.

4.1 Platform Architecture without Monitoring Component

Figure 4.1 shows the overall architecture of the software platform. The platform represents a client-server application. The future monitoring component is considered to be one of the services, that is why the backend side should be analyzed in detail.

Microsoft SQL Server 2008 is used as the database management system. It is physically located on the same computer - the server. This arrangement allows to interact with the database in a faster and more efficient way, since there is no need to connect to the database server through the network that would be entailed by a loss of performance. Two basic services are the main functional elements.

![Figure 4.1: The overall existing architecture of the platform](image)
4.1.1 Anonymous Service
One of the main purposes of the service is user authorization. When a user logs in a session is created. A special hash identifier is assigned for each user session. The relevance of the session could be checked by the mechanism of hash identifier comparison. However, there is a number of user actions that can happen without the procedure of authorization. The list of such actions may include the selection of platform’s language which doesn’t depend on user rights or privileges. This service is responsible for handling such actions.

4.1.2 Authoring Service
This service provides a set of tools for editing and management of information existing in the system. It should be noted that the platform UpTime is based on the internal type system that is optimized specifically for each customer. There are some basic types that have all the necessary properties and lead to maximum reuse.

4.2 Platform Architecture Including the Monitoring Component
With the commissioning of the monitoring component the overall architecture (Figure 4.2) stays the same, though it undergoes some local changes.

![Diagram of platform architecture including the monitoring component](image)

Figure 4.2: The overall architecture of the platform including the monitoring component

The new component is represented as an additional service (Diagnostic service) that has its own behavior and interacts with other parts of the platform. New items are indicated by dotted lines. New arrows which represent interactions between architectural components are marked similarly. A unit marked with OS plays a special role. It stands for the operating system which is installed on the server. Operating
system itself is not a part of the architecture, but the presence of this block is necessary for understanding of all aspects and goals of the monitoring component. The new component interacts not only with existing services, but also with operating system. The important difference of monitoring and diagnostic component from the other services is the fact that it is accessible only for a certain type of users. It means that the platform remains client-server based, but only specific clients can use the new component.

4.2.1 Interaction with Authoring Service
The UpTime platform is organized so that each time a user action is performed a special log message is sent by Authoring service using the UDP protocol on the specified port. This message has a complex structure that allows you to fit the message with useful information. Each message contains information about the platform and processes that are performed during each individual user action. Every user action can be followed by several messages, depending on the degree of complexity and processes, which are necessary to complete the action successfully.

The structure of each log-message is the following:
- date of an action or process;
- time of an action or process;
- stream number,
- message type;
- program class or other entity that is responsible for the action or process;
- time during which the action was performed;
- text of database query.

All log - messages can be classified by their type:
1. Information messages.
3. Platform errors.
4. Reports of a fatal crash.

Information messages contain information about important events that happen within the platform. These events are caused by going through the user's interaction with the platform. An example of this report can be a message about creating of a new document or editing of an existing one. This message is tagged [INFO].

Messages of debugging process are designed to track the behavior of the platform from the perspective of the developer. Depending on user actions such communication may contain information about allocated objects, the program classes that are directly involved in the allocation of these objects, queries to the database, and so on. This information is very useful for administrators and developers of the platform, as it helps to see all operational aspects from the inside and step-by-step. The distinguishing of this type of messages becomes possible by using a special tag [DEBUG].

When an error occurs or an exception is formed, a message that contains all necessary information for analysis is sent. This type is marked by the tag [ERROR].

In case of fatal errors of the platform that affect its performance and cause you to restart it for further work, the port is notified by the message with the tag [FATAL].
This type of messages is the most useful in terms of analysis of errors that occur, and finding the probable causes of their occurrence.

The monitoring component has a special parser that is focused on partitioning messages into components according to the structure described above. This partitioning allows the user to provide information in a convenient form, provide opportunities for quick navigation, search tools, and sorting data (Figure 4.3). It is important that the mentioned functions are included into the monitor, so the parser is a part of the component. Such approach allows to perform log listening without the usage of special additional tools.

![Figure 4.3: Parsed log messages](image)

In order to make the monitoring component more flexible and reduce its dependence on the particular software platform, it was decided to give the user the opportunity to specify the port number that he/she wants to listen to and check the log - messages received.

The monitoring component can not only monitor the logs, but also draws the user's attention to the error messages that occurred. This allows the administrator to respond to such situations and may contribute to faster identify the causes of errors.

### 3.2.2 Interaction with Database Server

Effective monitoring involves the creation of regular snapshot of the current performance to isolate processes that cause the problems, and ongoing collection of data to track trends in growth or decline in performance. The structure of Microsoft SQL Server and Microsoft Windows operating system includes tools to monitor the current state of the database and measure performance if the state changes.

Monitoring SQL Server allows you to:

1. Identify opportunities to increase productivity. For example, controlling the responsiveness of commonly used queries, you can specify whether you want to change the query text or index table;
2. Evaluate user activity. For example, controlling the users who connect to the instance of SQL Server, you can determine if your security settings are set correctly and test your applications and systems development. By controlling the execution of SQL-queries, you can determine whether they are written, and test results that they return;
3. Eliminate any problems or components to build software applications, such as stored procedures.

One means of monitoring the performance parameters of the database server are system tables of SQL Server 2008. These tables contain a number of useful information that can be obtained through the usual SQL-query. All system tables are hierarchically organized. You can separate the elements of the first level of the hierarchy according to the following categories:
- Integration Services;
- Backup and Recovery;
- Replication;
- Notification Services;
- Agent of SQL Server;
- Database Planning Services.

<table>
<thead>
<tr>
<th>Data which is obtained</th>
<th>Table’s name</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of tables which are present in the database, the number of rows in each table and the amount of space that is involved.</td>
<td>sys.indexes</td>
</tr>
<tr>
<td></td>
<td>sys.tables</td>
</tr>
<tr>
<td></td>
<td>sys.allocation_units</td>
</tr>
<tr>
<td>List of all database files that are stored on the server, the physical path to these files and their size.</td>
<td>sys.master_files</td>
</tr>
<tr>
<td>Information about sessions on the database server specifying the program name, user login, session start time, the current state of the session, CPU and RAM consumed, and the number of logical readings and records inside the session.</td>
<td>sys.dm_exec_sessions</td>
</tr>
<tr>
<td>Information about database queries to be performed at this time, indicating the login of user, query start time, CPU time for query and SQL - text of the query.</td>
<td>sys.dm_exec_requests</td>
</tr>
<tr>
<td></td>
<td>sys.dm_exec_sql_text</td>
</tr>
<tr>
<td></td>
<td>sys.dm_exec_sessions</td>
</tr>
<tr>
<td>List of indexes for tables with the name of the index, index creation date.</td>
<td>sys.indexes</td>
</tr>
<tr>
<td></td>
<td>sys.tables</td>
</tr>
<tr>
<td>List of other performance counters values.</td>
<td>sys.dm_os_performance_counters</td>
</tr>
<tr>
<td>List of blockers of SQL Server and instances that are blocked.</td>
<td>master.dbo.sysprocesses</td>
</tr>
<tr>
<td></td>
<td>sys.dm_exec_sql_text</td>
</tr>
</tbody>
</table>

Table 4.1 Abilities of Database Server monitoring block
Data which is obtained | Table’s name
--- | ---
Information about last 500 database queries specifying SQL - query text, the number of performances since the beginning of the last execution of the request, the time during which the query was executed. | sys.dm_exec_query_stats
sys.dm_exec_sql_text
Information about the configuration and setup of the database server with the name of the configuration option and its value. | sys.configurations

Table 4.1 Abilities of Database Server monitoring block (continued)

### 4.2.3 Interaction with Operating System

To make a right conclusion about system performance monitoring of software parameters is not enough. It is also necessary to monitor hardware characteristics and performance parameters. Information about static parameters is as important as information about parameters which are being dynamically changed in time.

All system performance parameters were classified by the following categories:

1. Parameters of current processes.
2. Parameters of BIOS.
3. Parameters of CPU.
4. Parameters of disk drives.
5. Parameters of logical disks.
6. Parameters of operating system.
7. Parameters of network connections.

It is important to notice that the Diagnostic service does not interact with hardware devices directly. It uses facilities of the operating system to obtain all required information (Figure 4.4).

![Figure 4.4: Detailed view on Diagnostic service – Operating system interaction](image_url)

14
5. Monitoring Component Architecture

The monitoring component is based on client-server architecture. It could be explained by a strong focus of the monitoring component on the UpTime platform. Since it is the part of the platform, its architecture must fit the environment.

5.1 Backend Architecture

When designing big and complex system, the principal of decomposition is often used. It lies in dividing of the system into separate sub-systems. In this case the level of decomposition is determined by the categories of the monitoring component’s services. Such approach allows to develop separate parts in isolation from each other and uniting them afterwards.

All services provided by the server side could be divided into the following categories:

1. Data collection services – responsible for collecting values of performance parameters.
2. Data analysis services – responsible for analysis of performance parameters and determining of potential risks.
3. Log services – responsible for log-messages listening and parsing.
4. Data export services – responsible for creating reports and e-mail sending.

But the problem of high cohesion between elements occurs. When solving their own tasks some services have to use a couple of other services. As a result cohesion grows. Such situation makes the maintenance process rather hard. To solve this problem the design pattern called Facade was used. This pattern provides one unified interface instead of the set of interfaces for a system, so the usage of the system becomes simpler. Facade is a certain object that accumulates a set of high-level operations for interaction with a system [5]. As described on figure 5.1 the services are independent from each other and could be involved by client through the facade.

![Diagram of Backend Architecture](image-url)

Figure 5.1: Backend architecture
5.2 Frontend Architecture

The frontend is built according to MVVM pattern. This pattern can be related to the family of MVC patterns. But it has its own significant differences, which form its originality.

MVVM pattern consists of three elements:
1. Model – stands for a fundamental data which is necessary for the work of the program (classes, data structures, etc.).
2. View – stands for a graphical user interface (forms, buttons, other control elements).
3. ViewModel – stands for an abstraction of View from one side and plays a role of Models’s data wrapper at the same time. It contains a Model that is transformed into a View. Also ViewModel contains commands which are used by View to affect Model [6].

The main advantage of MVVM over MVC is the mechanism of data-binding. In MVC the changes of the UI do not affect the Model directly. They are preliminary processed by Controller. Such technologies like WPF and Servelight contain the conception of data binding. That is why with the usage of these technologies applying of MVC pattern becomes inconvenient since data binding is not embedded into the conception of it.

The basic Idea of MVVM is to separate View (aka UI) from the logic that can occur in the form and to organize data binding between the layers. Such approach allows changing logic and interface separately. For example, the developer organizes the logic of work with data while designer works on the user interface.

MVVM pattern has also some disadvantages which are related to its applying in small systems with simple UI. Sometimes it can be overkill. It also could be hard to design ViewModel for big and complex systems. But if this pattern is used in a right way it gives a list of advantages.

The implementation of MVVM pattern could be seen on Figure 5.2. There is a set of ViewModels, a View and a Model. Each ViewModel is responsible for dynamic updates of an appropriate part of user interface (View in terms of pattern terminology).
The core idea of the monitoring component’s work becomes clear when you look at the backend and frontend architecture at the same time. User with the help of UI sends a request for obtaining required data to the server (to the particular service to be precise) which responses and provides the data. Services are not used directly but through the facade. Obtained data from the server is stored in Model on a client side. With the help of ViewModel’s data binding mechanisms obtained data is displayed to user via UI.

Figure 5.3 shows the interaction between server side and client side in terms of two described design patterns.
The monitoring component is designed with applying of 2 patterns which make possible to separate the development of frontend from backend development, but do not cause the problems in establishment of interaction between two sides.
6. Detailed Internal View on the Monitoring Component

In this chapter the detailed description of the monitoring component is given. The most important aspects starting from use cases and class diagrams, continuing with user interface description and argumentation, and finishing with the description of involved frameworks and tools, are described.

6.1 Typical Use Cases of the Monitoring Component

To show the functional abilities of the monitoring component a diagram for the most typical use cases (Figure 6.1) was created. The detailed description of each use case is provided in the table 6.1. The monitoring component is designed for one and only type of user: administrator of the platform. No more user roles are provided. Therefore the diagram shows only one actor. The administrator does not have any restrictions regarding information and can use the full range of the provided functionality.

Figure 6.1: The most typical use cases

<table>
<thead>
<tr>
<th>Use case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Info Obtainment</td>
<td>Detailed monitoring of system performance parameters. Also provides user with hardware parameters.</td>
</tr>
<tr>
<td>App Parameters Obtainment</td>
<td>Detailed monitoring of the resources that are consumed by the platform and are necessary for its work.</td>
</tr>
</tbody>
</table>

Table 6.1: Description of the use cases
<table>
<thead>
<tr>
<th>Use case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Parameters Obtainment</td>
<td>Monitoring of some useful performance parameters of database server.</td>
</tr>
<tr>
<td>Journal Errors/Warnings Obtainment</td>
<td>Monitoring of the entries in journals of the operating system. Entries marked as errors or warnings are the main thing to monitor.</td>
</tr>
<tr>
<td>Report Creation</td>
<td>Creation of documents in .pdf format which contain information about current performance parameters of the platform. Such feature is useful since it becomes possible to print or save the report for future analysis.</td>
</tr>
<tr>
<td>E-mail Sending</td>
<td>Sending of the created reports by e-mail to a specified address.</td>
</tr>
<tr>
<td>BIOS Info Obtainment</td>
<td>Obtainment of the detailed information about BIOS.</td>
</tr>
<tr>
<td>Settings Management</td>
<td>Creation or editing of the monitoring component’s settings which contain thresholds for the allowable values of performance parameters, address of the port that receives log messages, connection string to a database, e-mail address for reports. It is also possible to restore the default settings.</td>
</tr>
<tr>
<td>CPU Info Obtainment</td>
<td>Obtainment of the detailed information about CPU characteristics and its current load percentage.</td>
</tr>
<tr>
<td>OS Info Obtainment</td>
<td>Obtainment of the detailed information about operating system and its settings.</td>
</tr>
<tr>
<td>Logical Disks Info Obtainment</td>
<td>Obtainment of the detailed information about logical disks.</td>
</tr>
<tr>
<td>Disk Drives Info Obtainment</td>
<td>Obtainment of the detailed information about physical disk drives.</td>
</tr>
<tr>
<td>Network Info Obtainment</td>
<td>Obtainment of the detailed information about current network connections with their utilization percentage.</td>
</tr>
<tr>
<td>Parameters Analysis</td>
<td>Obtainment of the list of parameters which are higher or lower than set thresholds. Draws the user’s attention on errors and warnings in the system journals. Makes the selection of log messages which indicate errors of the platform.</td>
</tr>
<tr>
<td>App Logs Obtainment</td>
<td>Monitoring of the log messages which are arriving during the platform’s work. Provides user with sorting, filtering and searching tools.</td>
</tr>
</tbody>
</table>

Table 6.1: Description of the use cases (continued)
Some use cases can be interpreted as the basic functions of the monitoring component. The monitoring component cannot work properly without these functions. At the same time there is a list of additional abilities that extends the basic functionality.

6.2 Static Structure of the Monitoring Component
A static structure of the monitoring component could be presented with class diagrams. Class diagram defines the types of objects, different static relations and correlation between them. It contains a set of declarative elements like classes, types which are combined into a graph.

Each class contains a set of methods with specified access modifiers and a set of properties with specified types.

![Class diagram for a general backend structure](image)

The implementation of chosen design pattern can be seen on the figure 6.2: DiagnosticService plays the role of facade. Other blocks encapsulate the set of provided services.

The block of information accumulation (Figure 6.2 - InfoCollector) and analysis (Figure 6.2 - Analyzer) use predefined settings. These settings consist of database connection string, log messages port number, e-mail address to send reports and a set of thresholds for parameters being monitored.

All thresholds can be divided into 2 categories:
1. Important thresholds.

2. Critical thresholds.

The first type is intended for the situations when the current values of performance parameters are outside of the scope of permissible, but at the same time they do not affect the loss of performance. In such cases a warning message is created for user.

The second type is intended for the situations when the current values of performance parameters are outside of the scope of permissible and affect the loss of performance at the same time. In such cases an error message is created for user.

---

Figure 6.3: Settings class and its usage

To obtain system performance parameters and hardware characteristics class InfoCollector uses a set of other classes. Each of these classes is responsible for a separate category. Parameters of each category are presented as properties of the class. The classes are named in a way that it becomes possible to understand which category it is responsible for.
6.3 UI Specification

It is really important to design a system, to develop it and to make it work. But it is not enough to obtain the data. It is also necessary to design an appropriate UI to present the data in a human readable and understandable way.

It was decided to present accumulated data as a set of tabs. With such approach a division of data by different categories became possible. Figure 6.5 shows the tabs. The categories of system, database performance parameters and logs can be noticed. Also it was decided to create a home tab and a tab of settings.

The Home tab contains the most important and actual information for user. The consumption of main resources is shown. Also this tab aggregates two lists of errors and warnings which are aimed to draw user’s attention on potential problems and prioritized by their importance.
Generated reports are opened in a separate window for more convenience. The report includes the date of its generation and a set of data divided by predefined categories (Figure 6.6).
The System tab as well as the Database tab contains a set of sub-categories. The chosen category and tab are highlighted to make UI user friendly and easy to use. The information is commonly provided with lists where the name of parameter and its value are given (Figure 6.7).

Figure 6.7: System tab specification

But sometimes the data can be really huge and it takes a lot of time to find required information in a list. That is why the table approach was applied (Figure 6.8). A huge data which is organized into tables is much more understandable for users. It is also becomes possible to navigate the information with the tools of sorting and filtering and this is a benefit of such approach.

Figure 6.8: Table approach
The Settings tab also contains sub-categories. Since there is a rather big set of settings it can be difficult to manage them. With sub-categories of settings a user can easily find and set the value he needs.

![Settings tab specification](image)

Figure 6.9: Settings tab specification

### 6.4 Involved Development Tools
According to the design, existing hardware and software of the platform the development tools and technologies which make possible to integrate a monitoring component easily were chosen.

#### 6.4.1 Development Environment
The development of the monitoring component was done using Microsoft Visual Studio 2010 Ultimate Edition. This choice is explained by the fact that the platform is based on Microsoft .NET Framework.

#### 6.4.2 Programming Languages
The backend source code of the monitoring component was written using C#. This language is object-oriented and is aimed for development of programs which are based on Microsoft .NET Framework.

HTML-5 and JavaScript were used to develop the frontend part. With the help of HTML-5, the web UI was developed while JavaScript made the page more dynamic and interactive.

#### 6.4.3 Frameworks and Technologies
To organize interaction between a server and a client AJAX [7] was used. AJAX represents an approach of building of interactive user web interfaces. The main idea of this approach lies in background exchange of data between a server and a client. As a result, when the data is updated the page is not reloaded, so the program becomes faster and more convenient.
The usage of fast and multifunctional JavaScript framework, called JQuery, made it possible to manipulate HTML-elements, handle different events and create animation when a huge portion of data is being loaded. The main benefit of this framework lies in its cross-browser functionality. Universalism of JQuery caused the JavaScript code to be easier for understanding.

To implement navigation mechanisms for the data which is organized into tables a JavaScript framework called DataTables was used.

Some elements of the UI are too complicated to implement. To make the implementation of such elements (e.g. progress bars, tabs), the JQuery UI framework was used [8]. This framework contains a set of implemented effects, widgets and themes which are based on JavaScript.

Since the frontend part is based on MVVM pattern the appropriate interaction between its parts is needed. That is why another JavaScript based library was used. This library is called KnockoutJS [9] and is aimed to make the implementation of MVVM pattern easier.

A huge portion of the functional abilities of the monitoring component is built on WMI. WMI is one of the basic technologies for centralized management and tracking the work of computer infrastructure under Windows environment. Since WMI is built in object-oriented way, all data about operating system are represented as objects with properties and methods. All classes are grouped into namespaces which are hierarchically organized and logically connected. Classes have properties which are hierarchically depending on each other. Properties are used to determine the instance of a class and to describe the state of the consumed resource. In most cases all properties are read only, but some of them can be modified by a certain method.

6.5 Data Analysis and Priority of Results
As it was mentioned above the monitoring component is considered not only to obtain data about performance but also to analyze it. The goal of analysis is to define potential risks or errors that has occurred, and draw the user’s attention to them. There are several situations a user should be notified about:

- Critical threshold for a parameter is exceeded.
- Important threshold for a parameter is exceeded.
- Log message marked with tag [ERROR] or [FATAL].
- Operating system journal entry of type Error or Warning is found.
- Some blockers on database server are found.

The importance of these cases is not the same. That is why they were prioritized depending on the potential risk that could cause problems:

1. The highest risk can happen when the critical threshold for free disk space is exceeded. A program can work slowly with high values of CPU load or network utilization, but it cannot work with the lack of free disk space. Also it is extremely important to draw user’s attention when thresholds of memory usage by application or system are exceeded. In such cases the admin can monitor all
processes, find the heaviest and make some decision. Critical thresholds for other parameters like CPU load or network utilization are less important.

2. Errors of operating system are the next entity to look at. They could be a reason for unexpected behavior of the platform, so it is necessary to show all of them.

3. Sometimes the reason of crashes of the platform lies inside the platform. Monitoring of log messages helps to see the processes and actions inside the platform step by step. If something goes wrong the appropriate message will be obtained and should be displayed.

4. Since all data is stored in database it is necessary to monitor database server. Sometimes situations when one process blocks another can happen. Such situations are important to monitor and to notify about.

5. The less priority is given to situations when important thresholds are exceeded. Such situations are not extremely risky. They are considered to be warnings to pay attention on.
7. Evaluating the Monitoring Component

To see the performance of the monitoring component, time which is required to get different kinds of data from the server by requests was measured.

To conduct an experiment the list of requests, which the monitoring component sends to retrieve data, was made. All requests were divided into groups. Each group of requests is responsible for obtaining data for appropriate tab.

Execution time for each request was measured. The given execution time in the Table 7.1 is an average value of 10 measurements for each kind of request.

<table>
<thead>
<tr>
<th>Request</th>
<th>Execution Time, sec</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Tab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetHomeTabInfo</td>
<td>1.68</td>
<td>Gets current parameters to present them on main tab, gets errors and warnings which are obtained after analysis.</td>
</tr>
<tr>
<td>GetLogs</td>
<td>0.007</td>
<td>Gets logs which arrive from UpTime Application to specified UDP port.</td>
</tr>
<tr>
<td>CreatePDF</td>
<td>0.95</td>
<td>Generates PDF document with current parameters and errors.</td>
</tr>
<tr>
<td>Settings Tab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetSettings</td>
<td>0.016</td>
<td>Reads settings (thresholds, connections strings, etc.)</td>
</tr>
<tr>
<td>RestoreSettings</td>
<td>0.079</td>
<td>Reads default settings and restores current values</td>
</tr>
<tr>
<td>SaveSettings</td>
<td>0.019</td>
<td>Saves user settings separately from default settings</td>
</tr>
<tr>
<td>Database Tab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetDBTablesInfo</td>
<td>0.048</td>
<td>Gets information about all tables existing in specified database</td>
</tr>
<tr>
<td>GetDBFileSizeInfo</td>
<td>0.046</td>
<td>Gets information about .mdf files on SQL Server</td>
</tr>
<tr>
<td>GetDBSessionsInfo</td>
<td>0.025</td>
<td>Gets detailed information about sessions on SQL Server</td>
</tr>
<tr>
<td>GetRunningQueries</td>
<td>0.016</td>
<td>Gets detailed information about queries which are running at the moment</td>
</tr>
<tr>
<td>GetRecentQueries</td>
<td>0.58</td>
<td>Gets detailed information about queries which were running recently</td>
</tr>
<tr>
<td>GetIndexesInfo</td>
<td>0.063</td>
<td>Gets detailed information about indexes in tables</td>
</tr>
<tr>
<td>GetBlockersInfo</td>
<td>0.025</td>
<td>Gets detailed information about blockers and blocked processes if any</td>
</tr>
<tr>
<td>GetDBCountersInfo</td>
<td>0.02</td>
<td>Gets detailed information about database performance counters</td>
</tr>
<tr>
<td>GetServerConfig</td>
<td>0.016</td>
<td>Gets detailed information about SQL Server configuration</td>
</tr>
</tbody>
</table>

Table 7.1: Requests execution time
<table>
<thead>
<tr>
<th>Request</th>
<th>Execution Time, sec</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Tab: GetAllProcesses</td>
<td>0.02</td>
<td>Gets detailed information about all processes which are running at the moment</td>
</tr>
<tr>
<td>GetBIOSInfo</td>
<td>0.028</td>
<td>Gets detailed information about BIOS</td>
</tr>
<tr>
<td>GetCPUInfo</td>
<td>1.16</td>
<td>Gets detailed information about CPU</td>
</tr>
<tr>
<td>GetCPULoad</td>
<td>1.09</td>
<td>Gets the value of current CPU load percentage</td>
</tr>
<tr>
<td>GetLogicalDiskInfo</td>
<td>0.09</td>
<td>Gets detailed information about all logical disks</td>
</tr>
<tr>
<td>GetDiskDriveInfo</td>
<td>0.489</td>
<td>Gets detailed information about all disk drives</td>
</tr>
<tr>
<td>GetOSInfo</td>
<td>0.057</td>
<td>Gets detailed information about Operating System</td>
</tr>
<tr>
<td>GetNetworksByType</td>
<td>0.805</td>
<td>Gets a list of network connection according to user's choice</td>
</tr>
<tr>
<td>GetNetworksByName</td>
<td>0.493</td>
<td>Gets detailed information about specified network connection</td>
</tr>
<tr>
<td>GetSystemEvents</td>
<td>4.6</td>
<td>Gets warnings and errors from Windows System Event Manager</td>
</tr>
</tbody>
</table>

Table 7.1: Requests execution time (continued)

The obtained values differ depending on the complexity of a request, on size of processed and transferred data (Table 7.1). Also, it is important to notice that the current workload of the system makes influence on the request execution performance. In most cases the speed of request execution could be considered as good [10]. The values of execution time for different requests are fractions of one second. It means that it is almost impossible for users to notice the difference in performance in this case. Usually such high performance happens when complex calculations are not required.

However, some kinds of requests take more than one second to execute. GetHomeTabInfo request could be taken as an example. It aggregates and processes a huge amount of data. The potential reason of such behavior could also in a slow work of used API functions. Anyway, the resources of the system which are necessary to execute requests are not extremely big, so the monitoring component does not make bad influence on the overall performance of the platform.

On the example of GetSystemEvents request (Figure 7.1) it is possible to see that the values are uniformly distributed [11]. Each blue column represents one measurement and contains the value of execution time of the request. The figure illustrates the distribution of the result values. Such high values in comparison with other requests could be explained by the fact that this type of request processes thousands of Windows System Event Manager’s entries. The operating system itself requires really much time to process and present this data. It is obvious that the program cannot work faster than operating system. That is why this particular request is a weak part of the monitoring component. Unfortunately, nothing could be done to solve this problem with the help of
programming language capabilities because of described reasons. It was decided to perform this request only by specific user action and notify him/her about the progress of execution with the help of a progress bar.

![Figure 7.1: GetSystemEvents request time execution measurement](image)

The data provided by the monitoring component is full and correct. The data is obtained as a copy of information which exists in the system – no complex conversions are performed. To verify the correctness of this data, it was obtained manually and checked.
8. Conclusions

The work on master thesis project consisted of a sequence of stages. Beginning with problem analysis, continuing with different kinds of research on suitable solutions, designing and development of the program and finishing with evaluating the result each stage has been passed.

8.1 Achieved Results
Several results of master thesis work can be emphasized.

8.1.1 A Working Monitoring Component
A holistic program component is developed. It can be easily integrated into the platform and become a part of it. It is already possible to perform real-time monitoring of the platform and running processes, obtain performance parameters, resources consumption information, determine potential risks, problems and bottlenecks with the help of this monitoring component.

8.1.2 High Performance of the Monitoring Component
The design patterns and development tools which were chosen allowed me to achieve rather satisfactory results in monitoring component’s performance. Depending on the complexity of a task the time of its execution varies but stays on the permissible level.

8.1.3 Decrease of Laboriousness
It was rather difficult to obtain required parameters and other useful data to analyze the performance of the system before. It could require a deep knowledge from the user and could take a lot of time to make all needed actions to get the final information. By using the health monitoring component the process of obtainment information about performance parameters becomes faster and easier, so the laboriousness of this work is reduced. This benefit could be considered as an additional result of thesis work.

8.2 Future Work
The monitoring component is holistic and is finished in terms of thesis project. But there is no limit to perfection, that is why some future work on it could be considered.

As it was mentioned, the platform is organized in a way that it includes a list of inner types which are helpful in development and increase reusability of the components. Information about allocated objects of these inner types could be really useful to see what exactly happens inside and what the behavior of the platform is. One of the possible tasks for future work is to handle inner objects allocation monitoring.

There is a list of settings which are provided by the monitoring component. This list is enough for platform monitoring for now, but it could be extended. As a result platform monitoring would become more precise and detailed.

A list of available features could also be extended. Some graphical representation of information could be useful. Building of graphics on the basis of accumulated and stored statistical data about a certain performance parameter is one of the possible features to add.
One of possible ways to improve the monitoring component is to create some metrics to test its flexibility and modularity. Some changes based on the obtained results could be possibly required to increase the overall quality of the component.

The monitoring component will be integrated into UpTime platform. It will be used as a part of the fourth version release.
References


