004.415.2.045 (076.5)

O.P. Dyshlevyy, M.M. Kostiv
National Aviation University

TOOL FOR CALCULATION OF METRIC'S ABNORMAL VALUES OF SOFTWARE

The article is dedicated to creation the tool of calculation of metrics abnormal values. The analysis of statistical techniques has been made for measurement of abnormal metrics, and technique of calculation of metrics abnormal values of software has been proposed. Main software metrics have been described for determination of program maintenance complexity. Tool of calculation of metrics abnormal values has been developed, and it`s functionality has been demonstrated on example of several projects.

Стаття присвячена створенню засобу розрахунку аномальних значень метрик. Проводився аналіз статистичних методик для вимірювання аномальних значень метрик, та була запропонована методика розрахунку аномальних значень метрик програмного забезпечення. Описані основні метрики програмного забезпечення для визначення складності підтримки програми. Розроблявся засіб для розрахунку аномальних значень метрик, та була продемонстрована його функціональність на прикладі декількох проектів.

Статья посвящена созданию средства расчета аномальных значений метрик. Проводился анализ статистических методик для измерения аномальных значений метрик, и была предложена методика расчета аномальных значений метрик программного обеспечения. Описаны основные метрики программного обеспечения для определения сложности поддержки программы. Разрабатывалось средство для расчета аномальных значений метрик, и была продемонстрирована его функциональность на примере нескольких проектов.

Key words: Abnormal value, Measurement, Goal-Question-Metric paradigm, Arithmetical mean, Standard deviation, Metric

Introduction

There are many examples of project that have overrun their budgets and schedules [1, 2]. Software engineers have addressed these problems by continually looking for new techniques and tools to improve development and maintenance process and product. One of them is software measurement [1, 3].

Measurement is important for three basic activities [4].

First, there are measures that help to understand what is happened during development and maintenance. Engineers assess the current situation, establishing baselines that help to set goals for future behavior. In this sense, measurements make aspects of process and product more visible, giving better understanding of relationships among activities and the entities they affect.

Second, the measurements allow engineers to control what is happening on projects. Using baselines, goals and understanding of relationships, engineers predict what is likely to happen and make changes to processes and products that help to meet their goals. For example, it is possible to monitor the complexity of code modules, giving through review only to those that exceed acceptable bounds.

Third, measurement encourages engineers to improve processes and products.

With using any metric it is necessary to know what is too high or too low, too much or too little. In other words, some reference points are required, some means to link a particular metric value to useful semantics. Thresholds are based in statistical information [5].

Once the metric's values are existed, it is necessary to judge whether the value indicates critical situation or not. Thresholds can help to judgment. When design metrics exceeds a certain threshold, the design element can then be consider "critical" and must be redesigned.

Thresholds are defined as in heuristic values used to set ranges of desirable or undesirable metric values for measured software. These thresholds are used to identify abnormal values, which may be or not be an actual problem [6].

Abnormal values can help to detect bad smells from metrics, support refactoring and regenerate code. Relations between metrics and bed smells were defined on the basis of thresholds. Thresholds can be customized either by a product manager or by special application.

The main purpose of appearing of abnormal values is present classes for which refactoring is

necessary. But it is very important to correlate size of class and the metric's value for class.

Last research review

There are many approaches to use software metrics [3, 5, 7, 8].

There is situation when many metrics still is not used in practice. It was suggested there are several reasons:

- Companies must change lifecycle, development processes to making measurement of a part of their activities;
- Software engineers don't know software metrics;
- Many metrics are not described very well.
- It is hard to explain measurement results by software engineers.

Some software researches try to change current situation. They describe some features of software metrics, principles of using them and explaining results. And they use statistics rules for this purpose [5, 7-12]. All of them didn't consider abnormal values of software metrics as present classes for which refactoring is necessary.

This paper describe technique and tool for funding abnormal values, several case studies which explain how to use results of this work in practice.

Developing of technique for finding of abnormal values

The most pragmatic issue is how to use metrics values so that they provide real information and not just a numbers. In this context, GQM model define the necessity obligations for settings objectives before embarking on any software measurement activity [13]:

1) List the major goal for which metrics are going to be employed.

2) From each goal derive the questions that must be answered to determine if the goal is met.

3) Decide what metrics must be collected to answer the questions.

Having this amount of data simple statistical techniques were employed in order to determine for each of these metrics:

- the typical values, i.e., the range of values that includes the data from most projects;
- the lower and respectively the higher margins of the typical interval;
- the extreme high values, i.e., a value beyond which a value can be consider as outlier.

Two statistical means were used to find what the typical high and low values are:

1) AVG, to determine the most typical value of the data set (i.e., the central tendency);

2) STDEV, to get a measure of how many the values in the data set are spread.

Knowing the AVG and STVED values and assuming a normal distribution for the collected data (i.e., that most values are concentrated in the middle rather than the margins of the typical values interval for a metric and the threshold for very high values. These are:

- Lower margin: AVG - STDEV;
- Higher margin: AVG + STDEV.

These margins tell the researcher the meaning of low, high for a given metric [5].

The steps of technique were shown in the flowchart diagram (Fig 1).

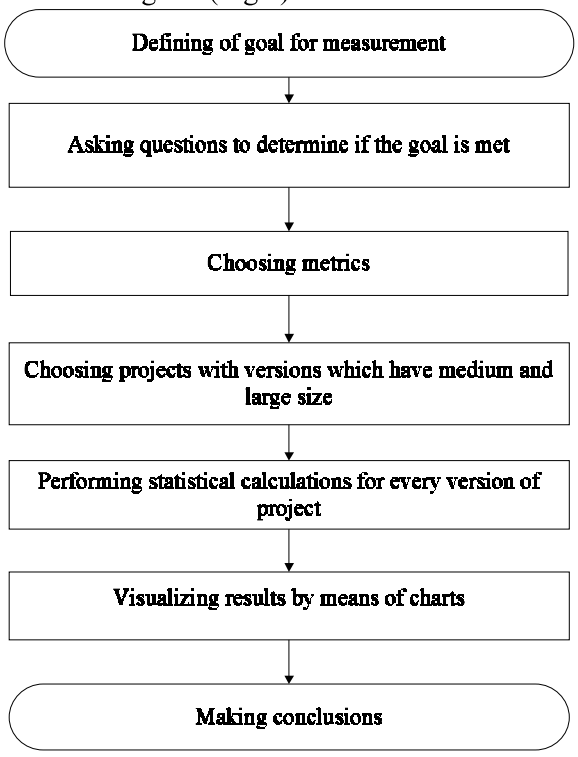


Fig. 1. The steps of technique for finding of abnormal values

The technique can be used both during software development and maintenance by developers, product managers and analytics. It helps to detect code smells, such as, for example, large classes, big conditional complexity. It is important to have several product versions. More product versions are, better statistical results will be.

Statistical calculations

The main purpose of the work is finding of abnormal values. That's why it is necessary to calculate the values of confidence interval using statistical formulas. The calculation can be divided in several steps:

- finding of average mean for generalizing;

- determining the value standard deviation using the value of average mean;
- calculating of limits for confidence interval by means of using the value of standard deviation.

In statistics, the average mean gives a very good idea about the central tendency of the data being collected [14-15]. The main significance of average mean is generalizing function: replacement of various individual values by average mean, which characterizes all range of values. Average mean is calculated by means of the next formula (1):

$$\bar{x} = \sum_{i=1}^{r} x_i f_i \qquad (1)$$

Standard deviation gives a measure of how the data are distributed according to average mean (2). x_i-x is a distance from a given number to a mean. Standard deviation is used for determining of limits for confidence interval [14-15].

$$\sigma = \sqrt{\sum_{i=1}^{r} (x_i - \bar{x})^2 f_i} \qquad (2)$$

For calculating the values of limits for confidence interval the three-sigma rule can be used. The value must be lain in the interval $[\bar{x} - 3\sigma; \bar{x} + 3\sigma]$. That's why $a = \bar{x} - \sigma$ and $b = \bar{x} - \sigma$, where a and b upper and lower limit relatively. By means of these formulas it is possible to develop program for finding of abnormal values. If the value of metric is less than a and greater than b and the frequency of metric is very low it means that the value is abnormal.

Tool development and description

Application components are displayed on figure 2. Data is displayed for user into open-flash-chart.swf component. It depends on from open_flash_chart_object.php that connects to the open-flash-chart.php that contains functions for charts calculations.

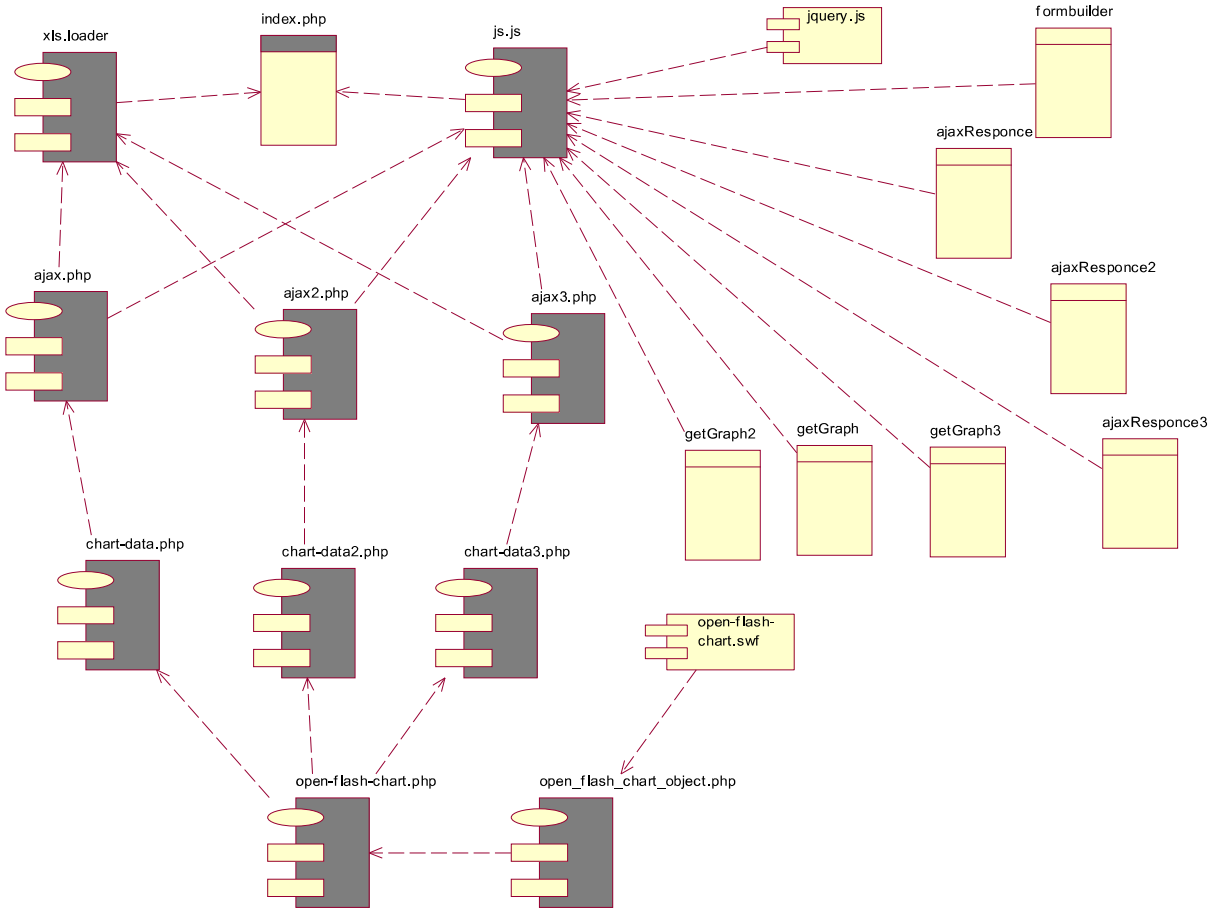


Fig.2 Component diagram of application

This need data that it can get from chart-data files (chart-data.php, chart-data2.php, chart-data3.php). *Chart data* files relates from ajax files (ajax.php, ajax2.php, ajax3.php). *Ajax files* depend on xls_loader.php or js.js file in conjunction of type of getting data. File js.js contain complex functions that work with DOM, create form and send ajax request to the ajax files.

Developed application calculates upper and lower limits of possible values of metrics and other statistical values. By means of application it is possible to find abnormal values of different values of metrics. By means of program software engineer can find class with abnormal values of metrics and then analyze and define why this values appear.

For using application it is necessary to gather data by means of iPlasma or other measurement tool and enter data to application: import data from Excel or use special input fields. Then it is necessary to analyze charts and make conclusions.

Case study

Selecting metrics. The goal is to analyze the complexity of classes for the purpose of improving, controlling maintenance. The metrics were chosen according this goal.

It is necessary to find the classes in application with very high complexity according to average mean of complexity for all classes. A very high complexity complicates the maintenance.

Complex classes need more time to develop and test. Therefore, excessive complexity should be avoided. Too complex should be simplified by rewriting or splitting.

WMC, CYCLO, NOM, DIT, LOC are metrics which were chosen in respect to goal of measurement.

Selecting products for research. Main principles of selecting products are size (medium or large) and many versions.

The selected products are: jfreechart, jhotdraw, jedit, struts, jcoverage, jfreemind.

Sizes of different versions of jedit product are in table 1.

Table 1
Versions of jedit

Version	Number of classes
jedit-4.3pre4	453
jedit-4.3pre7	518
jedit-4.3pre11	550
jedit-4.3pre14	546
jedit_4.4pre2	441
jedit4.5pre1	536

There are other metrics for versions of jedit in table 2.

Table 2
JEdit results

	WMC	DIT	LOC	NOM
jedit-4.3pre4	22,12	0,63	55	6,57
jedit-4.3pre7	21,35	1,27	143,5	7,43
jedit-4.3pre11	20,93	1,27	143,5	7,43
jedit-4.3pre14	19,83	1,24	142,5	7,07
jedit_4.4pre2	16,48	1,3	140	6,53
jedit-4.5pre1	6,54	1,22	143,4	7,12

The charts for WMC , DIT , LOC and NOM were built. Consider the chart of WMC metric (Fig 3) and chart of LOC metric(Fig 4). These charts represent results for two metrics of jedit. Other products and their metrics have the same tendency.

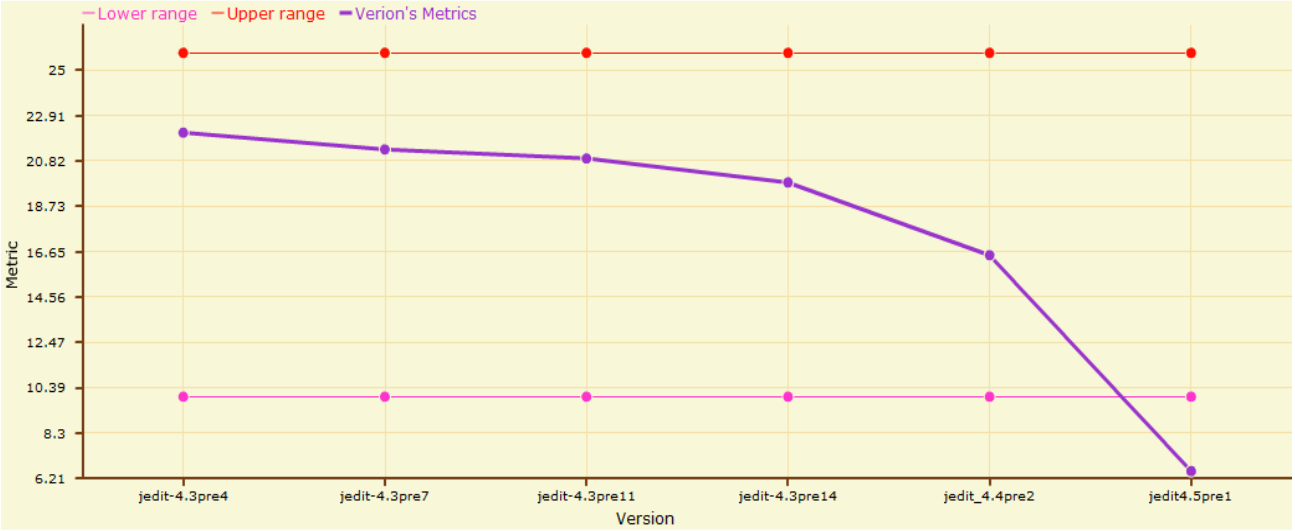


Fig.3 The chart with upper and lower limits of WMC metrics

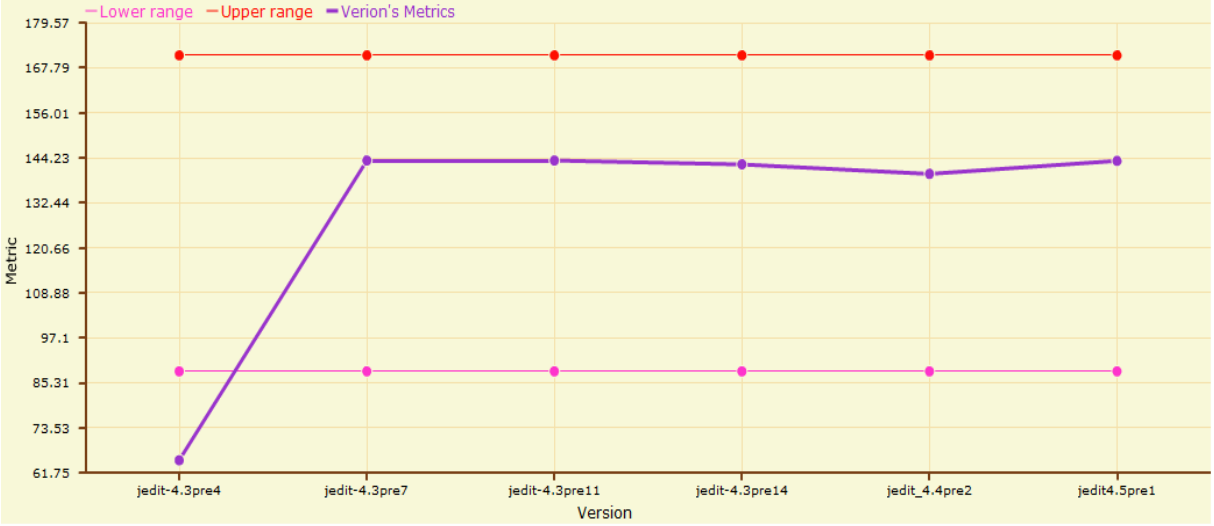


Fig.4 The chart with upper and lower limits of LOC metrics

In the jedit-4.5pre1 the outlier appeared (fig. 4). It can be caused by decreasing of quantity of total number of paths in the classes. Fig. 3 and 4 show LOC changes without outliers and doesn't influence on WMC. The complexity of some classes changed. The value WMC for class jEdit in jedit-4.3pre1 is 17 and in the last version is 10. The LOC for this class in jedit-4.3pre1 is 73 and in the last version is 70. The number of the total number of possible program paths decreased and application become less complex and less efforts and costs are necessary for maintenance.

Conclusions

Outliers can appear due to small conditional complexity of the last version of and mean the reducing of efforts and cost for maintenance and complexity. Users can see tendency of metrics changing in different application versions.

Sometimes abnormal values can be obtained by small functionality of application in the first versions in comparison to later versions. In most cases abnormal values can show the presence of code smell and indicate the necessary of refactoring when metric value go out the upper or lower range line.

References

1. *Соммервил Иан*, Инженерия программного обеспечения, 6-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 624 с. : ил. – Парал. тит. англ.

2. *Демарко Т., Листер Т.* Человеческий фактор: успешные проекты и команды, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2005. – 256 с., ил.

3. *Christof Ebert, Reiner Dumke, Manfred Bundschuh, Andreas Schmietendorf.* Best Practices in Software Measurement: How to use metrics to improve project and process performance. – Springer-Verlag Berlin Heidelberg 2005.-295p.

4. *Rakesh.L, Dr.Manoranjan Kumar Singh, Dr.Gunaseelan Devaraj,* (IJCSIS) International Journal of Computer Science and Information Security,Vol. 8, No. 2, 2010, Software Metrics: Some degree of Software Measurement and Analysis, 7 pages.

5. *Michele Lanza, Radu Marinescu and S. Ducasse* . Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer-Verlag Berlin Heilderbeg, 2006. – 213 pages.

6. *Knowledge-based software engineering,* Proceedings of the Seventh Joint Conference on Knowledge-based Software Engineering – 2006, 340 pages.

7. *Norman E. Fenton, Shari Lawrence Pfleeger* Software Metrics: A Rigorous and Practical Approach.-Cambridge University Press,1996.-638p.

8. *Linda M. Laird, M. Carol Brennan* Software Measurement and Estimation: a practical approach. John Wiley & Sons, Inc., Hoboken, New Jersey 2006.-257 p.

9. *Forrest Shull, Janice Singer, Dag I.K. Sjoberg* Guide to Advanced Empirical Software Engineering. – Springer-Verlag London Limited 2008.-394p.

10. *John C. Munson*. Software Engineering Measurement. – Auerbach publications 2003. – 564 p.

11. *Дишлевий О.П.* Предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення / О.П. Дишлевий // Вісник НАУ. – 2009. – №3. – С. 206–212.

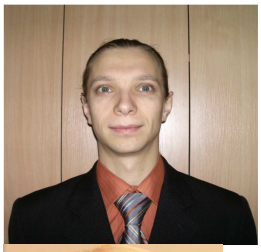
12. *Дишлевий О.П.* Підбір метрик для властивостей програмного забезпечення / О.П. Дишлевий // ПРОБЛЕМИ ПРОГРАМУВАННЯ. Науковий журнал. – 2010. – №2-3. – С. 237–242.

13. *Basili V.R., Weiss D.M.* A method for collection valid software engineering data // IEEE Transaction on Software Engineering, 10(6), pp. 728-38, 1984.

14. *Вентцель Е.С.* Теория вероятностей: Учеб. для вузов. – 7-е изд. стер. – М.: Высш. шк., 2001. – 575 с.: ил.

15. *Бабак В.П., Білецький А.Я., Приставка О.П., Приставка П.О.* Статистична обробка даних/ Монографія. – Київ: «МІВВЦ», 2001. – 388 с.

Відомості про авторів:



Дишлевий Олексій Петрович

старший викладач кафедри інженерії програмного забезпечення Факультету комп'ютерних наук, Національний авіаційний університет.

E-mail: oleksiy.dyshlevyy@livenau.net

Костів Мілана Миколаївна

студентка (магістр), 5 курс Факультету комп'ютерних наук, Національний Авіаційний Університет. Науковий напрям – емпірична інженерія програмного забезпечення, Національний авіаційний університет.

E-mail: milanab1@yahoo.com

Т.М.Заболотня, В.О.Каковський
Національний технічний університет України «Київський політехнічний інститут»

ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ КОНТРОЛЮ ДОСЯГНЕННЯ ЦІЛЕЙ

Дана стаття присвячена питанню аналізу сучасного програмного забезпечення для підтримки контролю досягнення цілей. Визначено ключові вимоги, яким повинні відповідати дані системи. Проведено вивчення існуючих програмних засобів, що вирішують завдання обліку та приоритизації цілей та кроків щодо їх досягнення, наведено порівняльну таблицю. Виявлено спільні позитивні та негативні риси розглянутих програмних засобів. Обрано систему, яку доцільно використати в ролі прототипу для подальшого вдосконалення.

Данная статья посвящена вопросу анализа современного программного обеспечения, предназначенного для поддержки контроля достижения целей. Проведено изучение существующих программных средств, решающих задачи учета и приоритизации целей и шагов по их достижению, приведена сравнительная таблица. Выявлены общие положительные и отрицательные черты рассмотренных программных средств. Выбрана система, которую целесообразно использовать в качестве прототипа для дальнейшего усовершенствования.

This article is devoted to the analysis of the software designed for the goal achievement support. The key requirements to be met by these systems are described. The research of existing software tools that solve the problem of goals and steps to achieve them accounting and prioritization is conducted by authors. The results are given in the form of comparative table. The general positive and negative features of the considered software are founded. The system, which should be used as a prototype for further improvement, is selected.

Ключові слова: вибір завдань на виконання, контроль досягнення цілей, управління завданнями.

Вступ

Ми живемо у такому інформаційному середовищі, що розвивається дуже швидко – кожного року кількість інформації збільшується в кілька разів. За даними вчених у 2011 року кількість інформації досягла 1,8 зеттабайт (10^{21} байт) [1]. Якщо всі ці дані записати на компакт-диски, то вежа з дисків перевищить відстань до Місяця [2]. Щоденно мільйонам людей доводиться стикатися з великими обсягами інформації – від спеціалізованої літератури до новин в соціальних мережах. Відповідно зростає і кількість завдань щодо обробки цієї інформації, які постають перед кожним з нас.

Для того, щоб ефективно використовувати свій час та зусилля людині необхідно мати змогу оцінити ступінь досяжності тієї чи іншої мети, яку вона собі поставила, та обрати ті кроки, реалізація яких дозволить їй отримати бажаний результат з меншими витратами своїх ресурсів. Наявність, як правило, декількох шляхів (тобто множин завдань, які треба виконати) досягнення мети суттєво ускладнює цей вибір. Для полегшення становища людини у такому випадку вона повинна бути здатною швидше здійснювати прийняття рішень, ніж кілька років тому.

Постановка задачі

Якщо дану ситуацію розглянути з точки зору отримання конкретного результату, то при відборі завдань на виконання людині варто керуватися їх доцільністю відносно досягнення поставленої мети. Перед кожною людиною, що ставить перед собою певну мету, виникають завдання:

1. Конкретна постановка цілі, яку людина хоче досягти.
2. Вибір серед усіх можливостей тих пріоритетних завдань, що дійсно наближають до мети.
3. Мотивація, контроль виконання та оцінка прогресу.

Автоматизація вирішення цих завдань має спростити відповідь на питання «що мені варто зробити щоб досягти цієї мети?» для людей, які відчувають потребу у самореалізації. Особливо важливим це стає тоді, коли виділення потрібної інформації з потоку вхідних даних перестає бути тривіальною задачею. Отже, **актуальною** є задача створення програмної системи,