Expert Decision Support System Modeling in Lifecycle Management of Specialized Software

Yuliia Kordunova[0000−0003−0151−8285], Oleksandr Prydatko[0000−0002−0719−9118], Olga Smotr[0000−0003−2767−5019], and Roman Golovatyi[0000−0002−7895−9321]

Lviv State University of Life Safety, Lviv, Ukraine
{kordunovayulia,olexandrprydatko,olgasmotr,romangolovatiy}@gmail.com

Abstract. The paper describes the complexity of the short-term planning process in lifecycle management of specialized software. The critical stages of sprint scope planning in projects, which works by Agile models were explored. Network planning methods have been adapted to determine the critical indicators in lifecycle management of specialized software. An algorithm for automated construction of a network model and its representation in PC memory is proposed, as a data structure. Based on it, algorithms for constructing and traversing graphs of the network model, determining early and late execution time, and determining the critical execution path and time reserves are proposed. Developed algorithms for constructing and traversing graphs of the network model to automate the calculation of its parameters will be embedded in the work of the expert decision support system in lifecycle management of specialized software. The developed expert system will allow making operative decisions on re-planning of duration and the maintenance of project works in real-time. The system was developed using the Java programming language. The results of the system are presented in the development environment IntelliJ IDEA.

Keywords: software lifecycle management, Agile, sprint planning, network graphs

1 Introduction

It is known that the software development process includes many important stages: requirements analysis, planning, design, development and programming, testing, support and operation [10]. One of the crucial stages of development is planning, because at this stage the development team faces several crucial tasks: determining the timing of project development, choosing methods and means of development, establishing methods of implementation and more. If the latter have clear rules and instructions on how to implement a method, a set of practices and means of implementation, the issue of scheduling software development deadlines is a difficult task and requires a clear analytical calculation. The overall success of the project also depends on the quality and balanced assessment of the amount of work to be performed within the set timeframe.

Nowadays, a number of mathematical and software tools for project planning exist and are successfully used, including software development. However, existing methods are not adapted to a dynamic environment, where in addition to system requirements, the time resources of individual sprints or even the composition of teams may change during development. In such conditions, the application of classical approaches to planning IT projects is experiencing some difficulties.

As a result, there is a need to adapt existing or develop new planning methods for software development that can produce effective results in a dynamic environment. This work is devoted to this issue, namely the adaptation of network planning methods, as well as the development of algorithms for bypassing the graph of the network model.

2 Problem Statement

In general, the process of sprint planning (the next stage of the software life cycle) includes such stages as creating a product backlog, starting a sprint and determining the minimum viable product (MVP), creating a sprint backlog, prioritizing user stories (backlog tasks) and their evaluation. These processes are adapted and tested in traditional project teams working with flexible software development models [18]. These are teams of developers, numbering 5-9 people with the division of responsibilities by roles (positions), who spend most of the sprint's time on solving tasks. The dynamism of such a process depends only on a possible change of scope tasks. Team size and execution time are fixed (Figure 1) .

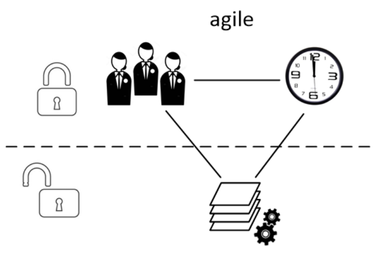

Fig. 1. Limitations on a flexible software lifecycle management model

However, if the focus is made on the teams of developers of operational formations, such as the State Emergency Service of Ukraine, which develops securityoriented services (SOS), the dynamics of specialized software development are characterized not only by the scope of work but also time resource. Some IT departments of operational formations, in addition to general tasks in the field of computer science and information technology, are engaged in the implementation of applied tasks for informatization of operational and daily activities, as well as design, development and support of computer and software systems security-oriented direction [15].

The members of such development teams in operational formations are the personnel of the relevant services, who, within the scope of their functional responsibilities, combine the activities related to the development of these services with other types of operational or service activities. According to the main participants' specifics of such teams, the dynamics of the project environment take on a slightly different meaning. Dynamics are now characterized not only by the volume of work but also by the time of their implementation. At first glance, we can assume that in such conditions, the process of developing specialized software should be organized according to the cascade model. However, this assumption is wrong, because the development of SOS is characterized by the dynamics of the specification and the need to constantly update the list of works during development. Under these conditions, the development of security-oriented services by project teams of operational (military) formations will acquire the model shown in Figure 2.

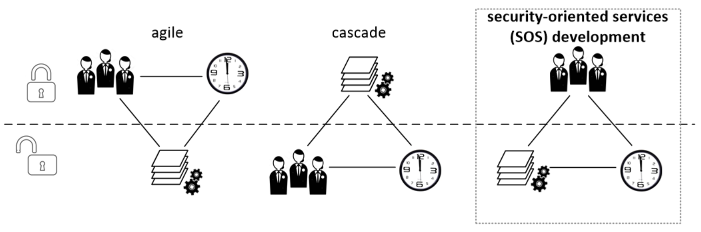

Fig. 2. Flexible and cascading software development models comparison with the actual SOS implementation model

From the presented model we can conclude that the dynamics of planning the volume of work on a particular sprint, as well as the lack of control (limitation) of time for their implementation, may be the cause of untimely or poor quality project implementation. That is why there is a need to study the existing methods of optimal planning of time resources for the development of SOS, which will correspond to the paradigm of flexible management and will allow responding quickly to deviations from the defined work plan. Therefore, the work aims to study existing methods and develop effective mechanisms (algorithms) for resource management of specialized software development projects (securityoriented services) in a dynamic environment (changing the content and scope of work, and adjusting execution time).

3 Main contribution

According to problem statement, the existing methods and management tools don't correlate with the conditions in which the development of specialized software is carried out, where in addition to variable requirements, execution time is fundamentally important.

So, according to the results of the study of existing methods of software life cycle planning was made a decision to use mathematical methods of planning. Based on it algorithms for constructing and bypassing the graph of the network model were developed for the first time.

These algorithms form the basis of an expert decision support system, which provides an opportunity to increase the efficiency of planning individual stages of software development and make their adjustments in real-time.

4 Literature Review

A large number of Ukrainian and foreign scientists worked on the problems of planning software development projects. In particular, in work [3] it was proved planning takes the center stage in Agile projects and other domains revolve around it. In work [11, 17] the importance of the planning process and expert evaluation of tasks during planning poker has been proved, and the scientific work [8] points out the importance of using the "task board" during planning. The scientific source [14] considered the necessary steps, which should be taken to get the most out of agile software development, and [7] suggested the use of a flowchart tool for decision-making in interdisciplinary research collaboration. The work [4] describes user story estimation based on the complexity decomposition using bayesian networks.

There are also many scientific papers [5, 19, 22] based on the use of network graphs in the project planning process In particular, the authors in [19] chose the method of solving the planning problem based on the application of the network planning method, which is based on the idea of optimizing the critical path with the involvement of additional limited funds. This approach does not correlate with the development of specialized software. Article [5] describes the process of developing a computer program for solving network optimization problems, but the calculation of the shortest paths is based on Dijkstra's algorithm, which is not universal and does not work for network planning of the software development process. The article [22] is based on the methods of PERT network planning, the use of elements of graph theory and the method of Gantt charts. This approach is relevant for the cascade management model, which is characterized by consistent execution of tasks and a certain execution time.

Based on papers [1, 2, 12, 20, 21], which lay the basic practices of modern product development planning and product planning practices in dynamic conditions, it was decided to develop flexible algorithms for the project planning process using network graphs in a dynamic environment. The work [20] proposes a planning framework in which multi-fidelity models are used to reduce the discrepancy between the local and global planner. This approach uses high-, medium-, and low-fidelity models to compose a path that captures higher-order dynamics while remaining computationally tractable. In work [21] was demonstrated an informational system or end-to-end workflow on time-series forecasting in Agile prosses. Work [2] based on the concept of a minimum viable product in product development, the new concept of a minimal viable production system (MVPS) is designed. The approach focuses on the reduction of inefficient planning processes due to changing product characteristics and aims to shorten the planning time and the level of maturity for defined planning tasks in order to facilitate early try out of production processes in a series production environment. Authors in work [1] propose a planning assistance platform based on solving the planning problem modeled as a constraint satisfaction problem (CSP). This helps project managers to analyze the project feasibility and to generate useful schedules and charts. In work [12] was argued and developed information graphics technologies of designing models of the processes of multiparameter technical systems in order to increase the effectiveness of determining the influence of many operating parameters on their dynamics, which help us to construct the network planning graph.

Authors in work [13] worked on developing security-oriented services by the hybrid management models. However, the paper describes only the results of the development and doesn't consider the key stages of software life cycle management. The algorithm development issue, in particular the security-oriented systems management, was also addressed by scientists in the work [16]. However, the algorithms developed in this work are difficult to adapt to the life cycle of specialized software management.

5 Materials and Methods

One option for achieving the goal is to use the mathematical apparatus of graph theory and adapt network planning processes by optimizing the calculations of its basic time parameters under the dynamic conditions of the specialized software development.

The adapted method of calculating the parameters of the network model will allow for operational re-planning of the project stages of implementation in a dynamic environment, as well as to determine the critical scope of mandatory work. Network planning will allow tracking in a dynamic environment of the possible risks of overtime, as well as prioritize the phasing of their implementation. Before proceeding to build a network graph, it is important to consider the process of planning design work for software development. Pay special attention to sprint planning. It should be emphasized, that the planning process consists of defining product requirements in the form of user stories [6]. In Agile methodology, it is not acceptable to break down requirements on technical tasks as such an approach does not allow developers to look integrally at the performance of certain program functions.

A user story is a short and simple description of product characteristics from the point of view of a user seeking new opportunities [9]. It includes the full range of development: from design to testing. Such an approach will allow the whole team to participate in the process of evaluating and developing this functionality. Once determined, the user story is broken down into small sub-tasks (functions) that will be performed directly by different team members. From this, it is clear that there is a high probability that several works can be performed in parallel (which will save a lot of time), and there may be work that depends on the previous ones. That is why determining the critical path that will take the most time to complete is extremely important.

Since works and events are the main elements of a network model in network planning, in the context of Agile scheduling, work is a user story or function that has been shredded. As a unit of measurement use story points are used, which means the relative value, in particular a combination of the development complexity, and the risk associated with it. Each user story or function must have a start and end, which means events on the network graph.

To obtain a network planning graph it is necessary:

1. Prioritize user stories and functions that need to be performed.

2. Evaluate user stories and functions in the story points

3. Identify previous user stories or functions to be performed for specific user story

At the beginning of building the network graph, number the initial event $j = 1$, and the previous event ($i = 0$) is absent. From it, draw a vector that will indicate the first user story or function and its weight in the Story Points. If several tasks can be performed in parallel, the required number of vectors is built. Since each user story must end with a final event, mark the event on the graph and increase its sequence number by 1 (do this for all parallel works, increasing j by 1).

If it is needed to build a user story, which is possible only after the previous one end, look for the final event of the work that interests you on the graph, and from it, draw the vector of the desired user story. If the execution of the user story depends on the completion of several previous ones, solve this problem using fictitious work. From each previous user story draw a vector, which means a fictitious work whose weight is 0 s.p., and end them in one final event, which will be the beginning of the new user story you need. Thus, build a network planning graph for the number of user stories and functions that interest you (within the scope of the sprint).

The algorithm for constructing a network graph (network model) of a certain amount of work on the sprint is shown in Algorithm 1.

Algorithm 1: The algorithm for constructing a network graph (network model) of a certain amount of work on the sprint

```
Initialization:
$prevTask[k][p]$ - array of previous works;
$pair < i, j >$, where $i = 1$, $j = 2$;
$n$- works, $n = 1$;
empty map $MapTask < n, pair < i, j >>$
for ($k = 0$; $k < prevTask.length$; $k++$) do
    for ($p = 0$; $p < prevTask[k].length$; $p++$) do
        if ($pverTask[k][p] == 0$) then
            Add $n$ and $pair < i, j >$ to $MapTask < n, pair < i, j >>$;
            $j++$;
            $n++$;
        end
    end
end
$i++$;
for ($k = 0; k < prevTask.length; k++$) do
    for ($p = 0; p < prevTask[k].length; p++$) do
        if ($prevTask.length == 1$) then
            for ($Map.Entry < n, pair < i, j >> works =$
              $MapTasks.entrySet()$) do
                if ($prevTask[k][p] == works.getKey()$) then
                    for ($Map.Entry < i, j > pairSet : pair.entrySet()$)
                      do
                        $i = paieSet.getValue()$;
                        Add $n$ and $pair < i, j >$ to
                          $MapTask < n, pair < i, j >>$;
                        $j++$;
                        $n++$;
                    end
                else
                    continue;
                end
            end
        else
            $j++$;
            for ($Map.Entry < n, pair < i, j >> works =$
              $MapTasks.entrySet()$) do
                if ($prevTask[k][p] == works.getKey()$) then
                    for ($Map.Entry < i, j > pairSet : pair.entrySet()$)
                      do
                        $i = pairSet.getValue()$;
                        Add fict. work 0 and $pair < i, j >$ to
                          $MapTask < n, pair < i, j >>$;
                        $j++$;
                    end
                else
                    continue;
                end
            end
            $i = j$;
            $j++$;
            Add $n$ and $pair < i, j >$ to $MapTask < n, pair < i, j >>$;
            $n++$;
        end
    end
end
Return $MapTask < n, pair < i, j >>$.
```

It should be recalled that the main elements of the network model are its events and works, which are evaluated in the story points. This type of estimation allows you to take into account the time and resource constraints of specific software development processes.

The main parameters of the network model for assessment of specialized software development processes at the planning stage include the early time of the event execution T_e the late time of the event execution T_l, event execution reserves R_e, and work performance reserves R_w. The algorithm for determining these parameters, which will be embedded in the work of the expert decision support system, will be discussed below.

In the first stage, the determination of the early time execution of events is implemented. This parameter characterizes the period before which the event can't occur and allows you to control the start of work that is not regulated by the order according to the network model. For the first event $T_{e(1)} = 0$, for all subsequent events of the network model T_e is determined from the expression:

$$T_{e(j)} = T_{e(i)} + t_{(i \to j)}, \tag{1}$$

where $T_{e(i)}$ – the early time for the previous event; $t_{(i \to j)}$ - the amount of work (unified resource) that precedes the event j (takes into account time and human constraints in Story Point).

t is necessary to provide the possibility of correct calculation of the early time execution for those events that are preceded by several works, taking into account the requirement $T_{e(i)} \to max$:

$$T_{e(j)} = max \begin{cases} T_{e(k)} + t_{(k \to j)} \\ ... \\ T_{e(m)} + t_{(m \to j)}, \end{cases} \tag{2}$$

where k, m are the indices of events preceding the event j.

Algorithmically, the procedure for determining the early term of work with all constraints can be represented as an algorithm for traversing the graph of the network model with an iterative definition of the parameter $T_{e(j)}$ (Algorithm 2), which was developed by the authors for the first time and adapted for flexible project management.

Algorithm 2: The algorithm for traversing the graph of the network model to determine the indicator $T_{e(j)}$

```
Initialization:
$i$ - index of the previous event, $i = 0$;
$j$ - index of the next event, $j = 1$;
$n$ - the volume of network model events;
$\{t_{i \to j}\}$- set of network model works;
$\forall j = 1$ then $T_{e(1)} = 0$ ;
$i + +$;
for ($j = i + 1; j \leq n; j + +$) do
    if ($i \to j \in \{t_{i \to j}\}$) then
        $T_{e(j)} = T_{e(i)} + \{t_{i \to j}\}$;
        if ($T_{e(j)} \in \{T_{e_j}\}$) then
            $max(\{T_{e_j}\}; \{T_{e(j)}\})$;
            if ($\{T_{e_j}\} < T_{e(j)}$) then
                $T_{prev(j)} \leftarrow T_i$;
                $HashMap < T_j, T_{prev(j)} > \leftarrow < T_j, T_{prev(j)} >$;
                $HashSet\{T_{e_j}\} \leftarrow T_{e(j)} >$;
            else
                continue $for$;
            end
        else
            $T_{prev(j)} = T_i$;
            $HashMap < T_j, T_{prev(j)} > \leftarrow < T_j, T_{prev(j)} >$;
            $HashSet\{T_{e_j}\} \leftarrow T_{e(j)}$;
            continue $for$;
        end
    else
        continue $for$;
    end
end
$i + +$;
if ($i \neq n$) then
    start $for$;
end
Return:
$\{T_{e_j}\}$ - the set of early term execution of $n$ event of the network model;
$HashMap < T_j, T_{prev(j)} >$ - the set of previous events for the next
 events in the format $< Key, Value >$.
```

The term of early execution of events for the last event of the network model n is the term of execution of the whole complex of sprint's works.

Next, to control the terms by which the events of the network model must be fully completed, to avoid cases of increasing time for the implementation of the entire project, it is necessary to determine the late time of event execution

$T_{l(i)}$:

$$T_{l(i)} = T_{l(j)} + t_{(i \to j)}, \tag{3}$$

where $T_{l(j)}$ - the late time of event execution; $t_{(i \to j)}$ - the amount of work performed after the event i and before event j (taking into account time and human constraints in Story Point).

The late time of event execution T_l for the last event of the network model is equal to the early time ot the event execution T_e:

$$T_{l(nlast)} = T_{e(nlast)} \tag{4}$$

Similar to the previous cases, when the next event i initialize the execution of $n \geq 2$ number of works, the late time of the event i is taken as follows:

$$T_{l(i)} = min \begin{cases} T_{l(k)} + t_{(i \to k)} \\ ... \\ T_{l(m)} + t_{(i \to m)}, \end{cases} \tag{5}$$

where k, m - indices of derived events, from the event i.

T_l for the first event of the network model, provided the correct execution of preliminary calculations, will be equal to the early time execution of this event T_e:

$$T_{l(nfirst)} = T_{e(nfirst)} \tag{6}$$

For all network model events, except the first and last, the following inequalities must be satisfied:

$$T_{l(n)} \geq T_{e(n)} \tag{7}$$

Algorithmically, the procedure for determining the late deadline for work, taking into account all the limitations, can be represented as an algorithm for traversing the graph of the network model with the iterative determination of the parameter $T_{l(i)}$ (Algorithm 3). The input data for the operation of the algorithm is the result of traversing the graph according to the previous algorithm.

Algorithm 3: The algorithm for traversing the graph of the network model to determine the indicator T_{l_i}

```
Initialization:
$i$ - index of the previous event, $i = 0$;
$j$ - index of the next event, $j = n$;
$n$ - the volume of network model events;
$\{t_{i \to j}\}$- set of network model works;
$\{T_{e_j}\}$- set of early time execution events;
$\forall j = n$ then $T_{l(n)} = T_{e(n)}$ ;
for $(i = j - 1; i \geq 1; i - -)$ do
    if $(i \to j \in \{t_{i \to j}\})$ then
        $T_{l(i)} = T_{l(j)} - \{t_{i \to j}\}$;
        if $(T_{l(i)} \in \{T_{l_i}\})$ then
            $min(\{T_{l_i}\}; \{T_{l(i)}\})$;
            if $(\{T_{l_i}\} < T_{l(i)})$ then
                $HashSet\{T_{l_i}\} \leftarrow T_{l(i)}$;
            else
                continue $for$;
            end
        else
            $HashSet\{T_{l_i}\} \leftarrow T_{l(i)}$;
            continue $for$;
        end
    else
        continue $for$;
    end
end
$j - -$;
if $(j \neq 1)$ then
    start $for$;
end
Return:
$\{T_{l_i}\}$ - the set of late term execution of $n$ event of the network model;
```

The results of the calculations allow us to estimate the maximum amount of work given the available resources needed to complete the sprint. The maximum amount of work in the network model reflects the critical path. The definition of the critical path is organized by the network of works $t_{(i \to j)}$ from the next late event to the event preceding it (the previous event of the vertex of the network graph specified in the corresponding sector). The construction of the critical path begins with the final event T_n and ends with the first event T_1:

$$T_n \to [t_{(j \to n)}] \to T_j \to [t_{(k \to j)}] \to T_k \to ... \to T_m \to [t_{(i \to m)}] \to T_i \to [t_{(l \to i)}] \to T_1, \tag{8}$$

where T_n - the last event of the network model (the first for the critical path); T_j - late intermediate event of the network model; T_i - early intermediate event of the network model; T_k, T_m - intermediate events of the network model; T_1 - the first event of the network model (the last for the critical path).

Algorithmically, the procedure for determining the critical path based on the results of traversing the graph of the network model is shown in Algorithm 4. The input data for this procedure are the results of previous models.

Algorithm 4: The algorithm for traversing the graph of the network model to determine the critical path

```
Initialization:
$j$ - index of the event, $j = n$;
$k$ - temporary variable;
$HashMap < T_j, T_{prev} >$ - the set of previous events for the next events
  $T_j$ in the format $< Key, Value >$.
$Stack.add(T_j)$;
while ($j \geq 1$) do
    $Stack.add(k)$;
    $j - -$;
end
Return:
$Stack$.
```

The peculiarity of the critical path of the execution of work is that it does not contain any resource reserve to achieve events (early and late terms of execution for all events at the critical level). This indicates that any delay in the implementation of network model events that lie within the critical path will encourage incomplete execution of a certain amount of work on the sprint or exceed the allowable time of their implementation.

However, other works which are included in the network model and do not belong to the critical path may have some reserves to achieve events and perform work. The value of these reserves will allow controlling the limits of the critical time of start and end of work that is not within the critical path. The calculation of these resource reserves is performed using the next expressions:

$$R_e(j) = T_{l(j)} - T_{e(j)} \tag{9}$$

$$R_{pw}(t_{(i \to j)}) = T_{l(j)} - T_{e(i)} - t_{(i \to j)} \tag{10}$$

$$R_{fw}(t_{(i \to j)}) = T_{e(j)} - T_{e(i)} - t_{(i \to j)} \tag{11}$$

$$R_{dw}(t_{(i \to j)}) = T_{e(j)} - T_{l(i)} - t_{(i \to j)}, \tag{12}$$

where $R_e(j)$ - the event time reserve; $R_{dw}(t_{(i \to j)})$ - the independent work reserve; $R_{pw}(t_{(i \to j)})$ - the full work reserve; $R_{fw}(t_{(i \to j)})$ - the free work reserve; $T_{l(i)}$ - late time of the previous event; $T_{l(j)}$ - the late time of the next event; $T_{e(i)}$ - the early time of the previous event; $T_{e(j)}$ - the early time of the next event; t - the scope of work; j - the next event of the network model; i - the previous event of the network model.

The algorithm 5 for traversing the graph of the network model to determine these reserves:

Algorithm 5: The algorithm for traversing the network model graph to determine time reserves

```
Initialization:
$i$ - index of the previous event, $i = 0$;
$j$ - index of the next event, $j = n$;
$n$ - the volume of network model events;
$\{t_{i \to j}\}$- set of network model works;
$\{T_{e_j}\}$- set of the early terms executed events;
$\{T_{l_j}\}$- set of the late terms executed events;
$Stack$ - the critical path of network model;
while ($j \geq 1$) do
    $R_{e(j)} = T_{l(j)} - T_{e(j)}$;
    if ($R_{(j)} == 0$) then
        $j - -$;
        continue $while$;
    else
        $HashSet\{R_j\} \to R_{(j)}$;
        $j - -$;
        continue $while$;
    end
end
$i \to n$;
for ($i = j - 1; i \geq 1; i - -$) do
    if ($i \to j \in \{t_{i \to j}\}$) then
        if ($T_i \notin Stack$) then
            $R_{pw}(t_{(i \to j)}) = T_{l_{(j)}} - T_{e_{(i)}} - t_{(i \to j)}$;
            $R_{fw}(t_{(i \to j)}) = T_{e_{(j)}} - T_{e_{(i)}} - t_{(i \to j)}$;
            $R_{dw}(t_{(i \to j)}) = T_{e_{(j)}} - T_{l_{(i)}} - t_{(i \to j)}$;
            $HashSet\{R_{pw}(t_{(i \to j)})\} \to R_{pw}(t_{(i \to j)})$;
            $HashSet\{R_{fw}(t_{(i \to j)})\} \to R_{fw}(t_{(i \to j)})$;
            $HashSet\{R_{dw}(t_{(i \to j)})\} \to R_{dw}(t_{(i \to j)})$;
            continue $for$;
        else
            continue $for$;
        end
    else
        continue $for$;
    end
end
$j - -$;
if ($j \neq 1$) then
    start $for$;
else
    Return:
    $\{R_{e(j)}\}, \{R_{pw}(t_{(i \to j)})\}, \{R_{fw}(t_{(i \to j)})\}, \{R_{dw}(t_{(i \to j)})\}$.
end
```

It should be emphasized that event time reserve $R_e(j)$ characterizes the maximum allowable period for which it is possible to delay the execution of event n without increasing the critical path and resource constraint on the sprint. The full reserve of work $R_{pw}(t_{(i \to j)})$ describes how long it is possible to postpone the start of work $t_{(i \to j)}$ or increase its duration within available resources and without increasing the total sprint time (critical path length). The free reserve of work $R_{fw}(t_{(i \to j)})$ is the time for which it is possible to postpone the start of execution of work $t_{(i \to j)}$ or extend its duration without violating the early term of execution events in the network model. The independent work reserve $R_{dw}(t_{(i \to j)})$ characterizes the delay time of the start of work $t_{(i \to j)}$ without increasing the total term of the sprint's tasks complex and without delaying any of the other works of the network model.

6 Experiment, Results and Discussion

The algorithms for automated construction and determination of network model parameters, when planning project works, were tested during the development of a security-oriented system commissioned by the State Emergency Service of Ukraine "Development consulting assistance software system to the population in case of threat or occurrence of emergencies with an integrated notification function based on mobile platforms". Input data and tasks for development were provided by the State Emergency Service of Ukraine during martial law, which confirms the work in dynamic conditions and limited time resources. After receiving the task, the project backlog was formed and determined MVP by the Product Owner. In order to determine the date of the first release, the team estimated user stories and shredded features in Story Points and determined the relationships between user stories, and the priority of execution. As a result, table 1.was formed .

Table 1. A set of input data for building a network graph

Number of task	1	2	3	4	5	6	7	8	9	10	11
Story point	27	35	15	12	25	30	17	13	15	27	35
Number of previous task	0	0	1	1	2	3,4	5	5	5	7,8,9	6,10

By implementing the algorithm of constructing a network graph (network model) of a certain amount of work on the sprint, the procedure of constructing a network graph in the IntelliJ IDEA development environment by the Java programming language was implemented. Figure 3 shows the result of building a network model of project works identified in the sprint.

To properly characterize all software development processes at the planning stage, it is necessary to determine all the parameters of the sprint's network model. Initially, graph traversal algorithms were implemented to determine the

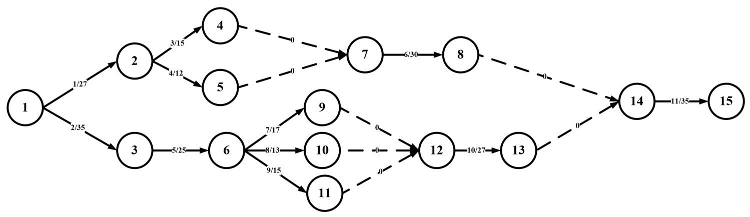

Fig. 3. The network model of the software development process, presented in the graph form

early and late execution times of sprint (project) events. On the basis of algorithms 2 and 3 the iterative process of specified parameters calculation is programmatically realized.

Based on the results of calculating the early and late time execution events, the critical path is programmatically determined (according to Algorithm 4). In Figure 4 the result of critical path automated determination of the studied network model is presented. The obtained result indicates events that are critical and the implementation of which does not involve time reserves.

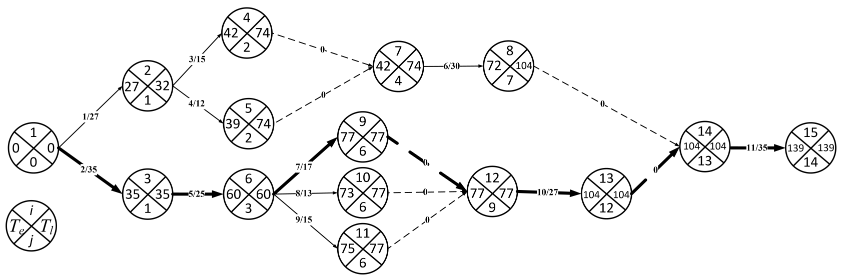

Fig. 4. The result of the critical path, the early and late execution times of events calculation of the software development

The last step is to determine the event reserves using time reserve algorithm (Algorithm 5). Table 2 shows the result of the specified algorithm of the software implementation and its application to determine time reserves for the studied network model.

Table 2. The result of a program that calculates time reserves

$Event$	2	4	5	7	8	10	11
R_e	32	32	35	32	32	4	2

Based on the results, we can conclude that the development period of the MVP of this software is 139 Story Points. The critical path is the path that passes through events 1, 3, 6, 9, 12, 13, 14, and 15. Events 2, 4, 5, 7, 8, 10, and 11 have time reserves. If you need to make changes to the list of works or adjust the time to perform a certain amount of work, the developed system for estimating the parameters of the network model based on the developed algorithms will instantly re-evaluate the main characteristics and anticipate possible deviations from the plan. This subsystem forms the basis of the decision support system for the operational management of the developing specialized software process in a dynamic environment. After all, it will allow you to monitor the critical path of work, available time reserves, and the amount of unfinished work in real-time.

7 Conclusions

In dynamic conditions, the planning stage of specific software development processes is extremely important. As the experience of such development has shown, the existing methods and management tools do not correlate with the conditions in which the development of specialized software is carried out, where in addition to variable requirements, execution time is fundamentally important. According to the results of the study of existing methods of software life cycle planning in scientific work, the following results were obtained:

1. Based on the results of optimization of mathematical methods of network planning for life cycle management processes of specific software, algorithms for constructing and bypassing the graph of the network model are developed to determine its main parameters, which provide an opportunity to increase the efficiency of planning individual stages of software development and make their adjustments in real time.

2. Based on the scientific results and implemented algorithms, an expert computer system was obtained, which allows determining the basic parameters of the network model and is used to support operative decision-making in the process of short-term life cycle planning of specialized software development projects.

References

1. Belkasmi, M.G., Bougroun, Z., Farissi, I.E., Emharraf, M., Belouali, S., Chadli, S., Saber, M.: Global IT project management: An agile planning assistance. In: Advances in Smart Technologies Applications and Case Studies, pp. 575–582. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-53187-4 ˙63
2. Bertling, M., Caroli, H., Dannapfel, M., Burggraf, P.: The minimal viable production system (mvps) – an approach for agile (automotive) factory planning in a disruptive environment. In: Advances in Production Research. pp. 24–33. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-03451-1 ˙3
3. Boral, S.: Domain V: Adaptive Planning (12 2016). https://doi.org/10.1007/978-1-4842-2526-4 ˙6

4. Durán, M., Juárez-Ramírez, R., Jiménez, S., Tona, C.: User story estimation based on the complexity decomposition using bayesian networks. Programming and Computer Software **46**, 569–583 (2020). https://doi.org/10.1134/S0361768820080095
5. Dymova, H., Larchenko, O.: Development of a computer program for solving network optimization problems. Computer-integrated technologies: education, science, production **41**, 143–151 (2020). https://doi.org/10.36910/6775-2524-0560-2020-41-23
6. Hallmann, D.: "i don't understand!": Toward a model to evaluate the role of user story quality. In: Stray, V., Hoda, R., Paasivaara, M., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming. pp. 103–112. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-49392-9 ˙7
7. Jansen, U., Schulz, W.: Flowchart tool for decision making in interdisciplinary research cooperation. In: International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. pp. 259–269. Springer (2017). https://doi.org/10.1007/978-3-319-58466-924
8. Karras, O., Klünder, J., Schneider, K.: Is task board customization beneficial? In: International Conference on Product-Focused Software Process Improvement. pp. 3–18. Springer (2017). https://doi.org/10.1007/978-3-319-69926-4 ˙1
9. Khmel, M., Prydatko, O., Popovych, V., Tkachenko, T., Kovalchuk, V.: Students r&d projects as a tool for achieving program competencies. Information and communication technologies in modern education: experience, problems, prospects ... (2021)
10. Kordunova, Y., Smotr, O., Kokotko, I., Malets, R.: Analysis of the traditional and flexible approaches to creating software in dynamic conditions. Management of Developement of Complex Systems pp. 71–77 (2021). https://doi.org/10.32347/2412-9933.2021.47.71-77
11. Lenarduzzi, V., Lunesu, M.I., Matta, M., Taibi, D.: Functional size measures and effort estimation in agile development: A replicated study. vol. 212 (05 2015). https://doi.org/10.1007/978-3-319-18612-2 ˙9
12. Liaskovska, S., Martyn, Y., Malets, I., Prydatko, O.: Information technology of process modeling in the multiparameter systems. pp. 177–182 (08 2018). https://doi.org/10.1109/DSMP.2018.8478498
13. Martyn, Y., Smotr, O., Burak, N., Prydatko, O., Malets, I.: Software for shelter's fire safety and comfort levels evaluation. In: Communications in Computer and Information Science, pp. 457–469. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-61656-4 ˙31
14. Meier, A., Ivarsson, J.C.: Agile software development and service science. GSTF Journal on Computing (JoC) **3**(3), 1–5 (2013). https://doi.org/10.7603/s40601-013-0029-6
15. Prydatko, O., Kordunova, Y., Kokotko, I., Golovatiy, R.: Substantiation of the managing student r&d projects methodology (on the example of the educational program computer science). Information and communication technologies in modern education: experience, problems, prospects ... (2021)
16. Prydatko, O., Popovych, V., Malets, I., Solotvinskyi, I.: Algorithm of rescue units logistic support planning in the process of regional life safety systems development. MATEC Web of Conferences **294**, 04002 (01 2019). https://doi.org/10.1051/matecconf/201929404002
17. Sachdeva, V.: Requirements prioritization in agile: Use of planning poker for maximizing return on investment. In: Advances in Intelligent Systems and Computing, pp. 403–409. Springer International Publishing (Jul 2017). https://doi.org/10.1007/978-3-319-54978-1 ˙53

18. Schwaber, K., Sutherland, J.: The scrum guide. Scrum Alliance (2020)
19. Seidykh, O., Chobanu, V.: Optomozation of the network graphics of the complex of works. Modern engineering and innovative technologi **1**(3), 61–67 (2018). https://doi.org/10.30890/2567-5273.2018-03-01-009
20. Tordesillas, J., Lopez, B.T., Carter, J., Ware, J., How, J.P.: Real-time planning with multi-fidelity models for agile flights in unknown environments pp. 725–731 (2019). https://doi.org/10.1109/ICRA.2019.8794248
21. Uzun, I., Lobachev, I., Gall, L., Kharchenko, V.: Agile Architectural Model for Development of Time-Series Forecasting as a Service Applications, pp. 128–147. Springer International Publishing (07 2021). https://doi.org/10.1007/978-3-030-82014-5 ˙9
22. Velykodniy, S., Burlachenko, Z., Zaitseva-Velykodna, S.: Architecture development of software for managing network planning of software project reengineering. Innovative technologies and scientific solutions for industries. **2**(8), 25–35 (2019). https://doi.org/10.30837/2522-9818.2019.8.025