January 26, 2018

Testing enviroments & optimization
Company: Amdocs

Adenilso da Silva Simao (ICMC), Andre Carvalho (Amdocs), Carlos Diego Nascimento Damasceno (ICMC), Danniany dos Santos (Amdocs), Edson dos Santos Moreira (ICMC), Fabio Tomita (Amdocs), Giovana Sachett Maia (ICMC), Hanna Pamplona Hortencio (Poli), Idan Nakel (Amdocs), Isabela Peronti (Amdocs), Jamielli Tomaz Pereira (UNICAMP), João Paulo Guardabaxo Siqueira (EESC), Jorge Francisco Cutigi (IFSP), Landir Saviniec (ICMC), Leandro Resende Mundim (ICMC), Lucas Esperancini Moreira e Moreira (ICMC), Luis Eduardo de Freitas (Amdocs), Luiz Henrique Cherri (ICMC), Maria Carolina dos Santos (Amdocs), Marina Barisa de Oliveira (ICMC), Misael Costa Junior (ICMC), Pratibha Chandekar (Amdocs), Raínne Florisbelo Gonçalves (ICMC), Renan Brito Butkeraites (UNIFESP), Ricardo Gesuatto (Amdocs), Rohit Goel (Amdocs), Sergio Mendonça (Amdocs), Stevão Alves de Andrade (ICMC), Thais Cardoso (Amdocs)

Summary:

The planning of test effort in complex projects is a challenge faced by several companies which develop software. Many constraints, such as the cost for the integration with legacy systems, the requirements of data integration, should be considered when the various scenarios are distributed among a set of environments. Amdocs is one of market leaders in Software for Telecommunication industry and the complex business handled by its systems requires several and complex test environments. This work aims at proposing an approach to help planning test effort with the involvement of multiple test scenarios. The approach is organized in a pipeline, with three main processes: (1) the a spreadsheet in the suitable format for testing engineering manipulation is **pre-processed**, generating the input to the (2) **optimization heuristics**, which provide the scheduling planning; (3) finally, the scheduling is **post-processed** to create a report which can be used by the test managers.

Contents

1 Introduction 3

2 Problem definition 3

3 Mixed Integer Linear Programming Model 5

4 Heuristic solution approach 7

5 Validation 8

6 Achieved Results 10

7 Final remarks 12

A Notes from the company 13

1 INTRODUCTION

Testing complex systems is a challenge faced by most of the IT companies that work with software development. Due to the large volume of information and other variables involved, the development usually becomes expensive. Moreover, it is common to employ more work to test systems' components than to develop themselves. The occurrences of errors during test operations may delay the process, and therefore, they must be minimized.

In complex systems, the test process is divided into a number of different scenarios that are tested (executed) in computational environments. Usually, each scenario involves several steps and different scenarios may share one or more identical steps. Moreover, particular features of the scenarios, such as, *the need of integration with legacy systems* and *the requirement of data* should be considered. Different environments may also have different setup times. Thus, it is possible to save some work if the scenarios are properly scheduled to specific environments.

In other words, the problem consists in scheduling a set of scenarios to be tested into a set of environments, considering test constraints, so that the time required to process all scenarios is minimized.

The problem was proposed by the IT company *Amdocs*, which is one of the market leaders in development of software for telecommunication companies. The company has an workforce of about 25,000 employees who work to provide software solutions to support the businesses of their customers in 85 countries around the world.

2 PROBLEM DEFINITION

The problem can be modeled as the problem of scheduling tasks to be processed in parallel machines. The scheduling problem deals with the allocation of resources to tasks and targets to optimize one or more objectives [2].

The Amdocs problem is a system with m identical machines in parallel. There are n tasks that have different processing times and due dates. Setup time are sequence dependent. There is an extra effort to add features to machines, such as integration with legacy systems and data.

There are four terms in the objective function, with the intent to minimize: the makespan (completion time of the last job), the number of machines, the number of delayed tasks and the number of machines that need integration or data required to process some tasks. This problem is NP-hard [2].

The approach is organized in a pipeline composed of three processes, as depicted in Figure 1:

1. Pre-Processing
2. Optimization
3. Post-Processing

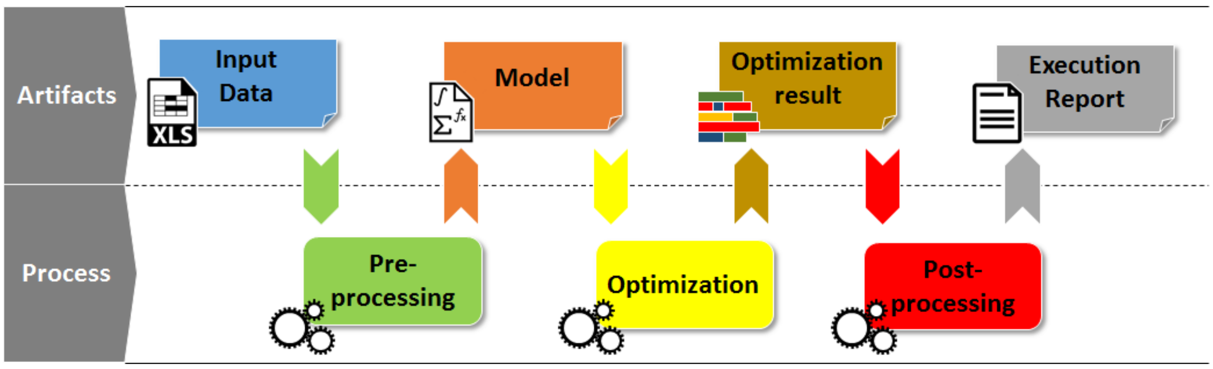


FIGURE 1: WORKFLOW.

The pre-processing converts the spreadsheet prepared by the test manager into a set of parameters for the optimization. Each sheet of the spreadsheet is detailed as follows.

- Scenarios
  Each scenario is given a unique id and a short description. The id is used throughout the document.

- Steps
  The functionality to be tested has a unique id and a set of parameters, defining its estimated execution time, its estimated development effort, the requirements of integration and/or data. The time/effort required for integration and for data inclusion is assumed to be additive.

- Scenario-Steps
  The sequence of steps of each scenario is provided.

- Env
  For each environment, a set of parameter is given, such as the cost of setup, integration and inclusion of data.

- Step Relation
  In some cases, the execution of one step can invalidate the data produced by another step. Pre-conditions specify existing conflicts between steps and constraints that must hold before a step is provided [1]. These are recorded in the *Env* sheet.

The information from the spreadsheet is analyzed and converted into the parameters for the optimization. For each scenario, the estimated execution time is computed by the sum of the estimated execution time of its steps; the requirements of the scenario (i.e., whether integration or data are required) are computed such that a feature is required if any of the steps requires it.

Then, each pair of scenarios is analyzed;

3 MIXED INTEGER LINEAR PROGRAMMING MODEL

In this section, we present the Mixed Integer Linear Programming model for the problem presented in the Section 2. First, the parameters used as input in the model are shown. Then, the variables are defined. Finally, the complete model is presented.

Parameters:

- α: coefficient to adjust the desire of reduce the makespan;
- β: coefficient to adjust the desire of reduce the number of environments;
- γ: coefficient to adjust the desire of reduce the number of environments with integrations;
- θ: coefficient to adjust the desire of reduce the number of lateness scenarios;
- n: number of scenarios;
- m: number of environments;
- l: number of integration types;
- V: a large number;
- $envCost$: cost of open an environment;
- $intCost_t$: cost of include the integration t into an environment for all $t \in= 1, ..., l$;
- p_i: processing time of scenario i for all $i \in= 1, ..., n$;
- s_{ij}: the setup time to process the scenario j immediately after the scenario i for all $(i, j) \in= 1, ..., n$. The setup processing time for $i = j$ is the initial setup processing time for open an environment k, for all $k \in= 1, ..., m$;
- $dueDate_i$: the due date of scenario i, for all $i = 1, ..., n$;
- λ_{jt}: 1 if scenario j require the integration of type t, for all $j = 1, ..., n, t = 1, ..., l$.

Decision variables:

- C_{max}: time to process all the scenarios (makespan);
- c_i: completion time of scenario i;
- y_k: 1 if environment k is open, 0 otherwise;
- z_{kt}: 1 if integration t must be implemented in environment k, 0 otherwise;
- x_{ijk}: 1 if scenario j is immediately after scenario i in environment k, 0 otherwise;

- L_i: 1 if occurs a lateness in scenario i, 0 otherwise.

MIXED INTEGER PROGRAMMING MODEL

$$\min \quad \alpha C_{max} + \beta \sum_{k=1}^{m} (envCost) y_k + \gamma \sum_{k=1}^{m} \sum_{t=1}^{l} (intCost_l) z_{kl} + \theta \sum_{t=1}^{l} L_t \tag{1}$$

$$\begin{aligned}
s.t. \quad & \sum_{i=0}^{n} \sum_{k=1}^{m} X_{ijk} = 1, && j = 1,...,n, && (2)\\
& \sum_{i=0}^{n} X_{ihk} = \sum_{j=0}^{n} X_{hjk}, && h = 1,...,n, k = 1,...,m, && (3)\\
& \sum_{j=0}^{n} X_{0jk} \leq y_k, && k = 1,...,m, && (4)\\
& \sum_{i=0}^{n} \sum_{j=0}^{n} \lambda_{jt} X_{ijk} \leq V z_{kt}, && k = 1,...,m, t = 1,...,l, && (5)\\
& p_i X_{0ik} \leq c_i, && k = 1,...,m, i = 1,...,n, && (6)\\
& c_j \geq c_i + s_{ij} + p_j + V(\sum_{k=1}^{m} X_{ijk} - 1), && i = 1,...,n, j = 1,...,n, && (7)\\
& c_i \leq dueDate_i + V L_i, && i = 1,...,n, && (8)\\
& c_i \leq C_{max}, && i = 1,...,n, && (9)\\
& X_{ijk} \in \{0,1\}, && i = 1,...,n, j = 1,...,n, k = 1,...,m, && (10)\\
& y_k \in \{0,1\}, && k = 1,...,m, && (11)\\
& z_{kt} \in \{0,1\}, && k = 1,...,m, t = 1,...,l, && (12)\\
& L_i \in \{0,1\}, && i = 1,...,n, && (13)\\
& c_i \geq 0, && i = 1,...,n, && (14)\\
& C_{max} \geq 0. && && (15)
\end{aligned}$$

In the model, the objective function (1) minimize the convex combination of makespan, the cost of open an environment, the cost to include integration on the environments and the number of lateness scenarios. Constraints (2) guarantee that each scenario is processed once. Constraints (3) guarantee that, if scenario h is processed in an environment k, there is a predecessor and a successor scenario for this scenario h. If scenario h is the first to be processed in an environment k, thus the predecessor is the artificial scenario 0; else if scenario h is the last to be processed in an environment k, thus the successor is the artificial scenario 0. Constraints (4) indicate that, if exists at least one scenario to be processed in an environment k, this environment must be open. Also, the maximum number of scenarios that can be the first to be processed is equal to one. If scenario j processed in environment k requires an integration t, Constraints (5) imposes that integration t is implemented in environment k. Constraints (6) and (7) guarantee that there is no subtours between scenarios processed in the same environment. These constraints are known as Miller, Tucker e Zemlim (MTZ) constraints in the literature. Constraints (8) indicate if the complete process time of scenario i is greater than its respective due date. Constraints (9) points out the greatest procession time among scenarios (makespan). Variables domains are shown in Constraints (11) - (15).

4 HEURISTIC SOLUTION APPROACH

In this section, we propose a heuristic solution approach to compute high-quality schedules to the problem. The approach is particularly interesting when the input data of the problem is large. Namely, there is a large number of tasks (scenarios) to be scheduled to several machines (environments).

We proposed a two phase approach, as shown in Figure 2. The approach consists in a constructive phase followed by an improvement phase. The first phase consists in constructing an initial solution with the input data received from the Amdocs. To construct the initial solution, we employ the greedy algorithm shown in Algorithm 1. The second phase consists in improving the initial solution constructed in the first phase. To improve the initial solution, we employ a Variable Neighborhood Search (VNS) metaheuristic. The VNS pseudo-code in shown in Algorithm 2.

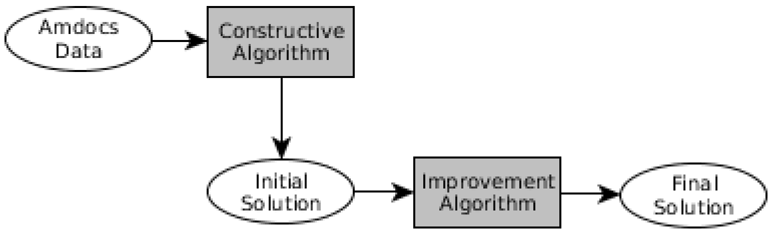


FIGURE 2: THE TWO PHASE APPROACH.

Algorithm 1 Pseudo-code of the constructive algorithm.

```
CONSTRUCT-SOLUTION()
1  Let $T$ be the list of tasks (scenarios).
2  Let $M$ be the list of machines (environments).
3  Initialize an empty solution $Z$.
4  while ($T \neq \emptyset$) do
5      Choose a random task $t \in T$.
6      Choose the machine $m \in M$ with the shortest completion time.
7      Assign the task $t$ to machine $m$.
8      Update the completion time of machine $m$.
9  return $Z$
```

Algorithm 2 Pseudo-code of the VNS improvement algorithm.

```
VNS($Z$, $iters$, $oper$, $ls$)
 1  $Z^* = Z$
 2  $i = 0$
 3  while ($i < iters$) do
 4      $k = 0$
 5      do
 6          $Z = oper[k].perturbation(Z^*)$    // perturb the best solution using the k-th neighborhood
 7          $Z = ls[k].run(Z)$                 // apply a local search using the k-th neighborhood
 8          if $f(Z) < f(Z^*)$ then
 9              $k = 0$
10          else
11              $k = k + 1$
12          if $f(Z) \leq f(Z^*)$ then
13              $Z^* = Z$
14      while ($k < ls.size()$)
15      $i = i + 1$
16  return $Z^*$
```

5 VALIDATION

To validate our approach, we defined a template to the input file. The input file is a Microsoft Excel spreadsheet in `xlsx` format containing all data required to our soution.

The spreadsheet has the following information:

- **Maximum number of environments**: describes the maximum number of environments available for processing data.
- **Time/cost for creating an environment**: each environment has a cost of creation. This parameter describes the cost for creating one new environment.

- **Number of scenarios to be allocated**: The number of scenarios that has to be allocated at the test environments.

- **Duration of each scenario (in hours)**: each scenario has an estimated execution time.

- **Complexity degree of each scenario**: depending on the steps to be run, scenarios have different complexity degrees.

- **Total number of types of integrations**: some types of scenarios can have specific requirements, such as integration with external systems. This parameter describes the total number of types of external integration

- **Cost of integration**: existe um custo de integração, para cada operação externa realizada. there is a cost for each type of integration. This parameter describes the cost for each type of integration to the scenarios.

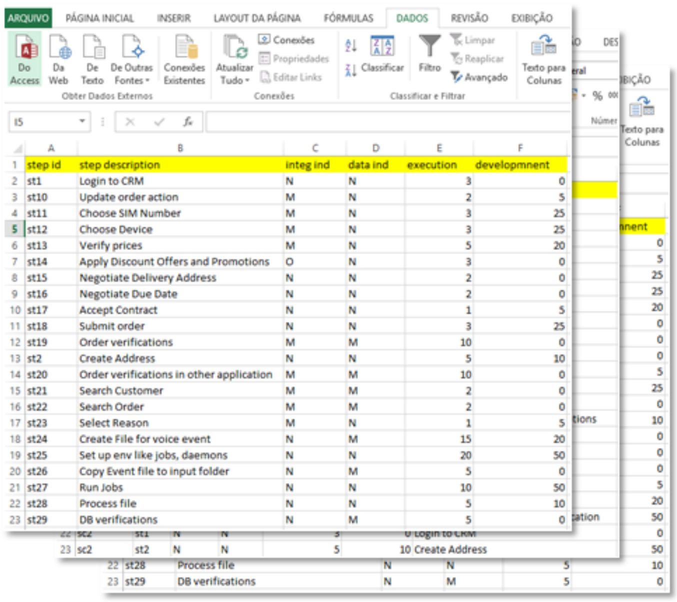
- **Integration × scenario matrix**: It describes the relation between the scenarios and the required integration to external systems.

- **Setup matrix**: Given the steps of each test scenario, it describes the gain of executing two scenarios in a row due to similar steps shared.

- **Number of testers**: Describes the number of testers available to execute tasks.

- **Skill level of the testers**: Describes the skill level of the testers to perform tasks.

Amdocs provided an `.xlsx` spreadsheet containing all the data aforementined, as shown in Figure 3.



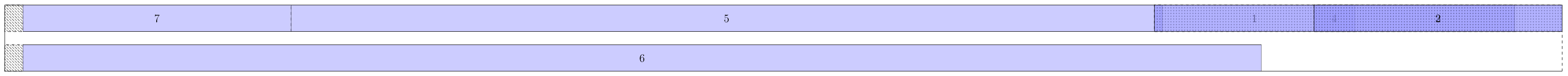
	A	B	C	D	E	F
1	step id	step description	integ ind	data ind	execution	developmnent
2	st1	Login to CRM	N	N	3	0
3	st10	Update order action	M	N	2	5
4	st11	Choose SIM Number	M	N	3	25
5	st12	Choose Device	M	N	3	25
6	st13	Verify prices	M	N	5	20
7	st14	Apply Discount Offers and Promotions	O	N	3	0
8	st15	Negotiate Delivery Address	N	N	2	0
9	st16	Negotiate Due Date	N	N	2	0
10	st17	Accept Contract	N	N	1	5
11	st18	Submit order	N	N	3	25
12	st19	Order verifications	M	M	10	0
13	st2	Create Address	N	N	5	10
14	st20	Order verifications in other application	M	M	10	0
15	st21	Search Customer	M	M	2	0
16	st22	Search Order	M	M	2	0
17	st23	Select Reason	M	N	1	5
18	st24	Create File for voice event	N	M	15	20
19	st25	Set up env like jobs, daemons	N	N	20	50
20	st26	Copy Event file to input folder	N	M	5	0
21	st27	Run Jobs	N	N	10	50
22	st28	Process file	N	N	5	10
23	st29	DB verifications	N	M	5	0

FIGURE 3: INPUT SPREADSHEET TEMPLATE

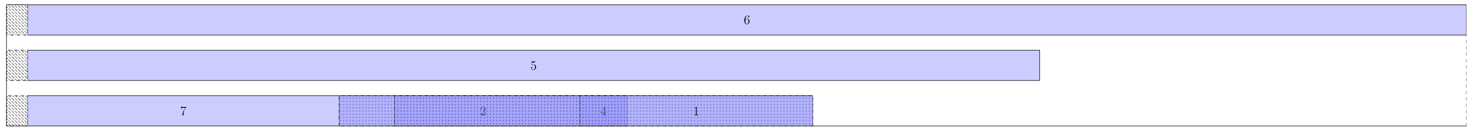
To format the data and calculate secondary information (i.e., derivable from the aforementioned parameters), we implemented an R script. This R script was responsible for reading the `.xlsx` file and generating the input file to the optimization algorithms.

6 ACHIEVED RESULTS

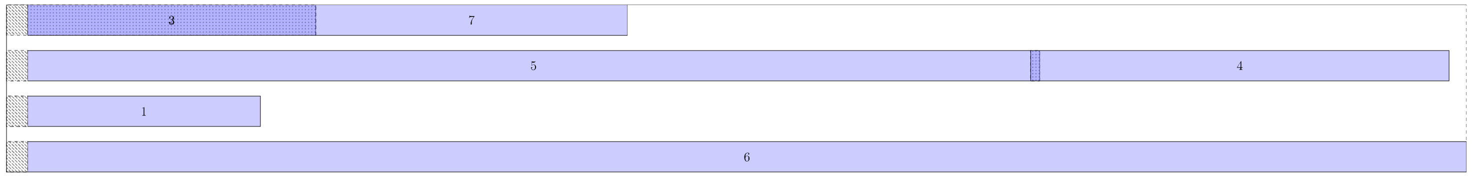
In this section, we report some simulations with data provided by the company Amdocs. The Figure 4 shows three different simulations made with the proposed heuristic approach. In Figure 4(a), we simulated a case with 2 environments. For this simulation, the heuristic provided a solution in which all scenarios were processed within 598 hours. In Figure 4(b) and 4(c), we simulated cases with 3 and 4 environments, respectively. For these simulations, the heuristic provided the same solution quality and all scenarios were processed within 482.5 hours. Note that the last cases represent a reduction in completion time compared to the first case, and they are of interest if the decision maker aims to minimize the time to process all test scenarios.



(a) Simulation with 2 environments (completion time = 598 h).



(b) Simulation with 3 environments (completion time = 482.5 h).



(c) Simulation with 4 environments (completion time = 482.5 h).

FIGURE 4: HEURISTIC SIMULATIONS WITH AMDOCS DATA.

7 FINAL REMARKS

We can highlight the importance of the workshop for undergraduate, masters, doctoral students, as well as, for the faculties and other researchers involved. The workshop was an important learning process, providing direct contact with real world problems and the possibility of developing teamwork skills.

It is important to mention that the company had an active participation in the project, taking part in the whole process of modeling and resolution of the problem, responding promptly all questions that arose during the process. The experiment included a session of work in the company's premises, with the participation of top engineers responsible for the testing sector.

We understand that the whole interaction, from the definition of the problem, well before the workshop, to the delivery of the solution was very rich in the sense to visualize the potential of applications of both theoretical and practical knowledge created at CEPID-CeMEAI.

We would also to point out some future improvements which could add functionalities and value to the solution: i) in the Optimization area - the solution could be improved by refining the both the exact and the heuristic approaches; ii) in the Software Engineering area - testing tools could be developed to assist the testers team to explore the capabilities made available by the solution, with eventual automation of parts of the process; iii) in the Artificial Intelligence area - knowledge management techniques could be used to extract relevant expertise information from the history database to be used in the refinement of future tests.

REFERENCES

[1] Patrícia Machado, Auri Vincenzi, and José Carlos Maldonado. Software testing: An overview. In Paulo Borba, Ana Cavalcanti, Augusto Sampaio, and Jim Woodcook, editors, *Testing Techniques in Software Engineering: Second Pernambuco Summer School on Software Engineering, PSSE 2007, Recife, Brazil, December 3-7, 2007, Revised Lectures*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[2] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 4rd edition, 2012.

A NOTES FROM THE COMPANY

1. Scenarios (Sc1,...,Sc5)
   - Creation of customer
   - Process of pagment file
   - Produce bill for the customer
2. Steps
   - A
     - Sc1 –> St1, St2
     - Sc2 –> St1, St2, St3, St4,...
     - Sc3 –> St1, St2, St3, St4, Sc17
   - B
     - St1 –> creation of content
     - X –> execution time – 2 min
     - V –> dev effort – ∅
     - T –> integration time – (M, N, O)
     - D –> data required – (Y, N)