Modeling and Analysing Operation Processes for Dependability

Xiwei Xu, Liming Zhu, Jim(ZhanWen) Li, Len Bass, Qinghua Lu, Min Fu
NICTA Sydney Australia
School of Computing Science and Engineering, University of New South Wales, Sydney Australia
firstname.lastname@nicta.com.au

Abstract—Application dependability issues depend on increasingly sophisticated activities during operation time for deployment, upgrade, scaling out/in and reactions to various failures. Traditional approaches to improving application dependability focus on artifact-oriented troubleshooting and improvements. In this paper, we present an approach using process models to represent and analyze operations with considerations of exception handlings and fault-proneness. Our goal is to reduce diagnosis and repair time for application failures that occur during operation activities such as deployment and upgrade.

I. INTRODUCTION

The interaction between operations and dependability has been recently emphasized by industry. Gartner has stated “Through 2015, 80% of outages impacting mission-critical services will be caused by people and process issues, and more than 50% of those outages will be caused by change/configuration/release integration and hand-off issues.” [1]. It is not so much that dependability issues may trigger operator actions, but rather the converse – i.e., dependability is often affected by operator actions. Many operator actions (e.g. backup, snapshot, maintenance), even performed correctly, may have performance and availability impact. Other operator actions (e.g. upgrade, failover, redeployment) are error-prone processes and require time-consuming troubleshooting, undoing some steps, problem fixing and redoing the undone steps. These error-prone processes and complicated exception handlings all have huge impact on the repair time for a failure.

In the past, research has been done on troubleshooting operation errors from an artifact and provenance point of view such as linking issues back to source code, configuration/log analysis [2] and provenance analysis [3]. However, these types of analyses do not view operations as a process with exception handling. Other work focuses on specific issues in rolling upgrade [4] and backup to improve scheduling and correctness. Our previous work considered these actions as black boxes and analyzed their impact on overall availability using SRN models [5]. Treating them as black-box actions and assuming they perform correctly has severe limitations on the real world applicability of the analysis results.

In this paper, we outline a process-oriented approach to modeling and analyzing operations for dependability. We model an operation as a set of steps executed by agents (automated scripts/tools or human) requiring various resources (computing power, readied environment, nodes, etc.). We can then populate the models with empirical data gathered either from one’s own environment or from literature on fault types/probability. The data gathered includes troubleshooting time and the actual time to repair. We intend to use the models and its analysis for two purposes: 1) to understand and monitor actual operations’ processes to help error diagnosis and recommend actions, and 2) to conduct sensitivity analysis to determine which actions can be improved the most to affect completion time, overall dependability and zero down time for critical portions. In this paper, we outline the approach and report some initial observations.

II. OPERATION PROCESS MODEL

There are many process modelling languages ranging from informal modelling of business processes (e.g. BPMN) to formal analysis of processes (e.g. petri-nets). We have chosen a language called Little-JIL [6] due to some of its unique features useful to our approach. We use its agent concepts to model human operators or scripts and its flexible resource concepts to model the resources both being operated on and required for computing power. We use its tree structure and its abstraction and recursion concepts to only model certain steps to necessary details with recursive reuses. Its strong exception management mechanism, pre/post-conditions and flexible parameter passing among steps are also useful. Our initial experiments involve the deployment of a typical HBase cluster on AWS EC2s.

Fig. 1 models the deployment process of HBase using Little-JIL. The process is a hierarchical decomposition of activities (called steps in Little-JIL), which are denoted graphically by black rectangular bars. The process is executed from left to right, in the depth-first order. The sequencing badge (in blue color) on the left side of the step bar represents the sequencing for a step. The exceptions that can be thrown by a step are marked with a red cross. The exception handler itself is a process model shown in the bottom-left corner. Normal exception propagation is assumed.

Due to space limitation, we cannot show the resource model. In Little-JIL, an agent (special resource) is an autonomous entity that is an expert in some part of the process. An agent could be a human engineer or automated deployment tools or script. Other resources include cloud infrastructure resources and required artifacts such as installation package and configuration files. Different resources will be modelled
with different attributes including their fault-proneness (from empirical data), availability and capabilities.

Troubleshooting is a key issue in operations relating to dependability. We explicitly consider key steps of troubleshooting processes in the exception handler. For example, some faults will only manifest themselves as a failure later in the process with significant error diagnosis time. Its fixing may require to first revert some actions and then redo the actions after the fixing. The exception handling process may have exceptions itself recursively expressed in the model.

III. INITIAL ANALYSIS AND VALIDATION

Our initial analysis and validation consists of three parts. First, we conducted several experiments to deploy HBase in Amazon EC2 using both manual and automated activities (Whirr/Cloudera and EMR). We collected specific exceptions, their problem diagnosis/fixing time. We used these to populate the error diagnosis and fixing time for our particular deployment process. Initial analysis shows that diagnosis time and undo/redo time are two key steps that are worth improving for overall dependability and time. We are currently listing a set of product and tool improvement requirements for shortening diagnosis time. We have also started working on an undo framework to support operators in cloud environment [7].

Second, we sought opinions from Hadoop ecosystem experts from industry using our process description. Interesting feedback that we are incorporating into our models include the careful pre/post-condition treatment on configuration resources during restart or redo, deployment processes being significantly different (i.e. high-availability requirements), choice of waiting or getting into exception handling for temporary failures and overall time optimisation using the models.

Third, we surveyed the literature for fault distribution and probability [8]. We summarised the data so that we can use them to inject faults into our process model for simulation. We are also using resource scheduling algorithms so the best choice among manual and automated deployment or rolling-upgrade schedules can be analysed with special consideration of faults. In conclusion, this is very much work in progress but our initial activities demonstrate that focusing on the process can give insight that results in reduction of the repair time for failures during operations activities.

REFERENCES