Chapter 9
Optimization Model for IT Change Management

Yixin Diao
IBM Thomas J. Watson Research Center, USA

Daniela Rosu
IBM Thomas J. Watson Research Center, USA

Leila Zia
Stanford University, USA

ABSTRACT
Today’s enterprises face the challenge of managing large, complex information technology (IT) systems consisting of software applications, servers, network routers, and other types of resources. Change management, and especially change scheduling, is known to be one of the most challenging problems in managing IT operations. Change scheduling must comply with strict availability Service Level Agreements (SLAs) and tackle complex IT environments. Various challenges arise from complex IT component dependencies, unpredictable durations of work processes, and large volume of concurrent changes and affected IT components. In this chapter, we address the problem of optimal IT change scheduling. We illustrate how the optimization problem can be formulated and solved using a mixed integer programming technique so that IT change management practitioners can efficiently schedule large change volumes and determine schedules that are robust to uncertainty in durations and costs with lower risks of SLA violations.

INTRODUCTION
IT Information Library (ITIL) defines change management as the process of controlling changes to the infrastructure and services, and of enabling the execution of approved changes with minimum disruption (ITIL, 2000). There are many challenges in efficient change management; examples include assessing change impact, estimating change cost, scheduling change requests, and managing change implementation. The focus of this chapter is on scheduling change requests, which is perceived as one of the most challenging problems in IT change management. In this scope, a change request indicates a request that

DOI: 10.4018/978-1-4666-8496-6.ch009
results in software, hardware, middleware, or process changes. Examples include requests for installing new software, increasing server memory, adding patches for databases, and removing or changing parts of an IT process. Thus, change scheduling is the problem of allocating these change requests to available change windows and practitioners, taking into account the various constraints that derive from service model, service level agreements (SLAs), business constraints and urgencies, as well as labor costs.

Multiple factors contribute to the complexity of IT change scheduling, including dependencies, time constraints, stochastic duration parameters, and volume. Dependencies arise when IT changes involve multiple tasks and/or multiple managed resources. Most often, a specific execution order is required across these tasks. For instance, the upgrade of database software has a pre-requisite in the operating system level. Therefore, the upgrade of the operating system must be executed before the upgrade of the database software. Similarly, the installation of a three-tier application environment often triggers changes on all of the nodes, and the change implementations on the outer tier must wait until related changes on the inner tier components are completed.

Another complexity factor is related to time constraints, which derive from both service agreements and business constraints. IT service agreements most often include negotiated change windows, which are periods in time when managed resources can be taken offline for application of changes. Typically, each change implementation must fit into a change window; otherwise, penalties are incurred as SLA violations. Business constraints may lead to requirements for urgent change execution in order to minimize business impact. For instance, the installation of a fix pack that solves a connectivity problem with mass impact for a critical sales application is an urgent change that must be implemented immediately in order to prevent business loss. Another example is a security-related patch, which must be implemented within a short period since its release in order to minimize the risk of security attacks. Energy-efficient cloud optimizations are a special case of urgent, automatically triggered IT changes with the purpose of reducing business costs.

Complexity rises from the stochastic nature of execution times for automated and manual tasks. Any realistic models for the change-scheduling problem need to address uncertainty in actual change execution times. However, most often, practitioners do not have a detailed quantification of the stochastic models, and can only provide educated guesses.

Finally, complexity arises from volume of concurrent change operations that must be coordinated across the IT environment. Events such as release of new security patches, or business decisions to upgrade OS or application software, may trigger a large volume of concurrent changes, even within a small IT environment. Change scheduling tools aim to tackle these four complexity factors while maximizing business value. Overall, the goal of IT Service Management tools is to integrate highly usable change scheduling components that can recommend optimal or close-to-optimal schedules. Usability derives from (1) how well the change models reflect the reality of change factors and costs, and (2) how efficiently practitioners or change management software can produce robust schedules, with control over the acceptable degree of change-parameter uncertainty.

Recently, there is increasing interest in the area of change management from both academia and industry. Keller et al. (2004) introduces CHAMPS, a system for automated change management that provides for change scheduling and deployment while accounting for precedence constrains and optimizing specific cost models. However, the scheduling solution uses heuristic techniques that typically are not strictly optimal. Sauve et al. (2006) and Reboucas et al. (2006) solve the change-scheduling problem by using a business driven approach that evaluates the impact of a change schedule in terms of the financial loss imposed on the service provider. However, the solution is obtained through exhaustive search, which is not scalable to complex change scenarios.