

Generating Project Plans for Data Center Transformations

Amrit Lal Ahuja and Girish Keshav Palshikar

Tata Research Development and Design Centre (TRDDC),
Tata Consultancy Services Limited,
54B, Hadapsar Industrial Estate, Pune 411013, India
{amrit.ahuja, gk.palshikar}@tcs.com

Abstract. Operations of modern organizations critically depend on Data Centers (DC). Due to ad hoc additions from diverse business units over time, the IT resources in a DC get unwieldy and complex. Transformations of DC - server consolidation, migration, application/data simplification, technology standardization - are important for cost, efficiency and reliability. Even when a specific transformation is identified (“consolidate these 100 existing servers into these 48 new servers”) it is difficult to generate a detailed optimal project plan for its execution. The project plan must identify all the tasks involved, identify an optimal team (size and expertise) and generate a detailed work schedule that meets the and respects the constraints and dependencies among the tasks. We present a methodology to generate such a plan automatically from given “high-level” IT transformation specifications (“as-is” and “to-be” states). We adopt a heuristic forward chaining metric temporal planner engine (SAPA) to generate a project plan that attempts to optimize the overall time and team-size. The idea is to capture the domain-knowledge as reusable planning action. This automation reduces the efforts and errors in manual project planning. The method can be extended to projects in other domains.

1 Introduction

Operations of modern organizations critically depend on *Data Centers* (DC), which consist of a large number of IT resources such as servers, storage, networks, OS, middleware, databases, business applications etc. Due to changing business requirements, changing workload patterns, new applications/users/customers/business partners and technology changes, the IT infrastructure in a DC keeps evolving. Due to *ad hoc* additions from diverse business units over time, the IT resources in a DC get unwieldy and complex. As a real-life example, a large bank has several DCs containing 30,000+ servers, 5+ OS with 20+ versions, 10+ database products, 25+ middleware environments, 2500+ app server instances, 10+ petabytes of storage, 35+ programming languages etc. As another example, the US Army has 800+ DCs which run 24,000+ servers and 9000+ applications, which they are planning to consolidate into 615+ DC by 2014, thereby hoping to save approx. \$5 billion (<http://www.army.mil/article/70991/> accessed on 04-July-2012).

Managing a DC consisting of such large, complex, heterogeneous and business critical IT resources is very difficult, leading to issues with performance, utilization, reliability, availability, costs (maintenance, license and power) and workload on support personnel. Thus DC transformations - e.g., server consolidations, application and data simplifications, technology standardizations - are important for cost, efficiency and reliability. Focus on green IT is another reason for DC transformations. DC transformations is an active area of research; e.g., [1,2,3,4,5,6] and much other work, most of which is primarily about problems and technologies related to DC transformations, including in cloud and other scenarios. Many IT organizations provide specialized services and products for DC transformations.

In this paper, we focus on two problems: (i) generating a detailed and reasonably good (not necessarily optimal) project plan for a given DC transformation project; and (ii) deciding the optimal team size for the project. Creating such a project plan is a complex and effort-intensive activity, as myriad details of identifying relevant tasks, interdependencies among IT resources as well as tasks, human resource related constraints etc. need to be taken into account. Manually creating such a plan is error-prone and time-consuming; ensuring that the plans are close to optimal is difficult.

In this paper, we propose an approach where detailed and reasonably good quality project plans for DC transformation projects are automatically generated using a standard temporal planner engine. The high-level details of the IT resources and their transformations are taken as input. The approach is based on a one-time activity of building a comprehensive library of transformation tasks, modeled as a set of actions (operators) in the domain file in PDDL. All the dependencies for each task as well as team-related constraints are built into the corresponding action. The transformation details are automatically translated to the problem file in PDDL. The planner engine then generates a location-wise plan (since a transformation project often includes multiple DCs) that includes start and end time for each task, along with the associated team member to carry it out. As far as we know, this is the first time a temporal planner is used to generate detailed project plans for DC transformation projects. The approach is extendible to projects of other kinds.

The paper is organized as follows. Section 2 defines the DC transformation task in detail along with a real-life case-study. Section 3 defines the approach and illustrates how it works for the case-study. Section 4 describes related work. Section 5 offers some conclusions and outlines future work.

2 DC Transformation Projects

2.1 Server Consolidation and DC Migration

We focus here on a particular kind of DC transformation strategy, called *server consolidation*. The task is to identify redundant, under-utilized, “small” or “old” servers in a DC and move all the data and applications on them to other “suitable” existing servers and/or on new servers, so that the new set of servers are “better” than the old set of servers (e.g., less power, less operating costs, better reliability, easier to manage etc.). Server consolidation is often coupled with *DC migration*, where all IT resources in a *source DC* are “consolidated” into resources in another *target DC* and the source DC is

then closed, thereby saving on some capital costs. A server consolidation specification consists of one or more movegroups. A *movegroup* (MG) [1] often consists of a group of closely related servers along with related resources such as business applications (e.g., cheque processing or ATM transaction processing in a bank), databases etc. There are carefully designed strategies to form the MGs in a server consolidation project.

The MGs are disjoint i.e., any server in a source DC belongs to at most one MG. Each MG G is a unit of server consolidation project and consists of a finite set $S = \{s_1, s_2, \dots\}$ of *source servers* from DC D , which need to be mapped or moved to a set T of *destination* (or *target*) servers in (possibly different) DC D' . We assume that $S \cap T = \emptyset$. Generally, $|T| \leq |S|$ i.e., target servers are fewer in number than source servers. For generality, we assume that source servers in all MG belong to the same DC D and target servers in all MG belong to the same DC $D' \neq D$. *MG specification* gives complete details of each source as well as target server in it; e.g., number of CPUs, memory, disk, OS, special-purpose HW devices connected to it, system software running on it, business applications running on it, old and new IP addresses etc. The MG specification also defines which source servers are mapped to which target servers and also the method of transformation (Section 2.2).

The MGs in a project may have dependencies among each other and so they need to be consolidated in a suitable order that “respects” these dependencies. However, we will ignore this issue and assume that we are given an MG which either has no dependencies on other MGs or all the MGs on which it is dependent are already consolidated in the target DC. This is not a serious limitation because our approach is designed to handle such dependencies.

If all the source servers in an MG were independent, then in principle they can all be consolidated in sequence, in parallel or in any combination of sequential and parallel steps. However, the source servers (and the applications running on them) within an MG may depend on each other in several ways:

- a source server may provide a shared service to other servers;
- a source server may host data (e.g., tables or files) shared by other servers;
- a source server may control access to a hardware resource (e.g., tape drive or communication link) by other servers.

Thus the dependencies among the source servers must be taken into account when preparing a project plan for an MG. In general, a server may run several business applications and there may be dependencies among these applications as well.

2.2 A Case-Study

We consider a real-life server consolidation and DC migration project, consisting of nearly 400 servers in source DC called DC1, which were to be consolidated and migrated to about 300 servers in target DC called DC2. This project was divided into 7 MGs. Figure 1 shows the specification for a particular MG consisting of 19 source servers (all in one DC) and 10 target servers (all in another DC). We have omitted some details such as source and target IP addresses etc. Servers marked as “shared” typically run common services or functionality (e.g., databases, user authentication etc.)

Source Server	Business Application Group	Server Function	CPU	Memory	Storage GB	SAW Boot	Target Server	Target State
1	Shared	App	1	8192	864.61	Yes	1001	LPAR
2	FRNA	App	2	767	20	Yes	1002	x86 V2V
3	FRNA	App	2	767	20	Yes	1002	x86 V2V
4	FRNA	App	2	2047	30	Yes	1002	x86 V2V
5	FRNA	App	2	1051	24.96	Yes	1002	x86 V2V
6	FRNA	App	4	3967	37.25	Yes	1003	new build on x86 P2P-M
7	FRNA	App	2	767	20	Yes	1003	x86 V2V
8	FRNA	App	2	767	20	Yes	1003	x86 V2V
9	FRNA	App	2	8191	33	Yes	1003	x86 P2V
10	Shared	Database	2	4096	157.24	Yes	1005	x86 P2V
11	FRNA	Database	2	3967	109.99	Yes	1006	new build on x86 P2P-S
12	Shared	Database	0.5	6144	615.25	Yes	1007	LPAR
13	Shared	Database	0.5	6144	598.03	Yes	1007	LPAR
14	GRPD_MFGPRO	MP	1	1023	8.43	Yes	1008	x86 P2V
15	FRNA	App	2	3967	50	Yes	1009	x86 P2V
16	GRPD_MFGPRO	App	2	2399	70	Yes	1008	x86 V2V
17	FRNA	Web	1	1563	20	Yes	1010	x86 V2V
18	STD	Web	2	3967	20	Yes	1011	x86 P2V
19	FRNA	Web	1	1563	20	Yes	1010	x86 V2V

Fig. 1. Specification of an MG

used by multiple applications. Here, LPAR indicates the process of migrating everything (OS, applications, data etc.) from a physical source server (IBM machine running AIX) to a virtual machine, which consists of a logical partition of hardware resources (e.g., CPUs, memory) on a physical target server from IBM running AIX. P2V indicates the process of migrating everything (OS, applications, data etc.) from a physical source server (Intel-based machine running Windows) to a virtual machine, typically emulating the same environment using virtualization software such as VMWare. Other transformations can be similarly understood. We have assumed that shared servers within this MG need to be consolidated first and the business application groups (FRNA, STD, GRPD_MFGPRO) can be consolidated in any order. Within FRNA, the application servers can be consolidated in any order and the database server must be consolidated last; the system allows more complex precedence relations to be set among the servers in a business application group.

2.3 The Processes of Server Consolidation and DC Migration

Basic steps in server consolidation are about moving all the OS, database, middleware, application programs and other IT resources from a particular source server onto its target server. These steps actually consists of several *tasks*, which depend on the specifics of the concerned server and the IT resources on it. Each task is executed either on the source server or on the destination server. Each task requires a particular skill and which generally takes a specific amount of time. In addition, the tasks themselves have dependencies among each other, which need to be respected when preparing a project plan. For example, suppose task *A* takes the backup of the data on source server, task *B* creates the necessary tables, users and permissions on the destination server and task *C* restores this data on the target the server; clearly task *A* and *B* must be done before task *C*. In general, the tasks may be structured hierarchically i.e., a task may have sub-tasks and so forth. Even the time required for each task should ideally be treated as a random

variable. However, for simplicity, we treat each task as an atomic unit which requires a specific skill and takes a fixed amount of time.

Any server consolidation and DC migration project typically involves some common tasks. The tasks can be divided into the following groups, where tasks within a group are functionally similar.

1. **Source Server Handover:** These tasks involve a formal handing over of a source server from the IT operations team to the server consolidation team.
2. **Source Server Pre-migration:** These tasks are related to system health checkup, installation checks and other tasks that need to be done before this server's consolidation can begin.
3. **Server image:** These tasks typically consist of creating an image (or a snapshot of the state) of the entire source server on a storage device, transferring this image to the target DC and copying the image onto the target server. Assuming that the target server has already been configured to have the system environment as the source server, the target server is then booted up and it starts in the same state as where the source server was stopped. Detailed specifications of these tasks are dependent on the type of the source and target servers. For example, AIX provides `mksysb` and related commands for non-SAN bootable servers.
4. **Data Backup:** This involves taking a backup of all the user files, tables etc. Detailed specifications of the tasks are dependent on the type of the source servers as well as the device on which the backup is taken (e.g., tape, USB, CD etc.). Also, the backup can be total or only incremental. Incremental backup is required if the source server is functioning during the server consolidation project.
5. **Data Transfer:** These tasks involve either physically shipping a storage device to the target DC or transferring the backed up data over a communication network, typically using `ftp` commands.
6. **Server Shutdown:** These tasks are related to shutting down a (source) server.
7. **Target Server Startup and Data Restore:** These tasks are related to starting up a source server, creating the same system environment on it (using virtualization if required) and ensuring that the required configuration parameters, environment variables, users, access permissions, devices etc. are correctly set. These tasks also include using a storage device to restore the backed up data from the source server into the target server and ensuring its completeness and correctness. The restored data may be part of a full or incremental backup. The business applications on the target are then started one by one and brought up "alive".
8. **Target Server Go-Live and Handover:** These tasks involve user acceptance testing of business applications in a live environment and a formal handing over of a source server from the server consolidation team to the IT operations team.

Detailed specifications of the tasks in each group are dependent on the type of the source server. The tasks have well-understood precedence relationships, which can be modeled as precedence graph or equivalently, a partial order. There are precedence relations even among the tasks in the same task group; thus the precedence graph is actually over all pairs of tasks and servers. However, in most cases, these dependencies are only for tasks to be performed on the same server; tasks on different servers are usually unrelated or unconstrained (i.e., there is parallelism among tasks across servers).

2.4 Project Plan

We focus here on creating a project plan for a particular MG. The approach can be extended easily to create a project plan for the entire server consolidation project (i.e., all the MG together). A project plan for an MG lists out all the tasks involved for consolidating each source server onto the destination server, along with start and end time for each task and assigns a specific person to carry out each task. Clearly, the requirement is to prepare a project plan that respects all server dependencies as well as task dependencies (which corresponds to the required domain knowledge), minimizes the overall time as well as minimizes the number of people required to execute the tasks. Additional people-related constraints need also to be taken into account while preparing the project plan; e.g., a person cannot work more than, say, 9 hours in a day and a person can work on exactly one task at any time (no “multiplexing” of tasks).

Clearly, creating such a detailed project plan manually (even for a single MG), is difficult, time-consuming and error-prone, particularly when the MG includes a large number of servers. It is difficult to ensure that the manually created plan is optimal in terms of delivery time and team size. Lastly, manually re-generating the project plan at some later stage during execution is even more difficult.

3 Using a Planner Engine to Generate Project Plans

3.1 The Approach

We propose an approach where a project plan for an MG is automatically generated by a planner engine from the given MG specifications. We maintain the required domain knowledge about the tasks and their dependencies, input pre-conditions and output effects (post-conditions), as a library of standard actions in the planner engine. Creating such a library is a one-time task, which is reused for generating project plans for multiple MGs as well as multiple server consolidation projects. This approach makes it easy to quickly generate optimal (or at least close to optimal) project plans. The steps in the approach are as follows:

1. Create a reusable library of tasks involved in server consolidation and DC migration as a domain file in PDDL; this is a one-time manual step.
2. Automatically generate a PDDL problem file from MG specifications
3. Automatically generate a detailed project plan from given problem file and the domain file, using a standard planner engine.
4. Automatically generate a Gantt chart and other project plan deliverables from the output plan.

3.2 Modeling Tasks as a Domain File in PDDL

Detailed specifications of these tasks specialized for each type of servers, including their precedence tasks and post facto effects, the skills required and expected duration for each task constitutes a reusable component of domain knowledge. We model each task as a durative action (or operation) in the planning domain and specify it using Planning Domain Definition Language (PDDL) 2.1 [7]. Briefly, a PDDL specification consists of:

1. *Objects*: in our case, these are the various IT resources such as servers, devices, operating system, system software, databases, middleware and business applications.
2. *Predicates*: Boolean properties of objects which can be TRUE or FALSE.
3. *Initial state*: in our case, these specify the initial conditions about the source and target servers.
4. *Goal state specification*: in our case, these specify the final conditions about the target servers.
5. *Actions/operators*: ways in which the objects can be manipulated to reach a goal state from the given initial state. Each action has a set of *parameters*, a *precondition* (which must be TRUE for the action to get executed and a *post-condition* (or *effect*, which is guaranteed to be TRUE after the action is executed. In our case, actions correspond to tasks. We encode the precedence relations for a task as part of the pre-conditions for the task. We use durative action facility in PDDL and associate a time duration with it; this means that we can only use temporal planner engines to generate the project plan.

In PDDL, the predicates and actions (operators) are defined in a *domain file* and objects, initial state and goal state specifications (for a specific instance of IT transformation) are defined in a *problem file*. This library (i.e., the domain file) of PDDL predicates and actions is reusable for generating project plans for multiple MGs and different server consolidation projects (i.e., for different problem files). If the given MG includes an unknown IT resource (e.g., a server with a previously unseen operating system) then we need to manually add the corresponding tasks to the domain file, including any changes to the precedence relations for all tasks.

Figure 2(a) shows PDDL specifications of one task. `SERVER`, `BA`, `LOCATION` and `TEAM` are (sets of) objects. Here, `BA` stands for a set of business applications, `TEAM` stands for a set of human beings involved in carrying out the tasks, `LOCATION` contains locations of the source and target DC. These objects have various Boolean properties, which are specified as predicates; e.g., each server has a type (predicate `serverTypeIs`) and a location (predicate `at`), each server contains one or more business applications (predicate `partOfBA`), each team member has a location (predicate `available-at`) and can be free or busy (predicate `free`) and so forth. This task depends on the shutdown task for the given server, which sets the flag `shutdown-done` for the given server. Note that a precondition for this action checks that the given team members is free to perform this task. The flag `virtualReplica-done-at` is set for the given server and location as a post-condition of this task, which the later tasks can check.

3.3 Modeling MG Specification as Problem File in PDDL

The actual MG is specified as a problem file in PDDL. The problem file contains a set of objects, the initial conditions and a specification of the goal state. In our case, the initial conditions correspond to the information about the source servers (e.g., their location is the old DC). The goal state specification corresponds to the facts that all source servers are moved to the target DC and are “working satisfactorily” i.e., all the tasks have been performed and the user acceptance testing is passed for each target server. We generate the problem file automatically from the MG specification (Figure 2(b)).

```

(:durative-action NetApp6080enapMirrorReplica_P2V
:parameters (
  ?srvr - SERVER
  ?ba - BA
  ?loc1 - LOCATION
  ?tm - TEAM
  ?loc2 - LOCATION
)
:duration (= ?duration 4.0)
:condition (and
  (over all (at ?srvr ?loc1))
  (over all (partOfBA ?srvr ?ba))
  (over all (serverTypeIs ?srvr Windows))
  (over all (toBeMigratedTo ?srvr ?loc2))
  (over all (toBeMigratedAs ?srvr P2V))
  (over all (notTakeFromISONet ?srvr))
  (at start (shutdown-done ?srvr))
  (over all (does-replica ?tm))
  (over all (available-at ?tm ?loc2))
  (at start (free ?tm))
)
:effect (and
  (at start (not (free ?tm)))
  (at end (free ?tm))
  (at end (not (shutdown-done ?srvr)))
  (at end (virtualReplica-done-at ?srvr ?loc2))
)
)

(define (problem server-consolidation-DC-migration)
(:domain itInfraTrans)
(:objects
  SharedServers STD GRPD_MFGPRO FRNA - BA
  S1, S2, ..., S9, ..., S19 - SERVER
  DC1 DC2 - LOCATION
  Legacy P2P V2V P2V LPAR - TARGET
  Windows - SERVERTYPE
  WEB DBA APP MFS - ITAPP
)
(:init
  ...
  (nameOfBA FRNA)
  (baCurrentlyAt FRNA DC1)
  (at S9 SVDC)
  (toBeMigratedTo S9 DC2)
  (partOfBA S9 FRNA)
  (toBeMigratedAs S9 P2V)
  ...
)
(:goal (and
  (migrationDoneTo S9 DC2)
  (UAT-done FRNA)
  ...
)
)

```

Fig. 2. (a) PDDL specification for one task. (b) Problem file for MG specification in Figure 1.

3.4 Finding Optimal Team Size

In our formulation of the multi-agent planning problem, the optimal number of agents is unknown and the planner is expected to generate a plan that is not only optimal in terms of time but also in terms of number of agents required to carry out the plan. We are working to reformulate the planning problem accordingly. In the meantime, we take a simple approach: we run the planner several times, each time varying the team size and checking the overall turnaround time for the project. We then select the optimal team size as one that delivers the project in the shortest time. For the example MG, we have clubbed handover and pre-migration tasks into a single skill-group, shutdown tasks constitute a separate skill-group, data backup, data transfer, data restore and system image as two separate skill-groups (for Microsoft Windows and for AIX), and finally, handover and go-live tasks as a separate skill-group. For brevity, we report only the results where we have kept the team-size for 4 groups as fixed and only vary the team-size for the skill-group for Windows. Figure 3 shows how the project turn-around time varies with team-size for this skill-group; clearly, the team-size of 14 is optimal for this skill-group. The overall team-size as per this optimal plan is 21 people and 5 skill-groups, with the following skill-wise and location-wise break-up: handover and pre-migration in DC1:1, shutdown in DC1:2, AIX in DC2:2, Windows in DC2:14, go-live and final handover (user acceptance testing) in DC2:2.

3.5 Project Plan for the Example MG

We use the metric temporal planner engine SAPA [8] to generate the plan using the domain file (which is fixed) and the given problem file, which is specific to a particular MG. The generated plan for the example MG contains a server and task-wise schedule that spans 40 hours, which is well within the given upper limit of 48 hours (a weekend). The composition of the optimal team is already discussed. Our system also has facilities to convert the generated project plan into user-readable artifacts such as schedules (Gantt charts), team utilization summary and other auxiliary outputs which are useful

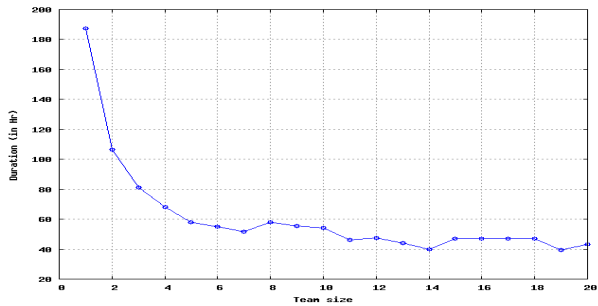


Fig. 3. Project turn-around time as a function of the team-size, for the example MG

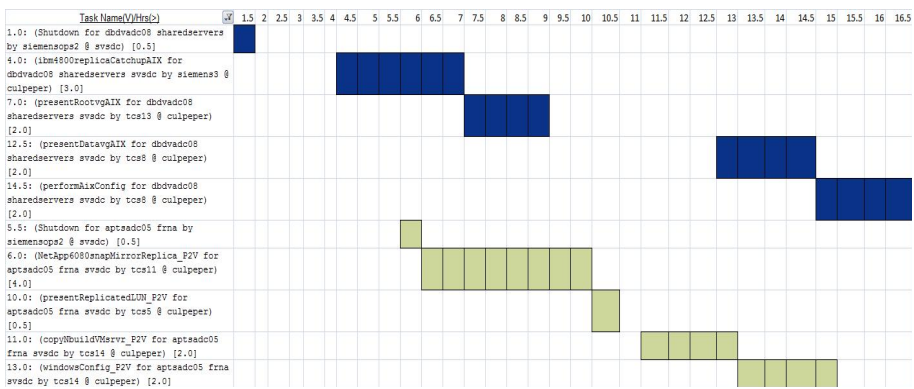


Fig. 4. Part of the Gantt chart produced for the example MG

for understanding the plan and during plan execution. Figure 4 shows a part of the Gantt chart for servers 9 and 12 in this MG. Even though 12 is a shared server, its consolidation overlaps in time with that of the server 9 in application FRNA. This is fine because there is no direct dependency between the two and the entire FRNA application does not go “live” until all the shared servers are already consolidated. This part of the Gantt chart is not shown.

4 Related Work

Automated planning plays a significant role in a variety of demanding applications, ranging from controlling space vehicles and robots to playing bridge [9]. As in our work, planners have been used for generating workflows for IT infrastructure management which transform a given initial state to a desired final state and where the transformation actions are encoded as planning actions. [10] used a partial order planner for this purpose. [11] generates workflows for IT infrastructure reconfiguration projects, where each step in the workflow preserves global constraints, which are specified by the end-users “outside” of any particular actions. [12] develop propositional planners

which use domain-specific heuristics to generate plans for the closely related problem of incremental transformations of a network topology so as to minimize service disruptions. In these papers, the actions are not durative and team size optimization is not considered. Planners are also used for somewhat similar problems of services software composition in business process management, which we do not review here; see [13].

A closely related planning problem formulation is called *multiagent planning*, in which the planner needs to create a plan taking into account concurrent actions of and dependencies of multiple autonomous agents, such as robots. Several approaches have been designed to create multi-agent plans; see, for example, [14,15,16,17,18,19,20,21,22] etc. The issues there are related to the nature of the planning process: dynamic or static, centralized or distributed, agents are truthful or not, nature of cooperation among agents etc. The focus is on coordination and control of agent actions. In this paper, we have considered a centralized, static project planning formulation, in the classical planning framework, where the tasks and their precedences are known and the time required for each task is also known. We have associated a specific skill with each task. The key difference is that the number of people required to execute the plan is not fixed and known; rather, we want to find the right plan along with the right team size. There are probabilistic planning approaches where quantities like task times can be treated as random variables to generate “most probable” plans; e.g., [17,23].

5 Conclusions and Further Work

In this paper, we use a standard temporal planner engine to automatically generate a detailed project plan for DC transformation projects, which takes into account the needed tasks and skills as well as various kinds of dependencies. The approach is based on one-time creation of a reusable library of DC transformation tasks in PDDL. We illustrated the approach using a real-life case-study.

A limitation of the current approach is scale: SAPA is unable to handle large projects involving 100s of servers. We currently do not take into account some precedence and priority constraints; e.g., FRNA and STD are treated as equal, without any precedence among them, except that shared servers are consolidated before both business applications. We do not take into account precedence among specific programs running on a server. The system is not currently usable to generate a partial plan, after some of the tasks are done and perhaps they have overshot their allocated time. End-users need a lot more information, summarized from the generated plan, such as utilization factors, workload measures etc. We are working on adding these features. A key aspect of our work is manually transforming the domain knowledge (for IT infrastructure transformations) into PDDL. For a planning problem with a large number of inter-related tasks and objects, this becomes complex and error prone; approaches like HTN to PDDL translation [24] can simplify this process and approaches like PDVer [25] can be used to verify the planning domain description. Planners which use PDDL3 [26] (with additions such as sometime, at-most-once etc.) can be exploited to make the generated plan more realistic. we need to generalize our approach by treating each task duration as a random variable (not as a constant) and then generate “most probable” plans. We are working on reformulating our planning problem to include team sizing and utilization as an integral part. A key aspect of project planning is identifying the risk factors, quantifying

risks and creating a project plan that minimizes risks (e.g., by including risk mitigating actions), as well as suggests alternative workflows in case a risk materializes. We are working on including these aspects in the planning problem formulation.

Acknowledgements. We thank Prof. Harrick Vin for supporting this work, Rahul Kelkar, Anjali Gajendragadkar and Manish Sarda for their domain knowledge and Pre-rna and Arpit for working on the mPlan tool. We thank the reviewers for their insightful comments.

References

1. Gajendragadkar, A., Bhardwaj, R., Sarda, M., Malavade, C.: A systematic approach for data center migration analytics driven methodology. In: TACTiCS, TCS Technical Architects Conference 2010 (2010)
2. Ramakrishnan, K.K., Shenoy, P., Van der Merwe, J.: Live data center migration across wans: a robust cooperative context aware approach. In: Proceedings of the 2007 SIGCOMM Workshop on Internet Network Management, INM 2007, pp. 262–267. ACM (2007)
3. Hacking, S., Hudzia, B.: Improving the live migration process of large enterprise applications. In: Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing, VTDC 2009, pp. 51–58. ACM (2009)
4. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, NSDI 2005, vol. 2, pp. 273–286. USENIX Association (2005)
5. Nelson, M., Lim, B.H., Hutchins, G.: Fast transparent migration for virtual machines. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC 2005, pp. 25–25. USENIX Association (2005)
6. Bradford, R., Kotsovinos, E., Feldmann, A., Schiöberg, H.: Live wide-area migration of virtual machines including local persistent state. In: Proceedings of the 3rd International Conference on Virtual Execution Environments, VEE 2007, pp. 169–179. ACM (2007)
7. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 61–124 (2003)
8. Do, M., Kambhampati, S.: Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20, 155–194 (2003)
9. Ghallab, M., Nau, D., Traverso, P.: Automated planning theory and practice. Morgan Kaufmann Publishers (2006)
10. El Maghraoui, K., Meghranjani, A., Eilam, T., Kalantar, M., Konstantinou, A.V.: Model Driven Provisioning: Bridging the Gap Between Declarative Object Models and Procedural Provisioning Tools. In: van Steen, M., Henning, M. (eds.) *Middleware 2006*. LNCS, vol. 4290, pp. 404–423. Springer, Heidelberg (2006)
11. Herry, H., Anderson, P.: Planning with global constraints for computing infrastructure reconfiguration. In: Proceedings of the 2012 AAAI Workshop on Problem Solving Using Classical Planners. AAAI Press (2012)
12. Yoon, Y., Robinson, N., Muthusamy, V., Jacobsen, H.A., McIlraith, S.A.: Planning the transformation of network topologies. In: Proceedings of the 2012 AAAI Workshop on Problem Solving Using Classical Planners. AAAI Press (2012)
13. Hoffmann, J., Weber, I., Kraft, F.M.: Sap speaks pddl: Exploiting a software-engineering model for planning in business process management. *Journal of Artificial Intelligence Research* 44, 587–632 (2012)

14. Jonsson, A., Rovatsos, M.: Scaling up multiagent planning: A best-response approach. In: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011), pp. 114–121. AAAI (2011)
15. Kvarnstrom, J.: Planning for loosely coupled agents using partial order forward-chaining. In: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011), pp. 138–145 (2011)
16. Nissim, R., Brafman, R.I., Domshlak, C.: A general, fully distributed multi-agent planning algorithm. In: Proceedings AAMAS 2010, pp. 1323–1330 (2010)
17. Beaudry, E., Kabanza, F., Michaud, F.: Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. In: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010), pp. 10–17 (2010)
18. Larbi, R.B., Konieczny, S., Marquis, P.: Extending Classical Planning to the Multi-agent Case: A Game-Theoretic Approach. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 731–742. Springer, Heidelberg (2007)
19. Ephrati, E., Rosenschein, J.S.: Multi-agent planning as search for a consensus that maximizes social welfare. In: Castelfranchi, C., Werner, E. (eds.) MAAMAW 1992. LNCS, vol. 830, pp. 207–226. Springer, Heidelberg (1994)
20. Dimopoulos, Y., Moraitis, P.: Multi-agent coordination and cooperation through classical planning. In: Proceedings IAT 2006, pp. 398–402 (2006)
21. Cox, J.S., Durfee, E.H.: An efficient algorithm for multiagent plan coordination. In: Proceedings AAMAS 2005, pp. 828–835 (2005)
22. Bacchus, F., Ady, M.: Planning with resources and concurrency: a forward chaining approach. In: Proc. International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 417–424 (2001)
23. Mausam, Weld, D.: Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research* 31, 33–82 (2008)
24. Alford, R., Kuter, U., Nau, D.: Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In: Proceedings of IJCAI 2009, pp. 1629–1634 (2009)
25. Raimondi, F., Pecheur, C., Brat, G.: PDVer, a tool to verify pddl planning domains. In: Proceedings of ICAPS 2009 Workshop on Verification and Validation of Planning and Scheduling Systems, Thessaloniki, Greece (2009)
26. Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6), 619–668 (2009)