

Leibniz Universität Hannover  
Wirtschaftswissenschaftliche Fakultät  
Institut für Produktionswirtschaft  
Prof. Dr. Stefan Helber

Hausarbeit im Rahmen der Veranstaltung  
Entwicklung von Anwendungssystemen im WiSe 2014/2015  
(Veranstaltungs-Nr. 173610)

RCPSP  
RCPSP

Andreas Hipp	Robert Matern
Ungerstr. 24	Plathnerstr. 49
30451 Hannover	30175 Hannover
Matr.-Nr. 3027520 ???	Matr.-Nr. 2798160

Abgabedatum: 24.03.2015

# Inhaltsverzeichnis

Abbildungsverzeichnis	iii
Tabellenverzeichnis	iii
Abkürzungsverzeichnis	iv
Symbolverzeichnis	v
<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen zur ressourcen-beschränkten Projektplanung und zu dem Framework Ruby on Rails</b>	<b>2</b>
2.1 Kapazitätsplanung . . . . .	2
2.2 Kostenplanung . . . . .	4
2.3 Ruby on Rails . . . . .	6
<b>3 Implementierung des RCPSP mittels Ruby on Rails</b>	<b>7</b>
<b>4 Kritische Würdigung des Anwendungssystems</b>	<b>7</b>
<b>5 Fazit</b>	<b>7</b>
<b>Literatur</b>	<b>8</b>
<b>A Anhang</b>	<b>10</b>
A.1 GAMS-Implementierung des Beispiels . . . . .	10
A.2 Ruby on Rails Programmcode . . . . .	10

## Abbildungsverzeichnis

1	Terminalfenster unter Apple Mac OS X . . . . .	6
---	--	---

## Tabellenverzeichnis

## Quellcodeverzeichnis

1	Beispielcode . . . . .	7
---	------------------------	---

# Abkürzungsverzeichnis

RCPS	Resource-Constrained Project Scheduling
RoR	Ruby on Rails
SGS	Schedule Generation Scheme

# Symbolverzeichnis

$d_i$	Dauer von Vorgang $i$
$FE_i$	frühestes Ende von Vorgang $i$
$i, h = 1, \dots, I$	Vorgänge
$k_{ir}$	Kapazitätsbedarf von Vorgang $i$ auf Ressource $r$
$kp_r$	verfügbare Kapazität von Ressource $r$ je Periode
$\mathcal{N}_i$	Menge der direkten Nachfolger von Vorgang $i$
$oc_r$	Kosten einer Einheit Zusatzkapazität von Ressource $r$
$O_{rt}$	Zusatzkapazität von Ressource $r$ in Periode $t$
$r = 1, \dots, R$	Ressourcen
$SE_i$	spätestes Ende von Vorgang $i$
$t, \tau = 1, \dots, T$	Perioden
$\mathcal{V}_i$	Menge der direkten Vorgänger von Vorgang $i$
$X_{jt} \in \{0, 1\}$	gleich 1, falls Vorgang $j$ in Periode $t$ endet, sonst 0

# 1 Einleitung

Bei einem Projekt handelt es sich um eine zeitlich befristete, relativ innovative und risikobehaftete Aufgabe von erheblicher Komplexität, die meist einer gesonderten Planung bedarf.<sup>1</sup> Dementsprechend von großer Bedeutung ist die vorhergehende und genaue Planung von Projekten.<sup>2</sup> Projektplanung ist die Planung aller Arbeitsgänge eines Projekts durch Zuweisung eines Startzeitpunktes, so dass die Zeitbeziehung zwischen den Vorgängen eingehalten und knappe Ressourcenkapazitäten nicht überschritten werden.<sup>2</sup> Durch das Zerlegen des Projekts in einzelne Arbeitsgänge wird versucht die Komplexität zu reduzieren und eine geordnete Abfolge der Arbeitsgänge zu erstellen, um das Projektziel zu erreichen.<sup>3</sup> Projektziele können dabei unterschiedlich kategorisiert werden, z. B. in Sach-, Termin- oder Kostenziele.<sup>4</sup>

Nach DIN 69900 hat ein Arbeitsgang oder ein einzelner Vorgang eines Projekts einen definierten Anfang sowie ein definiertes Ende und dient für das Projekt als Ablauelement zur Beschreibung eines bestimmten Geschehens.<sup>5</sup> Trotz der Zerlegung besitzen die einzelnen Arbeitsgänge des Projekts eine Beziehung, mit der die Reihenfolge der Ablauffolge bestimmbar ist.<sup>6</sup> Oft wird zur Darstellung der Vorgangsbeziehung ein Vorgangsknoten-Netzplan verwendet.<sup>7</sup> Ein Arbeitsgang ist i. d. R. verbunden mit dem Einsatz von Ressourcen, welche wiederum mit Kosten verbunden sind. Eine Möglichkeit, das Projektziel unter minimaler Ressourcenverwendung zu erreichen, ist die effiziente Planung der Ablauffolge der Arbeitsgänge eines Projekts.<sup>8</sup> Damit ist es möglich, mehrere Projekte bei einer gegebenen Zeitvorgabe unter Einhaltung von Ressourcenrestriktionen fertigzustellen bzw. bei konstanter Ressourcenkapazität ein Projekt in kürzerer Zeit abzuschließen.

Zur Bestimmung der optimalen Ablauffolge der einzelnen Arbeitsgänge eines Projekts kann ein Optimierungsmodell verwendet werden, bei der für eine festgelegten Ablauffolge eines Projekts und unter Berücksichtigung der Ressourcenbeschränkung die Fertigstellungszeit minimiert wird. Im Kapitel 2 wird eine solche Modellformulierung für das ressourcenbeschränkte Projektplanungsproblem als sogenannte Kapazitätsplanung vorgestellt.<sup>9</sup> Alternativ wird in dem Kapitel das Optimierungsmodell um die Bedienung erweitert, dass Zusatzkapazitätseinheiten gebucht werden können. Mit dieser Modellerweiterung wird von der Kostenplanung in Projekten gesprochen. Bezeichnet wird im Allgemeinen das ressourcenbeschränkte Projektplanungsproblem mit der englischen Bezeichnung des *Resource-Constrained*

---

<sup>1</sup>Vgl. Voigt und Schewe (2014)

<sup>2</sup>Vgl. Zimmermann et al. (2006), S. VI

<sup>3</sup>Vgl. Zimmermann et al. (2006), S. 4

<sup>4</sup>Vgl. Felkai und Beiderwieden (2011), S. 52

<sup>5</sup>Vgl. DIN 69900 (2009), S. 15

<sup>6</sup>Vgl. Kellenbrink (2014), S. 6-7

<sup>7</sup>?????

<sup>8</sup>Vgl. Bartels (2009), S. 11-12

<sup>9</sup>????

*Project Scheduling Problem (RCPSP)*. Bei dem RCPSP handelt es sich um eine abstrakte mathematische Modellformulierung. Ziel der vorliegenden Arbeit ist es das RCPSP in Ruby on Rails (RoR) zu implementieren. Bei RoR handelt es sich um ein Framework zur Entwicklung von Webdokumenten bzw. Internetseiten.<sup>10</sup> Es baut auf der Programmiersprache Ruby auf und ist ursprünglich von David Heinemeier Hansson entwickelt.<sup>11</sup> Die Implementierung bedarf einer Verknüpfung von RoR und GAMS<sup>12</sup>. Unter GAMS wird eine algebraische Modellierungssprache für mathematische Optimierungsprobleme verstanden, mit der das RCPSP gelöst wird.<sup>13</sup> Im Kapitel 3 wird die Entwicklung des Anwendungssystems zum Lösen des RCPSP ausführlich beschrieben. Ergänzt wird diese Arbeit durch eine kritische Würdigung des Anwendungssystems in Kapitel 4 sowie einem Fazit in Kapitel 5.

## 2 Grundlagen zur ressourcen-beschränkten Projektplanung und zu dem Framework Ruby on Rails

### 2.1 Kapazitätsplanung

Ein Großteil an Projekten besitzt die Eigenschaft eines beschränkten Ressourcenkontingents.<sup>14</sup> Soll demgemäß die vorgegebene Terminierung des Projektes als zuvor festgesetztes Ziel erreicht werden, muss neben der Reihenfolgerestriktion auch der Ressourcenbedarf der unterschiedlichen Arbeitsgänge sichergestellt werden. Mit der Einhaltung des Ressourcenbedarfs ist es möglich, alle zur Erfüllung des Projektes notwendigen Arbeitsgänge auszuführen und somit letztendlich das Projekt abzuschließen. Neben limitierten Ressourcen, die während des gesamten Projekts nur ein Mal zur Verfügung stehen, wie bspw. das Projektbudget, gibt es Ressourcen, die nach einer bestimmten Anzahl von Perioden erneuert werden können.<sup>15</sup> Erneuerbare Ressourcen sind bspw. die Produktionskapazität einer Maschine oder der Personaleinsatz für ein Projekt. In dieser Arbeit wird der Fokus auf diese erneuerbaren Ressourcen gelegt.

Zur Lösung des ressourcenbeschränkten Projektplanungsproblems kann das Modell RCPSP genutzt werden. Das RCPSP legt durch Fixierung der Aktivitätsstartzeitpunkte den Projektgrundablauf zur Zielerreichung der Minimierung der Projektdauer fest. Dies geschieht unter Einhaltung der Startzeitpunkt- bzw. der Vorrangsbedingung der einzelnen Arbeitsgänge sowie der Kapazitätsbeschränkung der erneuerbaren Ressourcen.<sup>16</sup> Die im folgenden aufgestellte Zielfunktion des RCPSP zur Minimierung der Projektdauer ist die gängige Version

---

<sup>10</sup>???

<sup>11</sup>???

<sup>12</sup>General Algebraic Modeling System

<sup>13</sup>???

<sup>14</sup>Vgl. Kellenbrink (2014), S. 11

<sup>15</sup>Vgl. Neumann-Braun et al. (2003), S. 21-22

<sup>16</sup>Vgl. Demeulemeester und Herroelen (2011), S. 23

der Kapazitätsplanung,<sup>17</sup> andere Variationen sind aber ebenfalls möglich.<sup>18</sup>

Nachfolgend wird das deterministische RCPSP in diskreter Zeit formuliert.<sup>19</sup> Charakteristisch für eine mathematische Modellformulierung in diskreter Zeit sind die Zeiteinheiten, die den Perioden  $t, \tau$  entsprechen.

## Modell RCPSP

$$\min Z = \sum_{t=FE_I}^{SE_I} t \cdot X_{I,t} \quad (1)$$

unter Beachtung der Restriktionen

$$\sum_{t=FE_i}^{SE_i} X_{it} = 1 \quad i = 1, \dots, I \quad (2)$$

$$\sum_{t=FE_h}^{SE_h} t \cdot X_{ht} \leq \sum_{t=FE_i}^{SE_i} (t - d_i) \cdot X_{it} \quad i = 1, \dots, I; h \in \mathcal{V}_i \quad (3)$$

$$\sum_{i=1}^I \sum_{\tau=\max(t, FE_i)}^{\tau=\min(t+d_i-1, SE_i)} k_{ir} \cdot X_{i\tau} \leq kp_r \quad r = 1, \dots, R; t = 1, \dots, T \quad (4)$$

$$X_{it} \in \{0, 1\} \quad i \in \mathcal{I}; t \in \{FE_i, \dots, SE_i\} \quad (5)$$

Es wird ein Projekt betrachtet, dass aus  $I$  unterschiedlichen Arbeitsgängen besteht. Jeder Arbeitsgang  $i$  hat eine definierte Menge von zu erledigenden Vorgängerarbeitsgängen  $h \in \mathcal{V}_i$ . Des Weiteren ist für die Fertigstellung des Projekts die Abarbeitung der Arbeitsgänge in topologischer Reihenfolge notwendig. D. h. der Vorgänger  $h$  hat stets eine kleinere Ordnungszahl als sein Nachfolger  $i$  ( $h < i$ ) und muss zur Fortsetzung des Projektverlaufs beendet sein. Die Bearbeitungsdauer eines Arbeitsgangs  $i$  wird mit dem Parameter  $d_i$  festgelegt. Bei dem RCPSP in diskreter Zeit wird die Annahme getroffen, dass die Dauer durch einen ganzzahligen Parameter abgebildet wird. Der Startzeitpunkt des Projekts ist  $t = 0$  und erstreckt sich über einen Gesamtzeitraum von  $T$  Perioden. Um die Reihenfolgebedingungen einzuhalten, werden einem Projekt zwei Dummy-Arbeitsgänge „Beginn“ ( $i = 1$ ) und „Ende“ ( $i = I$ ) hin-

---

<sup>17</sup>Vgl. Drexel et al. (1997), S. 98

<sup>18</sup>Vgl. Talbot (1982), S. 1200

<sup>19</sup>???



zugefügt, welche mit einer Dauer von 0 Zeiteinheiten bewertet werden.<sup>20</sup> Dadurch wird der Projektbeginn und das Projektende exakt terminiert. Der Parameter  $k_{ir}$  stellt die benötigten Kapazitäten der erneuerbaren Ressource  $r$  bei der Durchführung von Arbeitsgang  $i$  dar. Die Ressourcen  $r \in R$  sind in einer Periode innerhalb des Umfangs ihrer Kapazität  $kp_r$  nutzbar. Da es sich um erneuerbare Ressourcen handelt, stehen diese zu jeder neuen Periode in vollem Umfang erneut zur Verfügung. Ungenutzte Ressourcen sind jedoch nicht auf nachfolgende Arbeitsgänge und Perioden übertragbar.<sup>21</sup> Um den Fertigstellungszeitpunkt der einzelnen Arbeitsgänge  $i$  festlegen zu können, wird der Modellformulierung in diskreter Zeit die binäre Entscheidungsvariable  $X_{it}$  hinzugefügt.<sup>22</sup> Diese Binärvariable nimmt den Wert 1 an, falls der Arbeitsgang  $i$  zum Zeitpunkt  $t$  beendet wird.

Mittels der Zielfunktion (1) wird der Fertigstellungszeitpunkt des Projekts minimiert. Dafür wird der Zeitraum zwischen dem frühesten und spätesten Fertigstellungszeitpunkt  $FE_I$  und  $SE_I$  aller durchzuführenden Arbeitsgänge  $I$  betrachtet. Nebenbedingung (2) stellt sicher, dass ein Arbeitsgang  $i$  zwischen dem jeweiligen für diesen Arbeitsgang geltenden frühesten und spätesten Fertigstellungszeitpunkt nur exakt ein Mal durchgeführt wird. Die Reihenfolge restriktion wird mit der Nebenbedingung (3) eingehalten. Sie stellt sicher, dass jeder Vorgänger  $h \in \mathcal{V}_i$  beendet ist, bevor der Arbeitsgang  $i$  startet. Der Term  $(t - d_i)$  garantiert für den Arbeitsgang  $i$ , dass dieser erst beginnt, sobald der Vorgänger  $h$  mit der Dauer  $d_i$  abgeschlossen ist. Der Parameter  $kp_r$  spiegelt die Kapazitätsgrenze für eine erneuerbare Ressource  $r$  je Periode  $t$  wieder. In Nebenbedingung (4) findet zum einen eine formale Darstellung dieser Kapazitätsbegrenzung statt. Zum anderen wird der Ressourcenverzehr während der gesamten Bearbeitungsdauer der Fertigstellung beachtet, in dem der Kapazitätsbedarf  $k_{ir}$  aller Arbeitsgänge  $I$  summiert wird. Eben diese Summe wird schließlich durch  $kp_r$  beschränkt. Mit der Nebenbedingung (5) wird die Binärvariable  $X_{it}$  für den Zeitraum  $t = \{FE_i, \dots, SE_i\}$  formal definiert. Aufgrund der Reihenfolgebeziehung (3) darf der jeweils betrachtete Arbeitsgang nur in diesem Zeitraum fertiggestellt werden. Die gemischt-ganzzahlige Modellformulierung lässt sich durch Standard-Lösungsverfahren exakt lösen.<sup>23</sup>

## 2.2 Kostenplanung

Aufbauen auf der Kapazitätsplanung kann das RCPSP um die Nutzung von Zusatzkapazitäten der Ressourcen erweitert werden, damit dem Projektplanungsmodell gestattet ist den Vorgänge zusätzliche Kapazitätseinheiten der notwendigen Ressourcen bereitzustellen. Die Kapazitätsrestriktion wird dementsprechend um die Entscheidungsvariable  $O_{rt} \geq 0$  erweitert. Die Variable  $O_{rt}$  beschreibt die Einheiten an Zusatzkapazitäten einer Ressource  $r$  in der Periode  $t$ . Damit steht nicht die Einhaltung der verfügbaren Kapazitäten im Vordergrund,

---

<sup>20</sup>Vgl. Zimmermann et al. (2006), S. 66

<sup>21</sup>Vgl. Kellenbrink (2014), S. 12

<sup>22</sup>Vgl. Pritsker et al. (1969), S. 94

<sup>23</sup>z. B. mittels eines Branch-and-Bound-Verfahrens, Vgl. Kellenbrink (2014), S. 14

sonder unter Beachtung der Projektstruktur die aufgewendeten Zusatzkosten des Projekts. Dem Optimierungsmodell ist es damit gestattet durch Erhöhung der Kapazitäten der Ressourcen die anfängliche Ressourcenbeschränkung zu umgehen. Bei der Modellerweiterung der Kostenplanung wird der Parameter  $oc_r$  eingeführt, der für eine betrachtete Ressource  $r$  die Kosten einer Einheit der Zusatzkapazitäten beschreibt. Ziel des Optimierungsmodells ist es damit die Kosten des Projekts zu minimieren. Das Modell hilft damit der Entscheidung, ob durch Einführen der Möglichkeit von Zusatzkapazitäten das Projektziel verbessert erreicht wird. Es handelt sich um den Trade-off des frühzeitigen Erreichens des Projektziels durch Nutzung von Zusatzkapazitäten und der gesamten Projektkosten die für das Projekt aufgewendet werden sollen.

Nachfolgend wird das deterministische RCPSP+ in diskreter Zeit formuliert.

#### Modell RCPSP+

$$\min Z = \sum_{t=1}^T \sum_{r=1}^R oc_r \cdot O_{r,t} \quad (6)$$

unter Beachtung der Restriktionen (2), (3), (5) sowie

$$\sum_{i=1}^I \sum_{\tau=\max(t, FE_i)}^{\tau=\min(t+d_i-1, SE_i)} k_{ir} \cdot X_{i\tau} \leq kp_r + O_{rt} \quad r = 1, \dots, R; t = 1, \dots, T \quad (7)$$

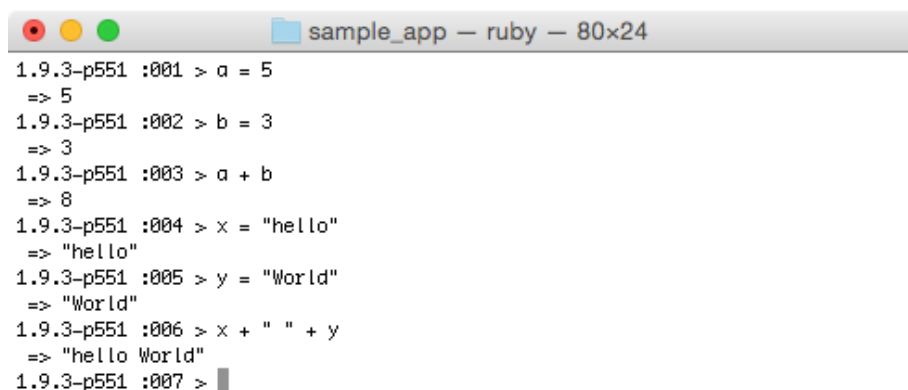
$$O_{rt} \geq 0 \quad r = 1, \dots, R; t = 1, \dots, T \quad (8)$$

Bei dem RCPSP+ wird die Zielfunktion insoweit formuliert, dass über alle Perioden  $t \in T$  und über alle Ressourcen  $r \in R$  die Summe der Kosten  $oc_r$  für die Anzahl an notwendige Einheiten an Zusatzkapazität  $O_{rt}$  minimiert wird. Weiterhin bleibt die Nebenbedingung (2) und (3) bestehen, dass jeder Vorgang exakt einmal zwischen dem frühesten Ende ( $FE_i$ ) und dem spätesten ( $SE_i$ ) fertiggestellt und die Topologie der Vorgänge eingehalten wird. Weiterhin gilt die Nebenbedingung (5), dass es sich bei der Entscheidungsvariable  $X_{jt}$  um eine binäre Variable handelt. Erweitert wird das RCPSP aus der Kapazitätsplanung mit einer modifizierten Nebenbedingung zur Einhaltung Kapazitätsbeschränkung. Mit der Nebenbedingung (7) wird die Kapazitätsrestriktion für eine Ressource  $r \in R$  in einer Periode  $t \in T$  eingehalten, jedoch ist es dem Modell gestattet die vorhandene Ressourcenkapazität  $kp_r$  um die Ausprägung der Entscheidungsvariable  $O_{rt}$  zu erweitern. Durch Lösen des Optimierungsmodells wird der Ablaufplan des Projekts unter Beachtung der unterschiedlich zulässigen Gesamtdauern  $SE_I$  generiert. Weiter wird für jede Ressource  $r \in R$  zur jeweiligen Periode  $t \in T$  die notwendige Anzahl an benötigter Zusatzkapazität  $O_{rt}$  ermittelt. Die Nebenbedingung (8) beschreibt die Eigenschaft der Entscheidungsvariable  $O_{rt}$ , dass es sich

um eine positive Variable bzw. einen Nullwert handelt.

## 2.3 Ruby on Rails

Das Frameworks Ruby on Rails (RoR) zur Entwicklung von Web-Applikationen mit Datenbankbezug wurde von David Heinemeier Hansson im Jahre 2004 erstmals vorgestellt.<sup>24</sup> Mit dem Name von RoR wird klar, dass das Framework die Programmiersprache Ruby nutzt. Ruby wird von den meisten gängigen Betriebssystemen unterstützt (Microsoft Windows, Apple Mac OS X, Linux, etc.) und ist bspw. dem Betriebssystem Apple Mac OS X in der Version 1.8.7 standardmäßig integriert.<sup>25</sup> Bei Ruby handelt es sich um eine objekt-orientierte Programmiersprache mit dem Grundsatz *principle of least surprise* und folgt einigen Besonderheiten, wie z. B. einer einfachen Sprachsyntax, keiner typisierten Variablen und einer reinen Objektorientierung.<sup>26</sup> Abbildung 1 zeigt das Terminal von Apple Mac OS X mit typischen Ruby Kommandobefehlen. RoR nutzt diesen einfachen Syntax zur Entwicklung von Web-Applikationen, wobei aufgrund einfacherer Bedienung auf integrierte Entwicklungsumgebung zurückgegriffen wird, wie z. B. RadRails oder RubyMine.<sup>27</sup>



```
1.9.3-p551 :001 > a = 5
=> 5
1.9.3-p551 :002 > b = 3
=> 3
1.9.3-p551 :003 > a + b
=> 8
1.9.3-p551 :004 > x = "hello"
=> "hello"
1.9.3-p551 :005 > y = "World"
=> "World"
1.9.3-p551 :006 > x + " " + y
=> "hello World"
1.9.3-p551 :007 >
```

Abbildung 1 Terminalfenster unter Apple Mac OS X

Mit Hilfe des RoR Frameworks lassen sich dadurch schnell Web-Applikationen mit Datenbankbezug entwickeln, wobei der wesentlichen Vorteile in der Softwarearchitektur des

---

<sup>24</sup>Vgl. Grimmer (2006)

<sup>25</sup>Vgl. Wintermeyer (o. J.)

<sup>26</sup>Vgl. ?, S. 297-298

<sup>27</sup>Vgl. Hartl (2012), S. 10

Model-View-Controller-Paradigmas liegt.<sup>28</sup> Das Paradigma besagt, dass eine durch einen Browser angestoßene Anfrage an den Server durch den Rails **controller** verarbeitet wird. Der **controller** verarbeitet die Anfrage und leitet die nachfolgenden Schritte ein. Bei Web-Applikationen erfolgt eine solche Verarbeitung durch anzeigen bzw. dem sogenannten *rendern* von HTML-Dokumenten der Rails **views**, die von Browsern angezeigt werden können. Der **controller** rendert die **views** und ermöglicht weitere RoR-Befehle im HTML-Dokument. Bei komplexen und dynamischen Seiten übernimmt der **controller** geforderte Daten aus den Rails **models**, die wiederum mit einer Datenbank verbunden sind. Durch diese Architektur lassen sich umfangreiche und an spezifische Anfrage angepasste Web-Applikationen entwickeln. Ein weiterer Vorteil von RoR ist die einfache Implementierung von Unterprogrammen. Ein Unterprogramm ist in Ruby/RoR ein **Gemfile**, das durch den Bundler zur bestehenden Web-Applikation hinzugefügt wird.<sup>29</sup> Im nachfolgenden Kapitel wird die Entwicklung einer Web-Applikation mittels RoR beschrieben. Dabei liegt die Besonderheit der Ausarbeitung auf Integration eines notwendigen Unterprogramms (**Gemfile**) und die Verbindung zum Programm GAMS, damit das in diesem Kapitel vorgestellte Projektplanungsproblem gelöst werden kann.

### 3 Implementierung des RCPSP mittels Ruby on Rails

IDEs=RubyMine

```
$ Put your code here.  
>> Put your code here.
```

Quellcode 1 Beispielcode

```
if draw  
  print([outputpath, 'mygraph.eps'], '-depsc')  
end
```

### 4 Kritische Würdigung des Anwendungssystems

### 5 Fazit

---

<sup>28</sup>Vgl. Walter (2008), S. 463

<sup>29</sup>Vgl. Hartl (2012), S. 9-17

# Literatur

- Bartels, J.H. (2009): Projektplanung–Grundlagen und Anwendungsbeispiele. In: Anwendung von Methoden der ressourcenbeschränkten Projektplanung mit multiplen Ausführungsmodi in der betriebswirtschaftlichen Praxis. Springer, S. 7–42.
- Demeulemeester, E. und Herroelen, W. (2011): Robust project scheduling. Bd. 3. Now Publishers Inc.
- DIN 69900 (2009): Projektmanagement - Netzplantechnik; Beschreibung und Begriffe. In: Berlin: Beuth.
- Drexl, A.; Kolisch, R. und Sprecher, A. (1997): Neuere Entwicklungen in der Projektplanung. In: Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung, S. 95–120.
- Felkai, R. und Beiderwieden, A. (2011): Analysieren und Formulieren von Projektzielen. In: Projektmanagement für technische Projekte. Springer, S. 45–64.
- Grimmer, L. (2006): Interview with David Heinemeier Hansson from Ruby on Rails. <https://web.archive.org/web/20130225091835/http://dev.mysql.com/tech-resources/interviews/david-heinemeier-hansson-rails.html>.
- Hartl, M. (2012): Ruby on Rails Tutorial: Learn Web Development with Rails. Pearson Education.
- Kellenbrink, C. (2014): Einführung in die ressourcenbeschränkte Projektplanung. In: Ressourcenbeschränkte Projektplanung für flexible Projekte. Springer, S. 5–18.
- Neumann-Braun, K.; Schwindt, C. und Zimmermann, J. (2003): Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. Springer.
- Pritsker, A.A.B.; Waiters, L.J. und Wolfe, P.M. (1969): Multiproject scheduling with limited resources: A zero-one programming approach. In: Management science. Bd. 16, Nr. 1, S. 93–108.
- Talbot, F.B. (1982): Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. In: Management Science. Bd. 28, Nr. 10, S. 1197–1210.
- Voigt, K.I. und Schewe, G. (2014): Definition Projekt Version 7 - Gabler Wirtschaftslexikon.
- Walter, T. (2008): Das Ruby-Framework Ruby on Rails. In: Kompendium der Web-Programmierung. Springer, S. 463–491.
- Wintermeyer, S. (o. J.): Installation von Ruby on Rails 3.2 mit RVM. <http://ruby-auf-schienen.de>. <http://ruby-auf-schienen.de/3.2/rails3-install-osx.html>.

Zimmermann, J.; Stark, C.; Rieck, J. et al. (2006): Projektmanagement. In: Projektplanung. Springer Berlin Heidelberg, S. 1–113.

## A Anhang

### A.1 GAMS-Implementierung des Beispiels

### A.2 Ruby on Rails Programmcode