INTERACT – Interactive Manual Assembly Operations for the Human-Centered Workplaces of the Future

Grant Agreement Number	**:**	**611007**
	:	**INTERACT**
Project Start Date	**:**	**1st October, 2013**
Consortium	**:**	**DAIMLER AG (DAIMLER)- Project Coordinator** **ELECTROLUX ITALIA S.P.A. (ELECTROLUX)** **INTRASOFT INTERNATIONAL SA (INTRASOFT)** **IMK AUTOMOTIVE GMBH (IMK)** **EMPHASIS TELEMATICS AE (EMPHASIS)** **HADATAP SP ZOO (HADATAP)** **UNIVERSITY OF PATRAS (LMS)** **UNIVERSITAET ULM (IMI)** **DEUTSCHES FORSCHUNGSZENTRUM FUER KUENSTLICHE INTELLIGENZ GMBH (DFKI)**

Title	**:**	**Deliverable 2.1.1 - Manual assembly simulation design and data formats**
Reference	**:**	**D2.1.1**
Availability	**:**	**public**
Date	**:**	**31.07.2014**
Author/s	**:**	**IMK, DFKI, DAIMLER, LMS**
Circulation	**:**	**EU, consortium**

Summary:

The content of this document is the description of the progress in developing the INTERACT software prototype. The main purpose is the presentation of the considered architecture, the technologies used, the progress in the development and the identification of further development steps and strategies.

D2.1.1 is the main outcome of Task 2.1 and is mainly based on the requirements of the deliverable 1.1.1.

Contents

1. EXECUTIVE SUMMARY

The content of this document is the description of the progress in developing the INTERACT software prototype. The main purpose is to present the architecture considered, the technologies used, the progress of the development and the identification of further development steps and strategies.

D2.1.1 is the main outcome of Task 2.1 and is mainly based on the requirements of the deliverable D1.1.1.

Deliverable D1.1.1 states three general requirements on a future human assembly task simulation tool:

- Improvement of the realism of the digital models of manual assembly operations, as well as increased confidence in the simulation results.
- Improvement of the performance of the planned assembly processes (for example in terms of ergonomics, throughput and utilization).
- Reduction of the time required to build digital models of assembly processes.

Section 2 and 3 provide an overview of the main ideas and definitions of the earlier deliverables and the transfer to the actual topic. The high level architecture of the software is presented, as it is planned up to this point. Furthermore, a description of the individual components including the functionality, methodological basics and a brief description of used algorithms, used technology and data formats in the sections 4 to 7.

2. INTRODUCTION

Work package 2 carries some of the core functionalities of the project as it is related to the motion synthesis. Up to this point of the project, the focus of the development has been on the logical concept, respectively the high level structure of the motion synthesis. Following Kruchtens' architectural view model (see Figure 1), the project shifts from the logical and therewith user-centered view to the development and process perspective. From the user-centered perspective, the basic structure of the software consists of input, motion synthesis, visualization, assessment and manipulation (see Figure 2). These modules work in a loop, regarded to the iterative character of the workshop use case (see D1.1.1).

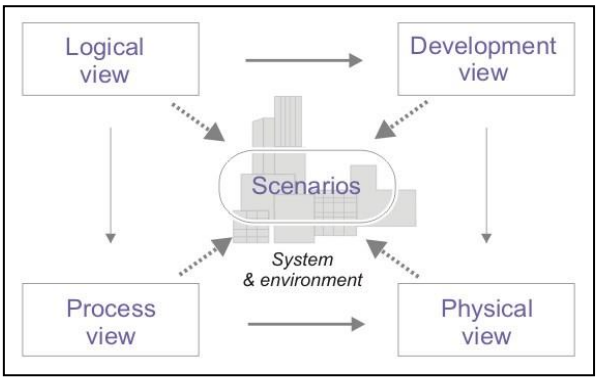

Figure 1: '4+1' architectural view model (Kruchten, 1995)

The next chapter will give a software overview to shift the perspective to a more development oriented view. The subsequent chapters will describe the main components of the motion simulation modules individually.

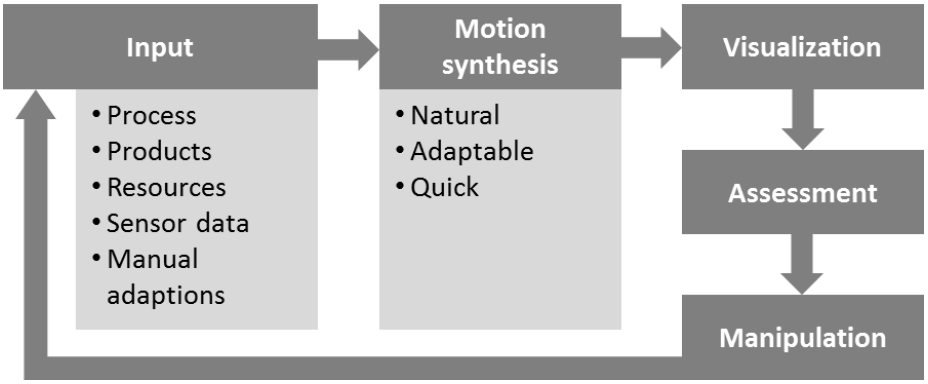

Figure 2: Logical overview from user perspective

3. MOTION SYNTHESIS DESIGN OVERVIEW

Motion synthesis in the INTERACT project is based on three key features:

- Input through controlled natural language (CNL)
- Motion synthesis with Motion graphs++
- Generation of alternative simulations in a workshop scenario

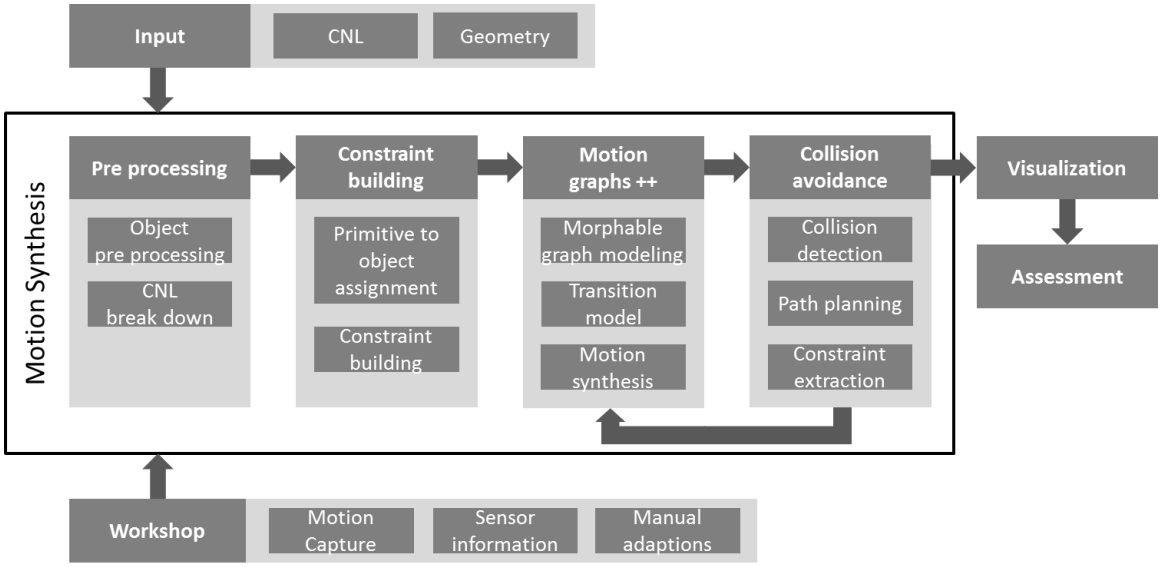

Figure 3: Motion synthesis overview

These features are the starting point for the development of a comprehensive motion synthesis module. These key features try to meet the three general requirements mentioned above. Next to these core technologies, there are several modules necessary to get to a comprehensive motion synthesis system. It is not possible to solely feed the Motion graphs++ algorithms with the semantic task description of the CNL. Motion graphs ++ requires geometric constraints (e.g. end effector positions) as input. Therefore, the CNL task description has to be interpreted and assigned to the virtual environment and the task relevant geometry. This pre-processing step requires an explicit object representation of all relevant geometry. After the assignment of relevant objects, references and positions in the observed process, in a constraint building step, the constraints for Motion graphs ++ are generated. The Motion graphs ++ step follows the procedure of graph modelling. The resulting graph comprises motion primitives as vertices and transition models as edges. This initial motion synthesis is followed by a collision avoidance component. This component consists of collision detection, the calculation of collision free trajectories and the extraction of a new updated constraint set for Motion graphs ++ motion generation. These steps try to ensure a natural looking collision free and consistent visualization of the planned work tasks.

The key feature of quick adaptation of motion simulations consists of motion capture, manual input and sensor data input. All three ways of changing the simulation generate a set of constraints as input to the preprocessing module, the geometry or the constraint building, where they have to be integrated and synchronized.

4. CNL TO EXPLICIT TASK DESCRIPTION

The Controlled Natural Language (CNL) in INTERACT is characterized by a very limited set of English words describing assembly tasks. Basically it consists of more or less complex verb phrases, as shown in example (1). To describe a task in this CNL, the user selects the activity (tighten), the individual components (arm support, cordless screw driver, middle console) and their roles in the phrase (object, tool, goal) using a menu-based interface.

(1) *Tighten arm support with cordless screw driver on middle console*

To visualize the semantics and pragmatics of a CNL expression, a breakdown into terms of elementary movement actions such as PICK, WALK, PLACE, SCREW etc.[1] is required, representing the scenario specific needs of the avatar getting hold of the screw driver, the screws and the arm support, which may involve various instances of walking, picking, etc.

Obviously many parameters influence the generation of a precise sequence of elementary actions to be visualized. Some parameters depend on the objects at stake. For instance, a screw is picked up differently than an arm support (one vs. both hands). The knowledge about each object being grasped by an avatar is depicted in the ontology. Other parameters are based on the state of the scenario at the point of time when the action is carried out. They include the positions of the avatar and of the objects at stake.

It is assumed that the INTERACT system has complete information of all objects that may be involved in assembly operations at any time. Thus it is possible to organize the breakdown into elementary movement actions by "asking" the environment. We define:

- pos(*x*) to be the position of *x* at a given state of the system;
- near(*x*) to be a predicate stating whether the position of the avatar is close enough to *x* so that it can PICK *x* or do some work, such as tighten something;[2]
- holds(*x*) is a predicate stating whether contact is established between *x* and one or both of the hands of the avatar.

This way it is possible to state whether, for instance, the avatar holds the screw driver, whether it can pick it, or whether it needs to walk near it in order to pick it.

The breakdown is implemented as a production system of condition-action rules with post-conditions representing the changes a rule has caused. Each CNL assembly task description – like (1) – is analyzed to match a semantic pattern associated with a set of condition-action rules describing all possible breakdowns within the scope of defined parameter values.

Let's look at an example to illustrate the techniques employed. The linguistic analysis (1) may result into the semantic representation (2). A generalizing pattern covering tightening actions is given in (3). Note that the class of the tool is restrained to the class *screw_drv* since only screw drivers imply that screws are needed.

(2) [PRED tighten, THEME arm_suppt, TOOL cdless_drv, GOAL mico]

(3) [PRED tighten, THEME *theme:thing*, TOOL *tool:screw_drv*, GOAL *goal:thing*]

The activity (2) requires, among other things, the avatar to get hold of the screw driver, the screws and the arm support. This can be described in terms of the following generic rules about picking, carrying and placing things. The post-conditions are prefixed by the "->" sign.

(4) *;;pickrule(object) - if successful,* holds *holds as a post-condition.*

[1] See Deliverable 1.1.1 for a complete list of elementary actions.

[2] "near" can be defined further in terms of the positions of the object and the avatar. NOT(near()) triggers a WALK, as shown below.

IF NOT(holds(*object*))

THEN IF NOT(near(*object*)) THEN (WALK(near(*object*)); PICK(*object*))

-> holds(*object*), …

(5) *;;Carry(object, goal)*

IF holds(*object)* THEN IF NOT(near(*goal*)) THEN WALK(near(*goal*))

-> pos(*object*) = near(*goal*)

(6) *;;placerule(object, goal)*

IF holds(*object*) THEN IF near(*goal*) THEN PLACE(*object, goal*)

-> pos(*object*) = on(*goal*)[3]

Getting hold of the objects, as required by (2), can now be expressed as sequences of *pickrule-carry-placerule* activities for the screw driver, the screws, and the arm support.[4] The arguments are called by value using the template (3), which is instantiated by the semantic representation (2). Note that at this level, only elementary actions, logical connectives and functions "asking the environment" are needed. At the same time, implicit activities such as walking or fetching the screws, are taken care of.

The rules are interpreted to derive an optimum sequence of elementary actions. Parameter settings, which are not shown here in full, may allow for more than one solution. An optimum solution can be derived on statistical grounds if no further selection criteria are available. The algorithm is based on the three-step cycle of matching, selecting and firing a rule as known from the production system literature [2]. Post-conditions modify the modeled state of the system.

The result of the breakdown is a sequence of elementary actions that will guide the visualization processes. This action sequence is accompanied by a set of motion constraints, e.g. if a screw driver is going to be picked then a constraint for the hand joint position is derived from the screwdriver hand attachment tag.

The breakdown is represented as a feature structure[5] in an XML format of choice.

[3] Assuming that PLACE knows a default position depending on the classes of its arguments.

[4] Obviously, in order to minimize WALK events, things may be carried jointly. Corresponding rules must be defined additionally.

[5] A feature structure is recursively defined as a tree structure, in which each node either represents a feature-value pair, or a feature structure.

5. MOTION SYNTHESIS

The goal of motion synthesis is the automatic generation of realistic or believable animations of a virtual human character for a large repertoire of behaviors controlled by user input and environmental constraints. In the INTERACT project the motion synthesis is based on the data-driven approach Motion graphs++ [3].

In general, the implementation of this approach uses motion data (here BVH files [4]) as input, performs dimension reduction and trains a set of statistical models on the reduced motion parameters for different motion primitives, e.g. left step and right step. A set of statistical models is then formed into a graph based on reasonable transitions between them. Transitions themselves are also represented using statistical models. This graph effectively forms a large "motion space", which can be sampled to synthesize new animations satisfying user specified and environmental constraints, e.g. a constraint on the hand position.

As shown in Figure 4, the Motion graphs++ algorithm consists of three parts: preprocessing, modelling of the morphable graph and the actual motion synthesis.

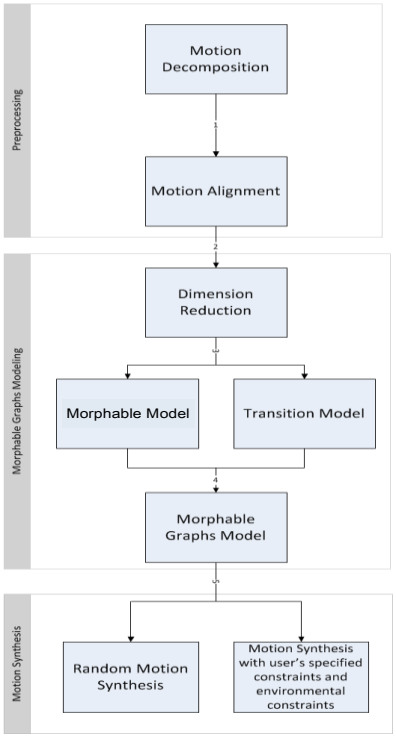

Figure 4: Workflow of Motion Graphs++

5.1. Preprocessing

As the first step of the preprocessing, a set of motion files is decomposed into segments of structurally similar motion primitives. This motion decomposition is done automatically based on the detection of user defined key frames. In addition to key frames, feature detection can also be used for the segmentation. Currently, a solution for the detection of walking primitives that is based on the distance between the feet has been developed.

For each motion primitive one example segment is selected as reference motion to define a canonical timeline. All the other examples of each motion primitive are then aligned to their reference motion segment based on Dynamic Time Warping (DTW) and a pose distance measure. The resulting aligned motion segments and time warping functions are suitable for generative statistical modeling.

5.2. Morphable Graphs Modelling

For the aligned motion data and separately for the time warping functions Principal Component Analysis (PCA) is applied to encode the kinematic and time variation into a linear combination of their principal components. This results in a dimension reduction of the motion parameters. Then a Gaussian Mixture Model (GMM) is trained for each motion primitive to model the distribution of these motion parameters in the original examples. This statistical model is called a morphable model.

The morphable graph is then manually defined as a directed graph of these models and Gaussian Process (GP) regression is applied to model the transition distribution for all known transitions between nodes in the graph. Known correspondences between parameters from subsequent examples from the preprocessing step are used as training data for the GP.

5.3. Control

The problem of controlled motion synthesis is formulated as an optimization problem, where user specified and environmental constraints form an error function that needs to be minimized while maximizing the likelihood of the statistical model of the original examples.

This problem is solved in two steps: First an optimal graph walk is found based on randomly sampled motion parameters and the enumeration of every possible graph walk of a user specified length. Then a gradient based optimization approach is applied to find the optimal motion parameters for this graph walk. Currently only the optimization of kinematic constraints is implemented.

6. COLLISION AVOIDANCE

In daily life people consciously or unconsciously avoid collisions. On the other hand in some cases people allow collision; this can be due to the fact that without collision it is not possible to execute the task, for example working in a cluttered environment. In a typical assembly line, the collision between the operator and the external objects should be avoided to improve the shop-floor safety, ergonomics and productivity. Digital human models (DHMs) are nowadays used to assess the ergonomics of the shop-floor proactively. The DHM simulation is used to design the shop-floor, plan the activities, modify the tasks etc.

Collision avoidance is a well-studied field in various domains such as computer graphics, robotics, animation etc. Many closed form and iterative algorithms are available in literature [5,6,7,8].

6.1. Methodology

In INTERACT, collision avoidance will be realized by adding collision constraints for the Motion graphs ++ algorithm. This approach differentiates walking / carrying activities, in which 2D collision avoidance is employed and other activities.

It is assumed that in assembly scenarios walking / carrying collisions are mainly created by obstacles and that the height of the obstacle is not important as the walk path are computed using the points projected onto the X-Y plane. Therefore, collision constraints are pre-processed before any trajectory is derived from Motion graphs ++. It is planned to employ a standard algorithm for walk path planning. The algorithm A* [9] is considered a good candidate to compute the shortest collision free path that exists between the initial position of the DHM and the target position. In the project, an approach to realistically smooth results based on Motion Capture data and to derive a minimum set of constraints has been developed. Algorithmic details of the developed approach have been published in [10].

For other activities, collision constraints are derived in an iterative process: First, a trajectory is sampled based on constraints that result from the explicit task description (see section 4) as well as the above described walk / carry constraints. This motion is likely to exhibit collisions. These collisions are addressed locally. First, collisions are detected and mapped onto motion primitives. Therefore, the motion primitives between which the collisions happens and themselves do not exhibiting collisions, are detected. Thus, it is possible to find out the *border* motion primitives and the in-between motion primitives that exhibit collision. These borders are used as start and end points for an algorithm that yields as few as possible new constraints that resolve the respective collision. The resulting collision constraints are checked for consistency with existing ones. Then a new trajectory is sampled from Motion graphs ++ using all constraints. The new trajectory is again checked for collisions and so on until either the path is collision free or a time limit is exceeded.

7. VIEWER/3D SCENE MANIPULATOR

In order to setup 3D environments for the motion synthesis algorithm and to visualize the results, a 3D user interface is going to be implemented by DFKI as a web application based on its XML3D technology. The development time of this component can be reduced by building on top of existing applications developed by DFKI.

7.1. XML3D and Xflow

XML3D is designed as an extension of HTML5 to make the 3D capabilities of web browsers provided by the WebGL API accessible to web developers via JavaScript. Therefore the 3D viewer/scene manipulator will be accessible in any web browser supporting the WebGL standard. For the scene definition and event handling XML3D makes use of the Document Object Model (DOM) of a web page by extending HTML5 with new elements for 3D graphics (geometry, materials, lightning, etc). 3D geometry can also be retrieved from distributed sources via URIs and semantic annotations of objects is also supported out-of-the-box via RDFa.

With Xflow XML3D also integrates a declarative data processing language that can be used to implement data intensive tasks specified as network flows. In the context of INTERACT Xflow is used for the implementation of mesh animations based on a skinning for a skeleton provided in a custom JSON-based format. An already existing Xflow-skinning for the RocketBox-skeleton can be reused from earlier Xflow demos developed by DFKI (Figure 5 shows an example Xflow animation).

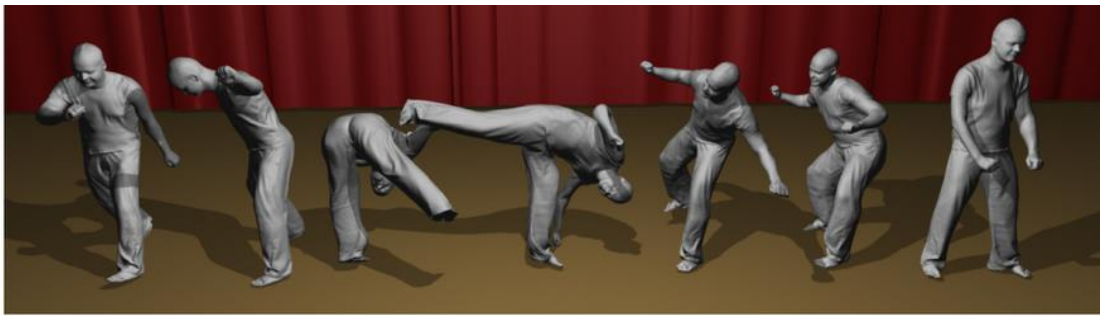

Figure 5: Playback of a multi-video performance capture in XML3D using Xflow

7.2. COMPASS and FiVES

In order to make the scene be synchronously viewed and edited on multiple clients, for example during the INTERACT assembly workshop, COMPASS (Collaborative Modular Prototyping And Simulation Server) is going to be used to manage the scene model. COMPASS is a web-client-server framework developed by the DFKI that is built on top of the JBoss middleware [11]. Access to the scene and other functions of the server is provided using REST-based web services with role based access control. In addition to that clients can communicate with each other via the "Extensible Messaging and Presence Protocol" (XMPP), which allows P2P sessions.

Furthermore, the extension of the COMPASS application server with the "Flexible Virtual Environment Server" (FiVES) enables collaborative work on dynamic scenes with animations as required for INTERACT. The motion synthesis server can then be integrated into this framework as an external Web service or directly as a plugin of COMPASS that has local access to the scene. Each frame of an animation generated by the motion synthesis server will then automatically be send to all connected clients.

8. ABBREVIATIONS

CNL	Controlled Natural Language
COMPASS	Collaborative Modular Prototyping And Simulation Server
DHM	Digital Human Modeling
DTW	Dynamic Time Warping
FiVES	Flexible Virtual Environment Server
GMM	Gaussian Mixture Model
GP	Gaussian Process
JSON	JavaScript Object Notation
OBB	Oriented Bounding Box
P2P	Peer-to-Peer
PCA	Principal Component Analysis
RDFa	Resource Description Framework in Attributes
URI	Uniform Resource Identifier
WebGL	Web Graphics Library
XML3D	Extensible Markup Language 3D Graphics
XMPP	Extensible Messaging and Presence Protocol

9. REFERENCES

1. Kruchten, P. B.: The 4+1 View Model of architecture. In: IEEE Software. 12, Nr. 6, S. 42–50, doi:10.1109/52.469759 (1995)
2. Davis and King. “An overview over production systems”. In: Eclock/Michie (1977)
3. MIN, J., and CHAI, J. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. In ACM Transactions on Graphics, 31(6) (2012)
4. Meredith, M. and Maddock, S.: Motion Capture File Formats Explained. Department of Computer Science, Sheffield. http://www.dcs.shef.ac.uk/~. (2001)
5. J.C. Latombe: Robot Motion Planning. Kluwer Academic Publishers, Boston, MA (1991)
6. LaValle, Steven M.: “Planning Algorithms.” http://ebooks.cambridge.org/ref/id/CBO9780511546877 (2006)
7. Conkur, E.S.: “Path planning using potential fields for highly redundant manipulators”. Robotics and Autonomous Systems 52. 209–228. (2005)
8. Udupa, Shirram M. :“Collision detection and avoidance in computer controlled manipulators.” (1999)
9. Hart, P. E.; Nilsson, N. J.; Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics SSC4* **4** (2): 100–107 (1968)
10. Manns, M. and Arteaga: Automated DHM Modeling for Integrated Alpha-Numeric and Geometric Assembly Planning, M. Abramovici and R. Stark (Eds.): Smart Product Engineering, LNPE, DOI: 10.1007/978-3-642-30817-8_32, pp. 325–334 (2013)
11. Jbosshttp://www.jboss.org (2014)