The Visual Computer manuscript No.
This is a pre-print of an article published in The Visual Computer.
The final authenticated version is available online at: https://doi.org/10.1007/s00371-020-01792-x

Combining Heterogeneous Digital Human Simulations

Presenting a novel co-simulation approach for incorporating different character animation technologies

Felix Gaisbauer · Eva Lampen · Philipp Agethen · Enrico Rukzio

Received: date / Accepted: date

Abstract Digital human simulation is important for various domains such as the entertainment, health care and production industries. A variety of simulation techniques and tools are available, ranging from motion-capture based animation systems and deep learning to physics-based motion synthesis. Each technology has its advantages and disadvantages and is suited for particular use cases. Therefore, a combination of multiple technologies would result in more sophisticated simulations, which can address heterogeneous aspects. However, the different approaches are mostly tightly coupled with the development environment, thus inducing high porting efforts if being incorporated into different platforms. A combination of separately developed simulation systems, either for benchmarking or comprehensive simulation is not possible yet. For the domain of plant simulation, the Functional Mock-up Interface (FMI) standard has already solved this problem. Initially being tailored to industrial needs, the standards allows exchanging dynamic simulation approaches such as solvers for mechatronic components. Inspired by the FMI standard, we present a novel framework to incorporate multiple digital human simulation approaches from multiple domains. In particular, the paper introduces the overall concept of the so-called Motion Model Units, as well as its underlying technical architecture. As main contribution, a novel co-simulation for the orchestration of multiple digital human simulation approaches is presented. The overall applicability is approved based on a quantitative evaluation using motion capture data and a user study.

F. Gaisbauer
Daimler AG, Wilhelm-Runge-Str. 11, 89081, Ulm, Germany
Tel.: +49-731-5052414
E-mail: felix.gaisbauer@daimler.com

E. Lampen
Daimler AG, Wilhelm-Runge-Str. 11, 89081, Ulm, Germany

P. Agethen
Daimler AG, Wilhelm-Runge-Str. 11, 89081, Ulm, Germany

E. Rukzio
University of Ulm, James Franck Ring, 89081, Ulm, Germany

1 Introduction

For many branches such as the entertainment or manufacturing industry, digital human simulation has become an essential technology. Whereas the entertainment sector relies on character animation algorithms to generate naturally interacting avatars, manufacturing industries use the approaches to plan and assess manual processes in a digital way. During the last two decades, a vast progress in character animations technologies has been achieved, resulting in more natural, accurate and computational efficient approaches. The available techniques can be generally subdivided into data-driven and model-based methods [26]. Data-driven systems synthesize motions based on underlying motion capture data and generate realistic motions with low computational efforts. Model-based approaches, in contrast, model the principles of the motion generation itself and are thus more generic [26]. In academia, both methods are important subjects of research, whereas ground-breaking publications related to data-driven [19, 22, 23] and model-based [32, 11] approaches were presented.

Even though a large progress in these areas has been achieved, currently no technique exists which accurately covers the multitude of humanoid motion generation scenarios for different use-cases. Moreover, the individual techniques are mostly realized in separate systems and tailored to these platforms. For many use-cases, the incorporation of these different technologies in a com-

mon system would enable great benefits. For instance, academia could profit by the possibility to benchmark different technologies in a common platform or examine the interplay and its possibilities. On the other side, industry could realize more comprehensive simulations based on the combination of different approaches, ultimately covering all aspects of the specific use-cases.

Presently, the incorporation of different technologies induces high porting efforts and costs. Standardized components, embedding heterogeneous approaches would significantly reduce these efforts. In the domain of plant simulation, the Functional Mock-up Interface (FMI) [6] already solved this problem by encapsulating simulation approaches using a common interface. Inspired by the FMI standard, we present a modular framework incorporating arbitrary character animation algorithms using so-called Motion Model Units (MMUs). Based on the framework, a novel co-simulation being able to orchestrate different units without explicit motion knowledge is presented. In particular, the main contributions of the presented work are the following:

a) Overall concept of a modular framework for digital human simulation approaches (Section 3.1).
b) Technical framework for coupling heterogeneous digital human simulations (Section 3.2).
c) Co-simulation concept to generate feasible motions from multiple MMUs (Section 4).
d) Exemplary implementation and validation of a system combining the previous aspects (Section 5).

The remainder of the paper is structured as follows: First, the state of the art regarding digital human simulations is revisited. Second, the general concept of the Motion Model Unit framework and its technical architecture is introduced (Section 3). A challenging problem linked to the modular concepts, is the coordination and co-simulation of the different units containing diverse motions. Based on the imposed restrictions, in Section 4, a novel co-simulation concept is proposed being able to orchestrate multiple MMUs. In Section 5, the validity of the co-simulation is examined within two different scenarios. Whereas the first scenario examines the general validity of the co-simulation based on motion-capture data, the second evaluation investigates the applicability using exemplary motion synthesis algorithms. The paper ends with a further conclusion and an outlook.

2 Related Work

The presented framework builds upon a large body of related work. In the following, an overview of related work in the context of digital human simulation, exchange of simulation approaches and co-simulation is provided.

2.1 Digital Human Simulation

Digital human simulations are nowadays increasingly important for various branches. An exhaustive number of technologies and tools for simulating human motion have been developed for various scopes of application.

2.1.1 Motion synthesis technologies

Motion synthesis technologies build the base for digital human simulation and are applied in heterogeneous domains. In general, the technologies can be subdivided into data-driven and model-based approaches [26].

The majority of the data-driven approaches use motion blending techniques and blending trees contained in state machines. These methods provide naturally looking motions while being computationally efficient. The existent approaches can be subdivided into Barycentric-, K-Nearest-Neighbor-, Radial Basis Function-interpolation and Inverse blending [12]. Based on automatic compositions of segmented motion capture clips, motion graphs generate sequences of natural motions without the need to explicitly model state machines [22,25]. Furthermore, based on unstructured motion capture data, the authors of [23] organize motion data into a high-dimensional generalization of a vector field, called a motion field, whereas appropriate postures are determined for each frame. Recently, an approach called motion matching was presented in [9], efficiently searching in unstructured motion capture databases. Due to the growing computational capabilities, moreover, deep learning based approaches received significant attention. Recent works present deep learning based animation systems [19,24].

Besides the data-driven approaches, model-based systems are also intense subject of research. In this category physics based character animations are frequently used. The approaches can be subdivided into trajectory optimization and reinforcement learning [26]. Approaches like [31] model the locomotion behavior based on an inverted pendulum. Others, such as [11] present composable controller for physics based simulation. Moreover, inverse kinematics (IK) approaches such as [4,7] are also frequently used to compute postures of digital avatars. In practice, IK is oftentimes utilized in combination with data-driven approaches or path planning algorithms.

Whereas data-driven approaches rely on motion capture data, model-based systems generate motions based on mathematical and physical models. Consequently, the first only cover the range given by the underlying data, however, generating naturally looking results. On the other hand, the latter are more generic but also more difficult to parameterize. The applicability of the technologies therefore strongly depends on the use-case.

2.1.2 Digital Human Simulation tools

Based on the available motion synthesis technologies, various tools for simulating human motion have been developed for different scopes of applications.

Tools like IPS IMMA, Santos and Siemens Jack focus on the analysis and design of workplaces and products. Since the addressed use-cases contain collision-afflicted environments, the systems use model-based simulation technologies. Musculoskeletal and bio-mechanical modeling tools like AnyBody and OpenSim [10] use highly detailed DHMs including a fine-grained representation of musculoskeletal- or organ-system. These tools precisely model motions of the human body, however, at the expense of real-time capability. Another cluster receiving significant attention, is the group of game engines like Unity3D, Unreal Engine and CryEngine. These applications provide gaming-related platforms including various tools (e.g. retargeting, blending) to easily animate human motion. Even though achieving outstanding results in terms of naturalness, difficult movements in collision-afflicted settings can only be scarcely simulated.

2.1.3 Modular Digital Human Simulation frameworks

Besides the mentioned tools, there are also systems available focusing on modular simulation. The HUMOSIM framework [27] is an approach primarily utilized for ergonomics simulation. The framework contains modules based on closed-form equations to control end-effectors of a digital human. Further available systems are ADAPT [29] and Real actor [8]. Whereas the first is used for agent prototyping, Real actor represents a behavior realization system for embodied conversational agents. Most related to our approach, Smartbody [30] provides an animation system focusing on the generation of human motion using hierarchical motion controllers. These controllers are embedded in the Smartbody platform, thus being limited in their interoperability. Moreover, the authors explicitly state that they do not intent to create a platform independent and modular architecture for exchanging character animations systems, since in their opinion those architectures either under-specify the interface or restrict the capabilities [28].

The vision of the proposed work in this paper, first published in [16], is to provide a modular simulation, inspired by the motion controllers of Smartbody. However, compared to Smartbody, it is the explicit target to create a platform and technology independent approach across the boundaries of heterogeneous systems. Even though this genericity induces increased manual efforts for the parameterization, it is considered as more important than easy parameterization in rather restricted environments as provided by other frameworks. To realize a generic encapsulation, so called Motion Model Units (MMUs) are used encapsulating the specific technologies and algorithms [15] across different environments. The novel method builds upon the core idea of hierarchical motion controllers and significantly extends the concepts based on the capabilities of the MMU framework.

2.2 Exchanging simulation approaches

For exchanging motions between different simulation tools, there are various formats such as Biovision Hierarchy (bvh) and Filmbox (fbx) available. Even though being widely used, they are only capable of storing pre-generated motions. Hence, it is not possible to integrate motion generations algorithms within the files itself.

For exchanging simulation functionality in a different domain, a widely used solution is available. Functional Mock-up Interface (FMI) is a standard that supports the exchange of dynamic simulation models as well as its co-simulation while being tool independent [2]. An instance of an FMI component is called a Functional Mock-up Unit (FMU). Using the FMI standard, it is possible to perform a simulation of different FMUs, containing appropriate solvers, whereas only the simulation results of the FMUs are exchanged after defined time steps. This approach is referred as FMI for co-simulation [6]. The concept of modular motion units, which is also referred as Motion Model Interface (MMI) approach, builds upon the idea of the FMI concept to further extend the standard to simulate human motion

2.3 Co-simulation

Orchestrating various sub-simulations as intended by the FMI or MMI approach, requires a superior instance managing the distributed sub-systems. In general, this orchestration process is named co-simulation, whereas the co-simulator updates the components and incorporates the results. Recently, in literature various co-simulation approaches for the FMI standard have been proposed [5,32,33], however, these systems predominantly focus on signal flow modeling. Since the co-simulation of character animation systems has entirely different requirements, these solutions cannot be directly used.

Summarizing the state of the art, it can be stated that no approach is currently available for the orchestration of heterogeneous character animation approaches. To bridge this gap, in this paper, a novel co-simulation concept is proposed which can be applied to the MMI approach. The concept allows to orchestrate various character animation systems in a common system.

3 The Motion Model Interface - Framework

Inspired by the FMI approach, we present a framework for exchanging character animation systems based on [16,17]. In the following, first, the overall concept is presented, whereas the technical architecture is addressed subsequently. A detailed specification is available at [14].

3.1 Overall concept

With the FMI standard, complex systems like industrial machines can be simulated using specialized approaches such as solvers for pneumatic cylinders or kinematic models. The respective sub-simulations are embedded in standardized modules (FMUs) [6], whereas several of these simulations are sequenced by a co-simulator. This component communicates with the FMUs at discrete points in time and incorporates the computed results in a common simulation. Transferring this concept to the domain of character animation, a so-called Motion Model Interface (MMI) and its implementations called Motion Model Units (MMUs) are presented which allow to incorporate diverse character animation approaches into a common framework. Fig. 1 shows the main idea of the approach.

3.1.1 Motion Model Units

The proposed MMUs are the fundamental part of the modular concept and provide the basic interface for encapsulating different character animation systems. These units contain the actual animation approach, being implemented in the required platform and programming language. For instance, an actual MMU can comprise a data-driven algorithm in Python, as well as model-based approaches realized in C++. By utilizing a common interface, and inter-process communication, the MMUs can be accessed independent of the platform. Fig. 2 depicts the provided key functionality of the interface.

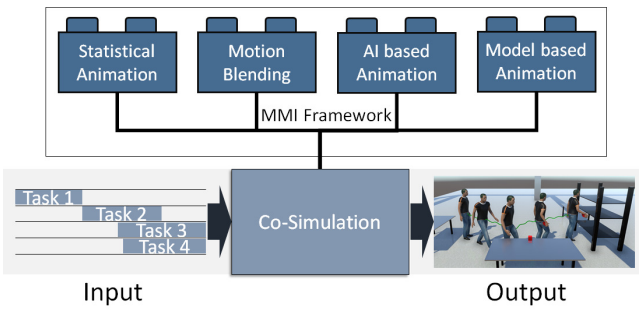

Fig. 1 Principle of the Motion Model Unit approach. By using standardized units and a co-simulation, multiple systems can be incorporated into a common platform.

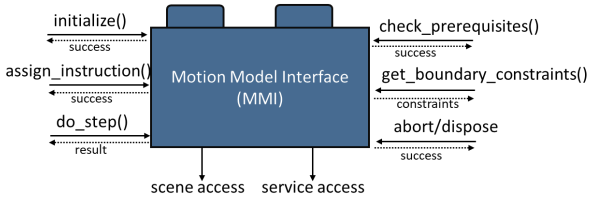

Fig. 2 Illustration of the key functionality of the proposed Motion Model Unit interface.

The individual MMUs are responsible for generating specific motions (e.g. walk or grasp). To externally specify the intended behavior, each MMU provides the functionality to assign the motion instruction (*assign_instruction*). Furthermore, the prerequisites being required to start the execution of the given instruction can be validated (*check_prerequisites*), whereas optional boundary constraints are provided. The MMUs generally compute individual postures on a frame basis. Therefore, the interface of the MMUs comprises a *do_step* routine which is executed for each frame to be simulated. In this context, the actual posture at the given frame is computed by the simulation approach contained in the MMU. For each frame, the MMU provides output parameters describing the generated posture, its constraints, intended scene manipulations and events. Since most motion generation approaches strongly rely on spatial information of the environment, the communication with the scene is an important aspect for realizing such an encapsulation. Thus, each MMU can access the information provided by the scene through a defined interface (see Fig. 2 scene access). In this way, the actual scene representation can be embedded in diverse target environments. Moreover, a common set of services (e.g. IK) is accessible from the MMUs (Fig. 2 service access).

3.1.2 Co-simulation

Given distinct MMUs comprising specific approaches, the separately generated postures must be merged and further processed to obtain natural motions. The co-simulation merges and overlaps the postures in each frame, while considering its constraints. Figure 1 illustrates the input and output data of the co-simulation.

Generally, a sequence of instructions with optional constraints is given as input to the co-simulation. The co-simulation computes a feasible motion by executing the inserted sequence, calling the respective MMUs and incorporation of the results. Since the scope of the MMI framework is to combine heterogeneous simulation approaches, the individual MMUs might comprise entirely different skeleton structures and anthropometries. To utilize these heterogeneous results in a common platform, a retargeting to a reference skeleton as well as the

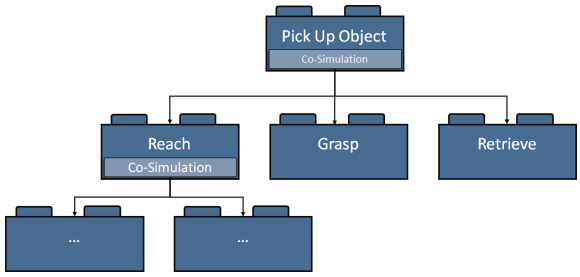

Fig. 3 Illustration of nested MMUs which co-simulate other MMUs by a hierarchical decomposition.

consideration of different anthropometries is required. Moreover, since two consecutive MMUs might start/end with a different posture (e.g. MMU1 ends with t-pose, MMU2 starts with idle pose), the transition between the respective units must be explicitly modeled.

Using a co-simulation, a sequence of MMUs can be principally orchestrated. Since the MMUs are realized in a black box manner, they internally can contain a vastly heterogeneous granularity. Whereas certain MMUs might realize complex motions such as picking up an object internally, others might utilize a subset of finer grained MMUs. In this context, the MMUs act again as a co-simulation. Fig. 3 visualizes the basic concept of nested MMUs based on a pick-up object motion. In this way, a basic set of fundamental motion primitives could be utilized for the aggregation of more complex motions. Given the overall MMU approach, recurrent MMU structures with internal co-simulation are considered as a promising measure to simplify development and increase re-usability of motion primitives.

3.2 Technical framework

For the realization of the aforementioned MMU concept, a technical architecture fulfilling the heterogeneous requirements, is necessary. The detailed specification of the architecture is available at [14]. Figure 4 visualizes the proposed technical architecture for the MMI framework. The system can be subdivided into individual components which are addressed in the following.

Motion Model Units To incorporate heterogeneous motion synthesis approaches, it is indispensable to provide compatibility for many platforms. Therefore, the MMUs can be realized in multiple programming languages, whereas each MMU is accessed by the platform-independent interface (see Figure 2). Each MMU provides a description file in which the motion type, specific parameters and the name of the unit are specified. The MMUs themselves have a programming language specific format: For languages such as C#, or C++, the MMUs are represented as dll.files and can be instantiated during runtime. It is noteworthy, that the co-simulation and MMU have identical interfaces to allow nested MMUs.

Adapters If the MMUs are realized as separate standalone applications, each MMU must implement the communication by itself which induces potential error sources and an implementation overhead. To avoid this, the so-called adapters are proposed. The adapters implement the communication functionality and are responsible for managing the MMUs. In particular, the adapters are provided for each compatible programming language (e.g. C#, C++, Python). The components contain a session handling to allow multiple consumers use the same adapter instance. Moreover, the adapters buffer the scene and provide access to the available services.

Target Engine The target engine acts as central accessing point for the end-users and is responsible for the visualization of the scene and the digital human model. The component provides the ground truth scene and is realized in a specific programming language.

Abstraction Closely linked to the target engine, the abstraction encapsulates the communication to the MMUs. The layer provides accessing functionalities of the MMUs as they were on the local machine. The layer is provided as implementation for each programming language.

Middleware As an essential component for connecting different modules of the framework, the middleware is a crucial aspect. To ease cross-platform implementation Apache Thrift is utilized. The overall communication-formats and services are defined using the Thrift interface description language, whereas the source-code for the different platforms can be automatically generated.

Services To further increase the usability and shrink implementation efforts, functionality which is oftentimes required by MMUs is provided by services. In particular, methods such as path planning, retargeting and collision-detection are available, accessible via middleware.

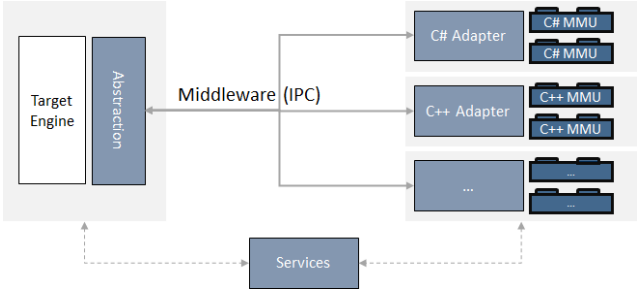

Fig. 4 Technical architecture of the MMI framework. A more detailed specification is available at [14].

4 A Co-Simulation Approach for orchestrating heterogeneous MMUs

In principle, the above described framework allows technically to incorporate heterogeneous character animation systems in a common environment. However, several questions are left open. In particular, it is unclear how the gathered results of different MMUs can be combined to generate feasible postures. Moreover, the handling of concurrent motions using distinct MMUs has not been addressed yet. In the following, we present a novel co-simulation concept which is able to orchestrate various MMUs while producing feasible results. The co-simulation works independently of the utilized animation technology within the respective MMUs. Furthermore, the concurrent behavior of motions is modeled. The initial concept was first presented in [17].

4.1 Restrictions based on the MMI concept

The proposed co-simulation in this work is strongly restricted by the underlying MMI concept and architecture. The main target of the MMI approach is to create a standard for integrating arbitrary technologies in a common framework. Even though this flexibility induces increased manual efforts for the parameterization, the integration of arbitrary technologies is considered as more important than an easy parameterization in rather restricted environments like [29,30]. Therefore, a potential co-simulation must be compatible to any kind of MMU, instead of being optimized to only work with specific approaches.

Due to the targeted genericity and black box realization, the specific content of the MMUs is not known by default. As a consequence, there is no common scheme for defining the motion types available. Thus, each MMU developer can specify custom motion types (e.g. "walk", "sidestep", "run") resulting in ambiguous MMUs. Moreover, the embedded skeleton and manipulated joints might vary strongly within each MMU. This lacking restrictiveness is required to provide a maximum compatibility to different application scenarios and technologies. On the other hand, the omitted availability of semantic motion descriptions strongly influences the design of a potential co-simulation. Consequently, the co-simulation must be able to orchestrate the approaches without explicit semantic knowledge of the contained motions. Therefore, a major difficulty addressed within the novel approach is the coordination of heterogeneous approaches without knowing the specific semantics. The presented co-simulation has been designed with the imposed restrictions in mind, hence differing from other available orchestration approaches.

4.2 Overall concept

Based on manually specified motion instructions such as "walk to table" and "pick up object from table", the co-simulation must incorporate and overlap different humanoid postures generated by the MMUs. In the following, the core concepts of the proposed co-simulation, ranging from hierarchical modeling to constraint handling and the actual workflow are proposed.

4.2.1 Hierarchical MMU modeling

The proposed co-simulation builds upon the concept of hierarchical motion controllers first introduced by Kallmann et al. [21]. As described in [13], the state of the character is manipulated by a series of stacked controllers. The output of the previous controller is set as input of the subsequent one. Fig. 5 visualizes the concept transferred to the MMUs.

Each controller knows the character state of the previous step, as well as the state during the current phase. The controller can either override, modify or ignore the state of the virtual character. In [13], the authors propose to utilize a generalization-specialization hierarchy, which means that lower priority controllers typically control a greater number of body parts, while higher-priority controllers control fewer. In this context, a full body motion (e.g. idle) is executed first, while more specific motions such as grasping are executed later/ with a higher priority. Fig. 6 gives an overview of the newly proposed co-simulation model and its workflow.

4.2.2 Priority Estimation

In the newly proposed co-simulation, each MMU has a specific priority in analogy to the aforementioned concept. The priorities of the respective MMUs are generally

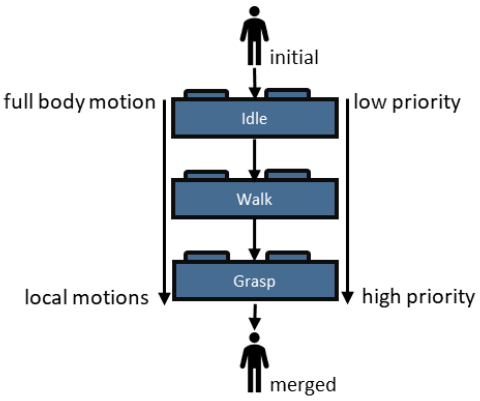

Fig. 5 Concept of hierarchical motion-controllers as proposed by [13] applied to the Motion Model Units.

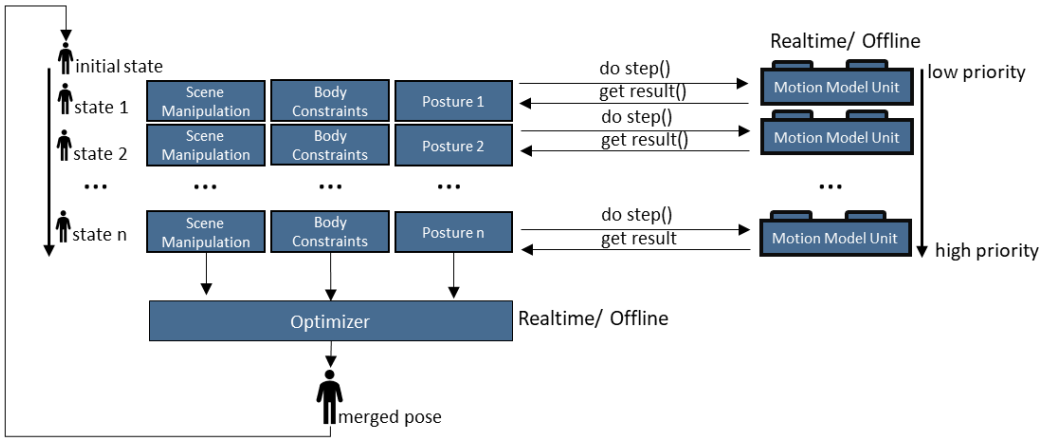

Fig. 6 Illustration of the proposed co-simulation model. The MMUs are executed for each frame according to their priority. The results which comprise the actual character posture, body constraints as well as scene manipulations are stored in a register within the co-simulator. These results are later on merged to a single feasible posture by an optimization algorithm.

assigned based on the characteristics and involved body regions of the given motion instruction (e.g. walking = low priority, grasping = high priority). More complex motion might involve multiple MMUs interacting at the same time-frame. Therewith, the priority assignment depicts a fundamental aspect of the co-simulation.

MMUs can contain entirely different approaches influencing varying body-parts. The priority assignment thus can be only carried out properly if the exact behavior of the MMU is known. Since the co-simulation has no in-build semantic knowledge of the motion contained in the MMUs, the priorities cannot be determined within the co-simulation itself. Therefore, the instructions must be annotated with a priority by the user/superior instance. A generalization of the priority assignment is currently not an intense subject of research since it would rather impose additional restrictions resulting in decreased compatibility of the overall framework.

4.2.3 Constraint de nition

If the above illustrated concept of hierarchical MMUs is strictly applied, the MMUs with higher priorities might completely overwrite the results of the previous ones, thus neglecting relevant criteria of the preceding posture. To prevent this, each MMU can define specific body constraints being essential for preserving the main features of the posture. For instance, an MMU which focuses on locomotion can set the foot and hip position as essential constraints of the posture. On the other hand, a grasp MMU marks the hand position and finger transformations as crucial constraints. The set of available constraint types in the proposed framework is limited to a finite amount (see [14]). In particular, the position and rotation of end-effectors (e.g. hand or foot) or properties such as joint rotations can be specified. The co-simulator stores the constraints of the respective MMUs (see Fig. 6 Body Constraints) for further considerations and processing in subsequent stages.

4.2.4 Co-simulation work ow

In general, the input of the co-simulation is a set of given motion instructions with dependencies between each other (e.g. put-down starts after walk is finished). The co-simulator evaluates the provided conditions for each frame until being fulfilled or aborted. In case of success, the *get_prerequisites* function of the MMU is executed, indicating whether all internal conditions for starting the instruction of the MMU are met. If the MMU returns true, the co-simulator starts the respective instruction via *assign_instruction*. Moreover, boundary constraints of subsequent MMUs (e.g. required start posture) are provided as input along with the motion instruction. After the assignment of the instruction, the started MMU is marked as active. Analogously, termination criteria are also handled by the co-simulation (e.g. terminate MMU2 after 2 seconds). The overall approach of scheduling the different instructions is strongly related to the BML realizer [34] concept.

As illustrated in Fig. 6, in every simulation step, the co-simulator executes each active MMU according to its priority, starting with the lowest. At the beginning of the frame, the initial state is provided as input which corresponds to the merged result of the last frame. Next, the respective MMU is executed by calling the *do_step* function, whereas the computed results of the current frame are obtained. The results comprise the generated posture, body constraints, as well as intended scene manipulations and events occurred during the current frame within the MMU. The MMU can actively modify, set or remove the constraints and scene-manipulations.

Therewith MMUs with higher priorities can always overwrite specified constraints and scene-manipulations by MMUs with less priorities.

All gathered results of the MMUs are stored by the co-simulator and are further integrated into the current state of the character, which is provided as input for the subsequent MMU. The currently available constraints are utilized to generate a state which represents a preview of the constrained posture at the present evaluation stage. In total, there are three different states accessible from the MMU: *initial*, *current state* and *current state constrained*. In particular, it is up to the specific MMU implementation of how to consider and incorporate these states into the respective model (see 4.2.5). However, all constraints, available at the end of the frame are processed by the optimization stage.

Depending on the available MMUs and configurations, the merging and processing of the postures might be already established by the hierarchical execution of the MMUs. However, to generate optimized results fulfilling all constraints and use-case dependent requirements, a separate optimization stage (see Fig. 6 Optimizer) might be necessary. Moreover, the available body constraints defined by the MMUs need to be incorporated into the final posture of the character.

4.2.5 Combination of postures

Different to other available animation frameworks such as [29], the combination of postures is not carried out on a global level. Since the co-simulation has no semantic knowledge of the contained motion, the MMUs internally perform the merging of different postures using a specific merge function f_{merge} (see equation 1). Generally, the MMU have access to the different states: constrained preview (p_{constr}), initial (p_{init}) and current (p_{cur}), which contain the full posture of the avatar. Moreover, a list of constraints (c_{cur}), scene manipulations (s_{cur}) and events (e_{cur}) of the previous MMUs in hierarchy are provided to the MMU as input. As output, the MMUs returns the computed posture (p_{res}), a list of constraints (c_{res}), scene manipulations (s_{res}) and events (e_{res}).

$$f_{merge} : \{p_{init}, p_{constr}, p_{cur}, c_{cur}, s_{cur}, e_{cur}\} \rightarrow \{p_{res}, c_{res}, s_{res}, e_{res}\} \quad (1)$$

The MMUs have full control of the actual merging process. For instance, a grasping MMU can utilize the current posture and just overwrite the upper body joints, while specifying additional constraints to preserve the grasping posture. On the other hand, a purely motion capture based MMU might overwrite the entire postures and constraints.

4.2.6 Posture optimization & constraint solving

The main task of the optimization stage is to apply the body constraints specified by the MMUs, as well as performing additional use-case dependent operations, such as foot-grounding or ergonomic optimizations like proposed in [18]. As outlined in 4.2.4, a set of specified constraints and all generated results by the MMUs are the input of the optimization stage. The constraints are generally limited to finite amount of different types such as inverse kinematics constraints for end-effectors like the hands or specified joint rotations. For applying these constraints, an inverse kinematics solver is mandatory. However, other solvers, such as foot-grounding or inter-frame blending, can be additionally added to the framework. The solvers are executed for every frame and consequently have a significant influence on the generated postures. It is therefore crucial to carefully select appropriate solvers regarding the computational performance and produced results.

4.3 Modeling the transitions between postures

Despite the scheduling and posture merging process, the modeling of the transition between postures of different frames is an essential aspect to obtain feasible motions. If no further transition modeling is applied, unnatural gaps between consecutive frames might occur. In most animation systems, motion blending is commonly used for this purpose. By applying crossfading between different motions, the transitions can be smoothly interpolated. However, in contrast to animation clips, the content of the specific MMU is not known and dynamically generated. Moreover, each MMU might have specific parameters for the transition between different postures. Therefore, a simple, globally performed crossfading is not possible, without possibly violating constraints. To establish a smooth transition between various postures generated by different MMUs, the novel co-simulation approach builds upon two concepts.

First, in the framework, the constraints defined by the MMUs are not actively removed by the co-simulator.

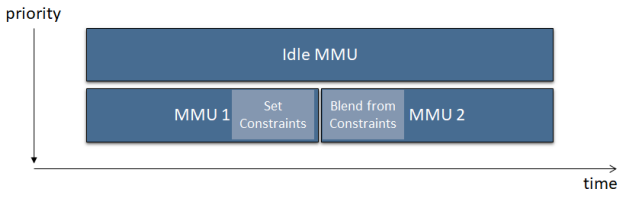

Fig. 7 Illustration of the transition modeling for the first scenario, in which a MMU finishes its motion and actively specifies constraints.

Thus, if an MMU finishes the motion and has end constraints preserving the posture such as "keep hand position", the constraints remain to be considered by the posture optimization and the MMUs until being actively removed or overwritten. Consequently, unnatural transitions with gaps between the postures can be avoided if end constraints are specified. Fig. 7 schematically visualizes this scenario. Since the constraints are still active after the lifetime of the MMU, the constraints must be explicitly considered by subsequent MMUs for the transition modeling. Therewith the subsequent MMU needs to overwrite/remove the constraints and perform a posture blending to generate a smooth motion.

Second, if an MMU finishes its motion and has no active constraints specified, it must ensure that the ending posture matches the posture of the current state, which corresponds to the resulting posture of the previous MMU in hierarchy. The transition modeling is therefore internally performed by the MMU, whereas the specific parameterization and knowledge of the MMU can be used. The process can be principally considered as a distributed modeling of the transitions which contrasts with commonly performed global motion blending. In this way, it is ensured that smooth transitions between the previous MMU in hierarchy and the respective MMU are obtained, after the MMU is finished. As illustrated by Fig. 8 (top), the first MMU blends to the underlying MMU until the motion is finished, whereas the subsequent MMU blends from the underlying state to the internal motion. Some technologies might be not flexible enough to produce postures or motions which can be incorporated in this way. Thus, optionally an end-posture can be defined for the MMU (see Fig. 8 bottom). If a specific end-posture is assigned, the end-posture must be used for blending instead of the posture of the current sate at the present frame.

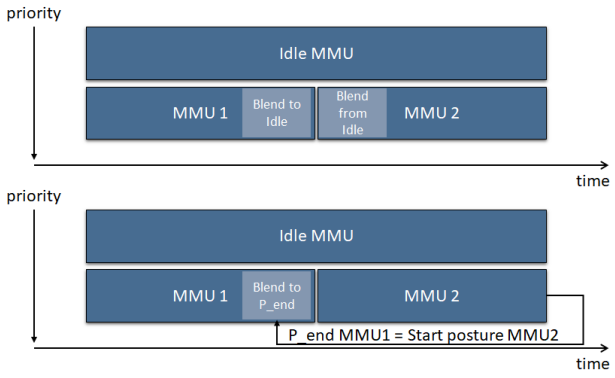

Fig. 8 Illustration of the transition modeling for the second scenario, in which the currently active MMU is responsible for blending to the underlying posture (top) or to a defined end posture (bottom).

4.4 Modeling concurrent motions

Given the presented architecture, it is possible to execute arbitrary MMUs based on their priority and generate a merged posture for each frame. In general, the sequence of motions to be executed must be provided as an input to the co-simulation. Using formats like the Behavior Markup Language (BML) [13], a basic scenario such as walk to, pick-up and put-down can be formulated. In this case, the pick-up motion starts after the walk motion has been finished. Analogously, the put-down motion has the prerequisites, that walk, and pick-up must be finished. With BML, these conditional constraints depending on other instructions can be formulated, whereas it is also possible to model timing constraints. However, given the language it is difficult to express constraints strongly related to the scene context, or which are not known at the time the instructions are created. Since the MMUs might comprise completely different animation technologies, the prerequisites can only be defined by the MMU itself. Therefore, each MMU provides the functionality to check the required prerequisites for executing a specific motion.

Examining humanoid motions, it can be encountered that most of the performed motions are commonly executed in parallel. For instance, a grasp motion might be performed during walking, the specific time and location when the grasping starts, however, strongly depends on spatial constraints and prerequisites of the actual grasp motion. Thus, it is not a straightforward task to define the exact timing and all constraints in before. To nevertheless cover the concurrent modeling in a generic manner within the co-simulation, the *check_prerequisites* method of the MMU interface can be utilized. This method validates whether all constraints are fulfilled in order to start the specific motion. Depending on the implementation and motion, the constraints can address vastly heterogeneous aspects such as the distance to a target object or the maximum velocity of the avatar.

For modeling concurrent behavior, the co-simulation initially checks if all external conditions for starting the MMU are fulfilled. If this is the case, next, the prerequisites of the specific MMU instance are verified. If the prerequisites of the instruction are fulfilled, the respective motion can be started. Applied to the aforementioned walking and grasping example, the grasp motion can be automatically started during walking if the required constraints such as distance to the target object are fulfilled (see Fig. 9). By modeling the runtime specific constraints in this way, the exact timing does not have to be explicitly specified in before. Moreover, each MMU implementation can adjust the constraints dynamically according to the used model.

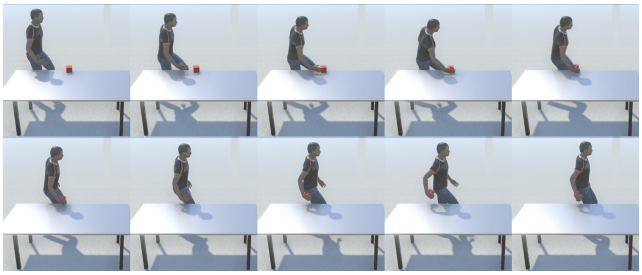

Fig. 9 Exemplary scenario generated by the novel co-simulation. The grasping is automatically started during walking, once the prerequisites of the grasp MMU are fulfilled.

For more complex examples such as feedback loops between gazing and hand movement, grasping and gesturing or lip syncing and emotion, the concurrency modeling can be likewise addressed in the proposed way. The MMUs can principally monitor the state of the avatar/scene and can dynamically react to it by changing the prerequisites and simulation outcome. Therewith, feedback loops between gazing and hand movement can be realized by analyzing the present hand state within the gaze MMU. Given an input sequence with timing restrictions (e.g. gaze starts after 1s), the gaze MMU could evaluate the pointing vector of the respective hand and adjust the gaze target correspondingly.

With growing complexity of the scenarios, the implementation effort and error proneness for the coordination and prerequisite handling generally increases. However, given the MMI approach it is also possible to realize these behaviors at a different granularity. For instance, gazing and hand movement could be realized within a single MMU without requiring external coordination. Furthermore, complex behaviors can be realized within nested MMUs, internally coordinating the other required MMUs (acting as co-simulator). For complex and rather use-case specific interactions, a realization in a dedicated/nested MMU is considered as a good trade-off between implementation effort and usability.

4.5 Conflict handling

Given the framework comprising multiple heterogeneous MMUs, the input must be carefully specified to eliminate potential conflicts and achieve the desired motions. However, in practice, different types of conflicts might occur during the co-simulation. Generally, it is assumed that technical errors are handled by the surrounding MMI framework. Therefore, in the following, only conflicts on a content-related level are discussed.

Behavior plan violation A first potential problem occurs if a given sequence of instructions, also referred as behavior plan, cannot be fulfilled. For instance, an MMU cannot be started due to unfulfilled prerequisites (e.g. waiting for the termination of a previous MMU) or unsupported input. In this case, the overall behavior plan might get stuck or corrupted. The co-simulator propagates the information back, whereas the behavior plan must be adjusted by the superior instance.

Constraint related errors Further potential conflicts are related to constraints being not fulfilled/realizable.

The first group of constraint-related errors occurs if desired constraints cannot be fulfilled given the avatar and scene (e.g. IK is not able to reach desired goal). Generally, the co-simulator solves the constraints in the optimization stage and provides the result as entry state for the next frame. It is up to the specific MMU to evaluate the constraint and react to it.

A second conflict scenario occurs if constraints of an MMU are accidentally manipulated by a subsequent MMU with higher priority. In this context, the constraints can be either actively removed or overwritten by contradicting constraints. For instance, a grasp MMU forces an end-effector constraint of the hand, whereas the subsequent MMU overwrites it. Thus, the constraint of the grasp MMU is never fulfilled, if the subsequent MMU is active. In most cases, this behavior might be desired, since the user specified the hierarchy and sequence. However, the co-simulation cannot distinguish whether the behavior is desired or not since it has no semantic understanding of the behavior plan.

For both cases holds, the MMUs can internally monitor the state changes of the avatar and scene. Moreover, the MMUs internally know the semantics and characteristics of the motion to be simulated best. Therefore, the idea of the co-simulation approach is to perform the constraint-related conflict-management in a distributed manner inside the MMUs. The actual reaction depends on the MMU implementations. For instance, if an MMU encounters that the constraint is ignored, it could provide an exception message to be handled by the superior instance. On the other hand, the MMU could also apply a different policy by waiting until the constraints are applied or providing an alternative motion planning.

Unfeasible postures Further potential outcomes of the co-simulation might be unfeasible postures generated by the MMUs. Generally, the detection and definition of unfeasible posture strongly depends on the use-case (e.g. bio-mechanical correctness vs. naturalness). The optimization stage allows to incorporate several post-processing components and solvers. Therefore, approaches for the detection of unfeasible posture and post-processing methods such as foot-grounding, foot-skating avoidance can be incorporated by the user.

5 Evaluation

After having outlined the novel co-simulation concept and the underlying framework, in this section, the validity of the co-simulation approach is examined in detail. Moreover, the applicability of the approach given different MMU implementations is investigated.

5.1 Validation of the co-simulation approach

Summarizing the role of the co-simulation, the main task is to incorporate different postures obtained from any heterogeneous MMUs, while generating feasible results. To validate the functionality, it is crucial to perform an evaluation in a controlled environment without any further error sources. Therefore, the co-simulation is validated separately using predefined motion capture recordings as ground truth data. In this way, it can be ensured that only the quality of the co-simulation approach itself is measured, instead of the characteristics of the algorithms contained in the MMUs.

5.1.1 Experimental design

As base for the evaluation, the Carnegie-Mellon University Motion Capture (CMU) library [1] has been utilized, which contains in total 2534 different motion capture clips. These clips were temporally and hierarchically segmented into smaller sections and embedded inside MMUs. Overall, it should be examined whether the co-simulation is able to reproduce the original motion based on the segmented motion capture data contained in the MMUs. Fig. 10 illustrates the evaluation concept. In particular, the temporal orchestration of multiple MMUs, as well as the hierarchical orchestration (segmentation based on involved body parts) is analyzed. The similarity is measured in terms of joint rotation difference between the individual postures occurring in the co-simulated and original motion. Generally, the validity of the co-simulation can be approved if no significant error can be encountered. The tested co-simulation was realized in Unity3D, whereas the update rate was set to 100 frames per second.

In practical use, the co-simulation will be utilized to combine heterogeneous character animation approaches. Since these approaches are rather based on dynamic simulation algorithms than just playback of motion-capture data, the contained approaches utilize constraints and model the transitions internally. Both aspects tend to affect the quality of the generated motions. Therefore, a further error estimation for applying the constraints and the transition modeling is carried out.

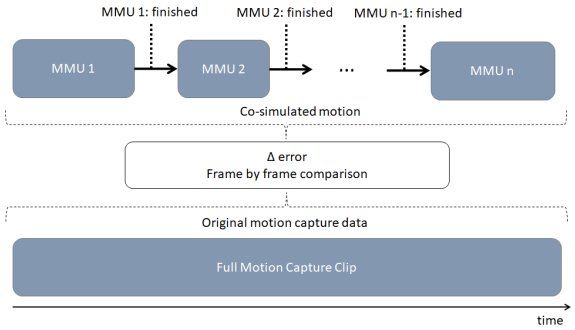

Fig. 10 Sequential evaluation. A motion capture clip is randomly cut into different sub-motions. These motions are embedded within MMUs.

5.1.2 Validity of temporal orchestration

Generally, the co-simulator schedules multiple MMUs based on given instructions and timings such as "start MMU2 after MMU1 is finished". For the validation of this functionality, the CMU library is used, whereas each individual clip is randomly split into multiple segments having a duration $d \in [0.50s; 5.00s]$. For short motion capture clips, it is ensured that at least two different segments are generated. A time frame between $0.50s$ and $5.00s$ is considered as a realistic assumption for common activities contained in MMUs (e.g. grasping, walking). Longer motions ($d > 5.00s$) were excluded since the number of test-data segments would be significantly reduced. Shorter time-frames ($d < 0.50s$) were skipped, because these would rather result in highly split motions with less semantics. In total, all 2535 clips of the CMU library were segmented according to the illustrated principle. A limitation imposed by the randomized splitting is that it is not known, at which state the transition takes place (e.g. during different phases in walk cycle). However, this is not considered as a major drawback since the target of the evaluation is to validate the co-simulation for generic motions.

Each segment is realized as a separate MMU and is started after the previous MMU has been finished. Figure 10 visualizes the decomposition of the overall motion capture clips into individual MMUs.

Results Overall, in every tested scenario, no difference with respect to the joint rotations of the individual postures compared to the original motion capture clip could have been encountered (difference below floating-point accuracy), resulting in a median and mean error of 0.00 degree. Therewith, it can be concluded that the temporal orchestration of the MMUs induces no additional error. Consequently, the validity of the co-simulation for the temporal orchestration of different MMUs is approved for the tested dataset.

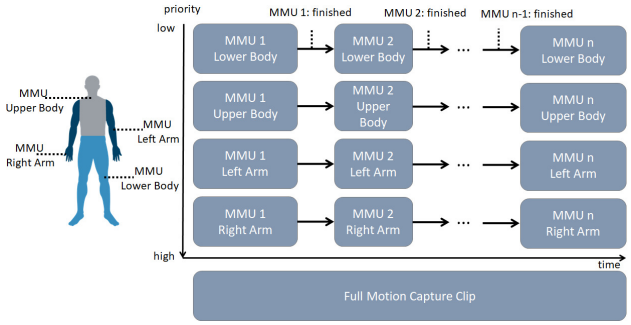

Fig. 11 Hierarchical evaluation. A motion capture clip is randomly cut into different sub-motions. The motions are furthermore split according the involved body parts.

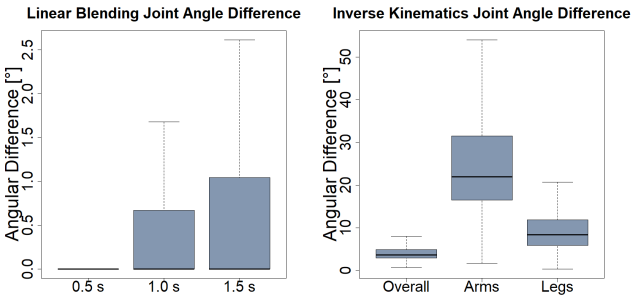

Fig. 12 Results of the evaluation for the inverse kinematics solver and linear blending. The linear blending was tested for three different blending windows ranging from 0.50 s to 1.50 s. The angular differences induced by the IK solver are further split into arms and legs.

5.1.3 Validity of hierarchical orchestration

For validating the hierarchical orchestration, the reference motions (CMU library) are additionally split according to the involved body parts, resulting in a total amount of four MMUs for each temporal segment. Fig. 11 (left) illustrates the MMU assignment based on the body parts. In particular, MMUs for the lower body, upper body and arms have been used. The priorities of the MMUs are constant and increase according to the generalization specialization scheme, as illustrated in Fig. 11. Within each MMU, the body parts which are not directly set by the MMU are modeled using a default idle posture. Generally, the MMUs with higher priorities (e.g. left arm) overwrite the generated postures of the ones with lower priorities (e.g. upper body). In the merging function f_{merge} contained in each MMU, the resulting posture p_{result} is generated as follows:

$$p_{result} = f_{Blend}(p_{n\ \ 1}, p_n, m_n, 1) \tag{2}$$

p_n represents the generated posture of the n^{th} MMU and $p_{n\ \ 1}$ of the previous MMU in hierarchy, whereas m_n is the blending mask of the n^{th} MMU (e.g. setting upper body joints to 1). f_{Blend} describes a linear blending function between two postures, with a blending mask and a specified weight.

Analogously to the previous evaluation, a limitation imposed by the segmentation is that the specific phase of the motion might vary. However, within the evaluation the validity of the hierarchical merging is focused which requires no semantic knowledge of the motions.

Results Similar to the temporal orchestration, the obtained motions by performing a hierarchical co-simulation showed no measurable difference compared to the original motion capture data, thus resulting in a mean and median error of 0.00 . Therefore it can be approved, that the co-simulation is able to orchestrate multiple MMUs without any further induced systematic errors.

5.1.4 Error estimation - transition modeling & constraint solving

As outlined in section 4, the transition modeling between consecutive MMUs and utilization of body constraints are essential aspects of the co-simulation. If the MMU uses constraints and performs no further transition modeling, the quality of the generated posture depends on the IK solving algorithm contained in the co-simulation. On the other hand, if the transitions are explicitly modelled internally by the MMU, the results strongly rely on the utilized blending approach. Therefore, in the following, an error estimation for a state-of-the-art IK solver and linear posture blending is carried out.

Inverse Kinematics constraints To analyze the influence of the constraint utilization on the generated results, inverse kinematics constraints are set for each frame by the MMUs. In particular, each MMU sets a constraint for a specific bone (end-effector position and rotation) based on the current posture of the motion capture clip. Whereas, the lower body MMU sets the feet and hip constraints, the left arm MMU sets the left-hand posture, same applies for the right arm. For the evaluation, the FABRIK implementation of [3] and the clips contained in the CMU database have been used. Overall, the generated postures utilizing the IK solver are compared to the original motion capture data.

Analyzing the gathered results (see Fig. 12), it can be denoted that the use of inverse kinematic constraints evidently influences the results in terms of joint rotation difference. The differences between the solved and original postures deviate from frame to frame. Overall, a mean (μ) of 4.45 , median (θ) 3.73 and standard deviation (σ) of 2.51 can be denoted. In particular, the joint angle differences of the arm chains are considerably higher with $\mu = 28.13$, $\theta = 21.95$ and $\sigma = 20.41$. The angle differences of the legs are slightly reduced

with $\mu = 9.96$, $\theta = 8.40$ and $\sigma = 7.28$. The affected joint values for the arms chains led in several cases to self-collisions being visible in the generated motions.

Summarizing the findings, it can be concluded that the IK solver in the optimization stage of the co-simulation strongly affects the overall postures. It is therefore crucial to select an IK solver minimizing the differences.

Posture blending Besides constraint solving, posture blending is a further aspect influencing the results of the co-simulation. For the evaluation, linear blending was used to model the transitions between consecutive MMUs. Here-again the CMU database served as ground truth data. Similarly to the previous validation, the differences between the co-simulated results with posture blending and the original motion are analyzed. For the validation, each MMU gets the starting posture of the subsequent MMU as boundary constraint. The MMUs internally use linear blending to match the posture. The blend time was set to $0.50s$, $1.00s$ and $1.50s$.

Analyzing the results, it can be concluded, that linear blending between MMUs affects the motion quality. As indicated by Fig. 12, the mean angular difference increases with a growing blend window. With a blending duration of 0.50 s, a median value of 0.00 , $\mu = 0.17$ and σ of 0.52 can be denoted. Moreover, a blend duration of 1.00 s resulted in $\mu = 0.47$, $\theta = 0.00$ and $\sigma = 1.02$. A longer blend window of 1.50 s led to $\mu = 0.80$, $\theta = 0.00$ and σ of 1.41 . By visually inspecting the generated motions, especially for long time frames, foot-sliding was encountered. Generally, these artifacts increase with growing blending duration. For optimal co-simulation results, it is therefore crucial to utilize appropriate interpolation approaches for blending to specified end-postures. However, the performed evaluation already indicates, that especially for the upper body smooth transitions can be obtained.

5.1.5 Discussion

Considering the individually examined properties of the co-simulation, the overall validity can be approved. For the tested motion-capture scenario, the co-simulation is able to reconstruct the original data without any measurable error. However, in practice, the MMUs will not be fed with motion capture clips, instead simulation algorithms will be contained. Therefore, the use of constraints and linear blending for modeling the transitions is an important aspect. As outlined by the validation, the IK solver and the linear blending affect the results. For further validation of simulation algorithms within MMUs this should be taken into account. If the transition modeling is done by MMUs internally, the overall error depends on the respective MMU implementation. If the constraints are handled by the co-simulation, the accuracy is currently limited by the IK solver. Transferring the results to practical use-cases, this means that the applied IK solver must be carefully chosen to minimize potential errors and self-collisions. For the transition modeling, linear blending should be preferably applied for short time windows only, or explicit measures such as foot-skating avoidance should be utilized.

5.2 Validation using an exemplary implementation

Even though the isolated evaluation approved the validity of the co-simulation, the practical applicability of the approach must be further investigated. For this purpose, the system was tested using an exemplary implementation incorporating different animation technologies.

5.2.1 Experimental design

Similar to the evaluation in 5.1, the validity of the co-simulation was investigated, however, instead of using pre-recorded motion capture data, different character animation algorithms were used. Therewith, a different experimental design must be chosen. The overall evaluation was built upon a scenario containing in total ten individual sub-tasks, each being modeled by a separate MMU (see Fig. 13). For the validation, three different configurations were tested. Within configuration A (concurrent), the individual tasks were handled in a concurrent way (hierarchical modeling) using the proposed co-simulator. In scenario B (sequential), the novel co-simulation was utilized, whereas the sub-tasks were modeled strictly sequentially (temporal modeling) with transition modeling. Within scenario C (no co-simulation), the MMUs were sequentially executed without any co-simulation. Overall, the applicability of the co-simulation with different character animation technologies was investigated. For measuring the quality of the generated motions, a user study was carried out. In particular, the co-simulated sub-motions and single motions generated by the MMUs were compared and rated by the participants. The participants rated the naturalness in a pairwise comparison (e.g. AB, AC, BC) and chose the preferred motion or rated both as equal. Furthermore, the overall clip containing the co-simulated sequence of the comprehensive scenario was compared to the MMUs sequence without applied co-simulation. The validity of the co-simulation can be principally approved if the quality of the individual clips is not decreased compared to the original motions contained in the MMUs. Moreover, the co-simulated sequence must be rated better than the sequence without applied co-simulation.

5.3 Apparatus

The utilized co-simulation was implemented in the Unity engine and comprised multiple heterogeneous MMUs. In particular, MMUs for idle, walk, grasp, carry, position and release were incorporated. The idle MMU is based on the Unity Mecanim animation system, looping an idle animation. For the walk MMU, two different implementations were used: The first implementation is based on the recent publication of [19] which models the locomotion behavior using deep neural networks (Table 1 AI). The second implementation uses the Mecanim animation system of Unity, a set of motion capture clips and motion blending to generate walk motions (Table 1 Blending). The grasp MMU is realized by a data-driven statistical motion synthesis approach in Python (Table 1 Statistical). The release and position MMUs are based on a model based algorithm utilizing path planning and full body inverse kinematics (Table 1 Model-based). For the concurrent co-simulation, the MMUs for grasp and position (put-down) have the prerequisites that the root position of the avatar must be in range $[0.2m; 1.2m]$ to start the motion. The MMUs internally model the transition by using linear blending with a time frame of $0.50s$. Moreover, a full body inverse kinematics solver [3] was used to apply the body constraints. All external MMUs were accessed via the proposed MMI framework using the defined MMU interface and Apache Thrift. The time accuracy of the simulation was set to $10ms$. All skeletons contained within the MMU were re-targeted to the Unity Mecanim skeleton using the retargeting functionality of the Unity engine. The priorities of the MMUs have been set according to the generalization-specification approach proposed in [13]. As illustrated in Figure 13, the experimental scene contained three main interaction points (1-3). In total, ten sub-tasks were modeled by the different MMUs. Table 1 gives a detailed overview of the individual tasks and the utilized simulation techniques.

Technical setup process Within the utilized test environment, MMUs originating from different programming languages are used. Given the proposed framework, the setup process of the respective MMUs is similar across the different platforms. First, the MMU project is set up using the desired language (currently supported: C#, C++, Java, Python). For each language, the respective MMU interface and base classes are provided within a dedicated software library. The actual motion synthesis implementation is contained and executed within the MMU class. However, external code can be called from the MMUs as well (e.g. model-based MMU uses external library for path planning). For the incorporated approaches within this work, the main difficulty and effort was the adaption of the utilized models to the exchange formats and mechanisms of the MMI framework. In particular, it is important to ensure that the embedded methods allow accessing the joint values of the skeleton on a frame-based level. For reactive simulations, it is moreover important to set the model's joint values. Although the core part of the motion synthesis technologies can be reused, the functionality for converting, accessing and setting the embedded model needs to be manually implemented in the MMU. Moreover, events, constraints and scene-manipulations provided by the MMUs must be additionally specified. It is important to note, that the MMUs must be aware that the transmitted skeleton structure might differ from the internally used one. Thus, the MMU might require a retargeting from/to the internal skeleton using the available retargeting service. The MMUs are deployed as artifacts comprising the MMU binary, a description file and optional dependencies.

Table 1 Table illustrating the simulated task-sequence, as well as the utilized simulation techniques of the MMUs.

Task	Technique
Walk from start to 1	AI
Grasp large object	Statistical
Walk to 2 while carrying large object	AI
Put down large object on table	Model-based
Release large object	Model-based
Walk to small object	Blending
Grasp small object	Statistical
Walk to 3 while carrying small object	Blending
Grasp large object (support grasp)	Statistical
Position drill on large object	Model-based

For setting up the scene in the target engine, first all relevant objects within the scene are marked as scene objects, being actively synchronized within the framework. Next, connections to the available Adapters are established and the desired MMUs are instantiated based on their description. Afterwards, the co-simulation is setup with the selected MMU instances and priorities. The individual tasks are defined based on BML sequences and are assigned to the co-simulation.

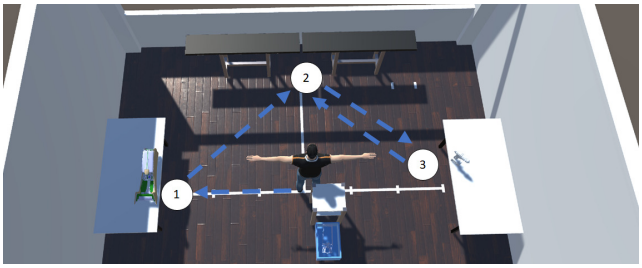

Fig. 13 Visualization of the utilized example scene. The environment comprises in total three interaction points.

5.4 Procedure

For validating the quality of the co-simulation, a user study was carried out in which the participants select the preferred motion in a pairwise video comparison. The order of the videos was randomized. In total 30 different pairwise comparisons were shown to the participants for the sub-motions and three pairwise comparisons for the full motion sequence. At the start of the user-study, the procedure of the survey was explained, and the participants had the possibility to do a test question. Afterwards, a form with the age and gender was filled out. Then, the 30 pairs, each depicting the same sub-motions were presented to the participants in a random order. The participants could watch the video multiple times before selecting the answer. Finally, the full clips were also randomly shown to the participants. For the sub-motions the participants were asked to select which video is better in terms of naturalness. A neutral selection (identical) was also possible. For the full clips, the participant rated the naturalness of the overall motion.

5.5 Results

Overall, the survey was conducted with 20 participants (5 females, 15 males, age: $\mu = 29.10$, $\sigma = 8.35$). The results for the sub-motions and the full motion sequence are separately described within the following.

Sub-motions Fig. 14 visualizes the results for the sub-motions generated by the MMUs. In particular, each whisker plot aggregates the results of the ten sub-tasks as illustrated in table 1. Comparing the individual motions of the sequential and no applied co-simulation, in $\mu = 47.37\%$ of the cases both motions were rated as identical ($\theta = 50.00\%$, $\sigma = 21.73\%$). In $\mu = 42.35\%$ of the cases, the motion of the sequential co-simulation was considered as more natural ($\theta = 40.00\%$, $\sigma = 15.86\%$). In $\mu = 11.58\%$ cases, the original motion was regarded as more natural ($\theta = 10.00\%$, $\sigma = 10.40\%$). For the concurrent co-simulation, $\mu = 48.95\%$ ($\theta = 50.00\%$, $\sigma = 7.28\%$) rated it better compared to the original, $\mu = 33.68\%$ ($\theta = 30.00\%$, $\sigma = 22.30$) as equal and $\mu = 17.37\%$, $\theta = 20.00\%$, $\sigma = 11.62\%$ preferred the original motion. Comparing the sequential and concurrent co-simulation, it can be encountered that the majority rated both approaches equal with respect to the naturalness of the sub-motions ($\mu = 52.63\%$, $\theta = 50.00\%$, $\sigma = 14.81\%$). The concurrent sub-motions were preferred in 31.10% ($\theta = 30.00\%$, $\sigma = 14.83\%$) of the cases, whereas the sequential ones were preferred by $\mu = 16.31\%$ ($\theta = 20.00\%$, $\sigma = 13.50\%$) of the participants.

Overall, aggregating the results of all three independent comparisons, the most frequent answer was “equal” (44.60%). The concurrent co-simulation was selected in 26.67% of the cases, followed by the sequential (19.12%) and no applied co-simulation (9.65%).

Full motion sequence With regard to the full motion sequences, the results strengthen the tendency that the co-simulated results are perceived as more natural (see Fig. 15). Generally, the co-simulated approaches achieved better scores than the non-co-simulated motions. Comparing the sequential co-simulation with no applied co-simulation, the sequential approach received a mean of 84.21% of the votes ($\theta = 100.00\%$, $\sigma = 36.46\%$). $\mu = 17.65\%$ rated both approaches as equal, whereas zero persons preferred the sequence with no co-simulation applied. Analyzing the results for the comparison of the concurrent co-simulation and no applied co-simulation, the discrepancy between the co-simulated and non-co-simulated motion further increases. The concurrent motions were chosen in 89.247% ($\theta = 100.00\%$, $\sigma = 30.69\%$), whereas a mean of 10.52% of the participants rated both as equal ($\theta = 0.00\%$, $\sigma = 30.69\%$). No participant preferred the version with no co-simulation. Comparing the sequential to the concurrent co-simulation, most of the participants preferred the concurrent version ($\mu = 73.68\%$, $\theta = 100.00\%$, $\sigma = 44.03\%$), whereas 21.05% ($\theta = 0.00\%$, $\sigma = 40.79\%$) rated both as equal and 5.26% ($\theta = 0.00\%$, $\sigma = 22.33\%$) preferred the sequential one.

Combining the results of the three independent comparisons, the concurrent co-simulation achieved the highest overall rating of 50.88%, whereas the sequential co-simulation achieved 29.82%. No clear tendency (equal) was selected by 19.30%. The full clip with no applied co-simulation was selected by no participant at all.

5.6 Discussion

Analyzing the gathered results for the individual sub-motions, the co-simulated motions achieved constantly a higher percentage compared to the non-co-simulation motions. For the sequential co-simulation, less differences can be encountered, since the majority considered both approaches as equal. In the trend, the concurrent modeling achieved higher results than the sequential co-simulation if compared to the non-co-simulated version. Possible reasons for this are that the tasks itself are strongly affected by the concurrent modeling. For instance, the walk motion in the concurrent scenario additionally contains a grasp motion, since the grasp MMU is automatically started if the avatar is in range of the target object. According to the results, the participants

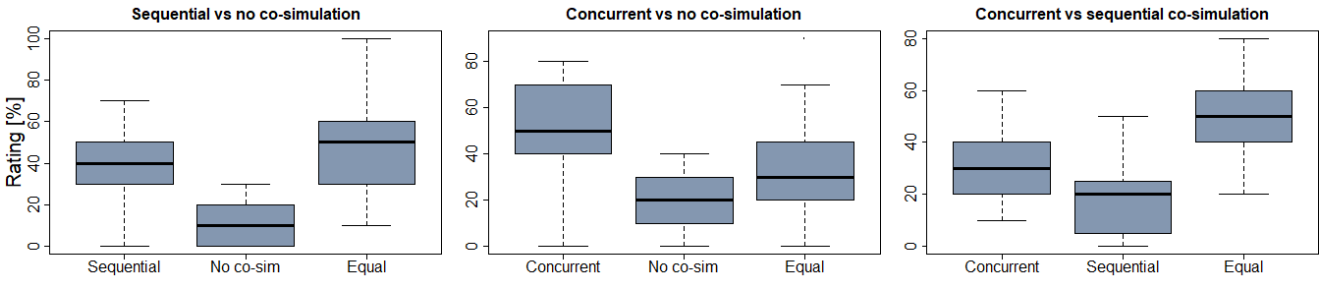

Fig. 14 Results of the performed survey for the naturalness rating of the individual motions.

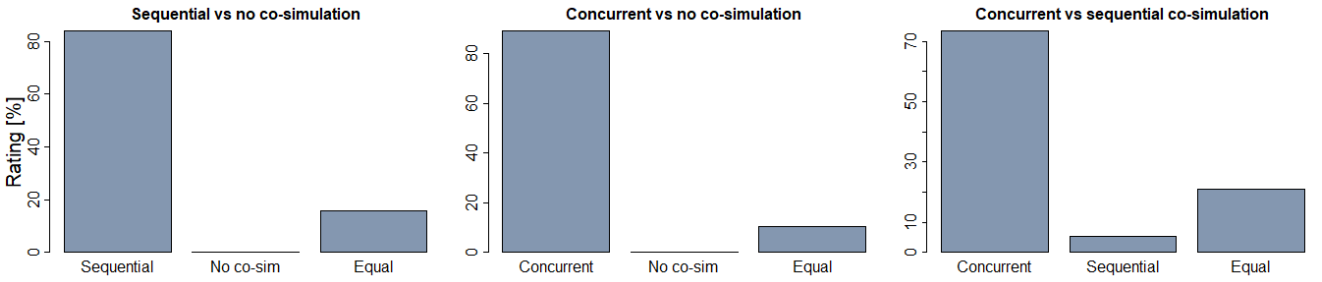

Fig. 15 Results of the performed survey for the naturalness rating of the full motions sequences. The bar charts visualize the mean percentage for each comparison and answer possibility.

perceived those motions as more natural. The difference between the sequential and non-co-simulated motions can be traced back to the different starting/ending postures if no co-simulation is applied. Given the obtained results, it can be summarized that the quality of the individual motions is not decreased by the co-simulation.

Examining the results of the full motion-sequence, it can be denoted that the full motion is rated consistently better compared to the concatenated MMUs without any co-simulation. As outlined by the gathered data, the non-co-simulated motion was preferred by no participant at all, whereas most participants preferred the co-simulated versions in both cases. These results approve the validity of the co-simulation for the given motion sequence. Comparing the concurrent and sequential co-simulation, it can be uncovered that the majority preferred the concurrent co-simulation. Possible reasons are, that the concurrent motions were perceived as more natural since human motions comprise concurrent behavior to a large extent. In contrast purely, sequential motions are performed rather seldom.

Summarizing the findings, the performed user study shows, that the novel co-simulation approach can be applied to generate feasible human motion based on distinct MMUs. In particular, the co-simulation is able to handle the execution of concurrent motions which especially leads to increased naturalness scores. From a use-case perspective, this means that the co-simulation and the underlying framework can be principally used to combine and benchmark multiple approaches, which depicts a major step towards an open modular framework for digital human simulations.

6 Conclusion and Outlook

Within the paper, a novel co-simulation for orchestrating different character animation systems is presented. The validity of the novel concept was evaluated by two different evaluations. First, the overall applicability was examined using motion capture data. Second, a user study was conducted investigating the naturalness of the generated motions. Overall, the novel co-simulation approach can preserve the quality of the original motions contained within the MMUs, while generating feasible results. Therefore, the proposed co-simulation and the framework can be utilized to incorporate heterogeneous simulation approaches in future.

Limitations and future work Even though the generic problem of orchestrating heterogeneous MMUs is addressed by the novel approach, there are still several limitations. Currently, the priorities of the MMUs must be manually assigned. If a system comprises multiple MMUs being active at the same time, the priority assignment might depict a complex and error-prone process. A possible overcome is to integrate more functionality into a single or nested MMU, thus avoiding external coordination of complex sequences. To simplify the coordination,

on the other hand, a scheme/priority assignment could be defined for a subset of compatible MMUs as an extension to the proposed co-simulation. A generic scheme for arbitrary MMU and motions, however, is currently considered as rather unfeasible, imposing additional restrictions on the framework and reducing compatibility.

Given the proposed co-simulation, multiple MMUs can be combined in real-time. However, since the input of each MMU contains the result of the previous MMU, there is a strong sequential dependency. Scaling up the amount of MMUs therefore leads to a performance bottleneck, since each MMU must wait for the result of the previous MMU in hierarchy. The available frame time for each MMU is therefore reduced with each additional MMU. To nevertheless allow real-time systems incorporating a large amount of MMUs, a parallel co-simulation in which the input state is predicted is a possible solution. In this context, the future state of an MMU could be either predicted, by the MMU itself or externally via approaches like Kalman filters.

Despite the discussed limitations, there are further consecutive topics which can be addressed in future work. In this context, novel posture interpolation and improved inverse kinematics approaches should be examined. In addition, aspects like the modeling of the influence of previous and subsequent actions on the current motions can be analyzed. For instance, it is expected, that the specific parameterization of a put-down motion is strongly influenced by the previous pick-up motion. Furthermore, building upon heterogeneous simulations embedded in MMUs, Monte-Carlo simulations which vary the input parameters could be investigated. Finally, the MMU concept will be further developed in the international ITEA 3 research project MOSIM [20] and provided as open source standard after the project.

Acknowledgements The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany within the MOSIM project [20] (grant number 01IS18060A-H).

References

1. CMU Graphics Lab Motion Capture Database. http://mocap.cs. cmu.edu/
2. 07006 ITEA Project MODELISAR - website (2017). URL www.itea3.org/project/modelisar.html
3. Rootmotion final ik - website (2019). URL http://www.root-motion.com/final-ik
4. Aristidou, A., Lasenby, J.: Fabrik: A fast, iterative solver for the inverse kinematics problem. Graphical Models **73**(5), 243–260 (2011)
5. Bastian, J., Clauß, C., Wolf, S., Schneider, P.: Master for co-simulation using fmi. In: Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany, 63, pp. 115–120. Linköping University Electronic Press (2011)
6. Blochwitz, T., Otter, M., Akesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D.: Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In: Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany, pp. 173–184. Linkping University Electronic Press (2012)
7. Buss, S.R.: Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation **17**(1-19), 16 (2004)
8. Cerekovic, A., Pejsa, T., Pandzic, I.S.: Realactor: character animation and multimodal behavior realization system. In: International Workshop on Intelligent Virtual Agents, pp. 486–487. Springer (2009)
9. Clavet, S.: Motion matching and the road to next-gen animation. URL https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road
10. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelen, D.G.: Opensim: open-source software to create and analyze dynamic simulations of movement. IEEE transactions on biomedical engineering **54**(11), 1940–1950 (2007)
11. Faloutsos, P., Van de Panne, M., Terzopoulos, D.: Composable controllers for physics-based character animation. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 251–260. ACM (2001)
12. Feng, A., Huang, Y., Kallmann, M., Shapiro, A.: An analysis of motion blending techniques. In: International Conference on Motion in Games, pp. 232–243. Springer (2012)
13. Feng, A.W., Xu, Y., Shapiro, A.: An example-based motion synthesis technique for locomotion and object manipulation. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 95–102. ACM (2012)
14. Gaisbauer, F.: Mosim research project deliverable 2.2: Mmu concept and interface specification (2019). URL https://itea3.org/project/mosim.html
15. Gaisbauer, F., Agethen, P., Br, T., Rukzio, E.: Introducing a Modular Concept for Exchanging Character Animation Approaches. In: E. Jain, J. Kosinka (eds.) EG 2018 - Posters. The Eurographics Association (2018)
16. Gaisbauer, F., Agethen, P., Otto, M., Bär, T., Sues, J., Rukzio, E.: Presenting a modular framework for a holistic simulation of manual assembly tasks. Procedia CIRP **72**, 768–773 (2018)
17. Gaisbauer., F., Lehwald., J., Agethen., P., Sues., J., Rukzio., E.: Proposing a co-simulation model for coupling heterogeneous character animation systems. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP,, pp. 65–76. INSTICC, SciTePress (2019)
18. Hanson, L., Högberg, D., Carlson, J.S., Bohlin, R., Brolin, E., Delfs, N., Mårdberg, P., Stefan, G., Keyvani, A., Rhen, I.M.: Imma–intelligently moving manikins in automotive applications. In: Third International Summit on Human Simulation (ISHS2014) (2014)
19. Holden, D., Komura, T., Saito, J.: Phase-functioned neural networks for character control. ACM Transactions on Graphics (TOG) **36**(4), 42 (2017)

20. ITEA: ITEA Project 17028 MOSIM - website (2018). URL https://itea3.org/project/mosim.html
21. Kallmann, M., Marsella, S.: Hierarchical motion controllers for real-time autonomous virtual humans. In: International Workshop on Intelligent Virtual Agents, pp. 253–265. Springer (2005)
22. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: ACM SIGGRAPH 2008 classes, p. 51. ACM (2008)
23. Lee, Y., Wampler, K., Bernstein, G., Popović, J., Popović, Z.: Motion fields for interactive character locomotion. In: ACM Transactions on Graphics (TOG), vol. 29, p. 138. ACM (2010)
24. Li, Z., Zhou, Y., Xiao, S., He, C., Li, H.: Auto-Conditioned LSTM Network for Extended Complex Human Motion Synthesis. arXiv preprint arXiv:1707.05363 (2017)
25. Min, J., Chai, J.: Motion graphs++: a compact generative model for semantic motion analysis and synthesis. ACM Transactions on Graphics (TOG) **31**(6), 153 (2012)
26. Müller, B., Wolf, S., Brueggemann, G., Deng, Z., Miller, F., Selbie, W.: Handbook of Human Motion. Springer International Publishing (2018). URL https://books.google.de/books?id=9CmsAQAACAAJ
27. Reed, M.P., Faraway, J., Chaffin, D.B., Martin, B.J.: The humosim ergonomics framework: A new approach to digital human simulation for ergonomic analysis. In: SAE Technical Paper Series, SAE Technical Paper Series. SAE International400 Commonwealth Drive, Warrendale, PA, United States (2006). DOI 10.4271/2006-01-2365
28. Shapiro, A.: Building a character animation system. In: International Conference on Motion in Games, pp. 98–109. Springer (2011)
29. Shoulson, A., Marshak, N., Kapadia, M., Badler, N.I.: Adapt: the agent developmentand prototyping testbed. IEEE Transactions on Visualization and Computer Graphics **20**(7), 1035–1047 (2014)
30. Thiebaux, M., Marsella, S., Marshall, A.N., Kallmann, M.: Smartbody: Behavior realization for embodied conversational agents. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1, pp. 151–158. International Foundation for Autonomous Agents and Multiagent Systems (2008)
31. Tsai, Y.Y., Lin, W.C., Cheng, K.B., Lee, J., Lee, T.Y.: Real-time physics-based 3d biped character animation using an inverted pendulum model. IEEE transactions on visualization and computer graphics **16**(2), 325–337 (2010)
32. Van Acker, B., Denil, J., Vangheluwe, H., De Meulenaere, P.: Generation of an optimised master algorithm for fmi co-simulation. In: Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, pp. 205–212. Society for Computer Simulation International (2015)
33. Wang, B., Baras, J.S.: Hybridsim: A modeling and co-simulation toolchain for cyber-physical systems. In: Proceedings of the 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications, pp. 33–40. IEEE Computer Society (2013)
34. Welbergen, H., Reidsma, D.: A bml realizer for continuous, multimodal interaction with a virtual human (2010)