Virtualization System for Life Science Automation Laboratory

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

Submitted by:
Yanfei Li
from Rostock, 2014
born on 25th, August 1981 in Longyou, China

urn:nbn:de:gbv:28-diss2015-0218-3

Gutachter:

1. Gutachter:

   Prof. Dr.-Ing. habil. Kerstin Thurow,
   Institut für Automatisierungstechnik, Universität Rostock, Germany

2. Gutachter:

   Prof. Dr. Hartmut Pfüller,
   Institut für Angewandte Mikroelektronik und Da-tentechnik,
   Universität Rostock, Germany

3. Gutachter:

   Professor David Kaber,
   Department of Industrial and Systems Engineering,
   North Carolina State University, USA

Datum der Einreichung: 05. February 2014
Datum der Verteidigung: 28. May 2014

Acknowledgement:

There are many thanks to several persons as followings in this doctoral dissertation. Without them, the dissertation would not be completed.

First of all, I would like to express my pretty gratitude to Prof. Thurow and Prof. Stoll for providing such an interesting topic, and for giving me so much guidance, supports and helps in the whole dissertation process. They are always nice, tolerant, and patient to me when I met difficulties or had halts in the work. Without their continuous motivation and trusts, it's impossible for me to insist on and finish this dissertation finally.

Besides, many thanks I would like to say to my colleagues of the Institute of Automation and the Center for Life Science Automation (celisca). They are always so nice to supply helps and supports when I met problems no matter in work or in daily life. Especially I'd like to thank Dr. Steffen Junginger, Dr. Thomas Roddelkopf, Mr. Lars Woinar and Dr. Hui Liu for their professional guidance and helps in my work. Also sincere thanks go to Mrs. Anett Ahrens, Mr. Peter Passow and Ms. Ricarda Lehmann for their helps in work affairs and daily life.

Specially, I would like to express my pretty appreciations to Dr. Steffen Junginger and his family for all helps and concerns for my family. Their kindness and sincerity give us the feeling of family members.

Last but not least, I would like to express my heartfelt gratitude and love to my family -- my husband Dr. Hui Liu, my little boy Mr. Xing Liu, my parents Mr. Meiyou Li and Mrs. Zhangwen Liu, and my sister Yanyu Li for their selfless love, support and encouragement. They are always my motivations in my life.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

3D	Three-dimensional
4D	Four-dimensional
VS	Virtualization System
ALP	Automated Labware Positioner
API/APIs	Application Program Interface/ Application Program Interfaces
CAD	Computer-aided Design
CAE	Computer-aided Engineering
celisca	Center for Life Science Automation
COM	Component Object Model
COMS	Complementary Metal Oxide Semiconductor
CS	Control System
CV	Control Virtualization
DDE	Dynamic Data Exchange
DOF	Degree of Freedom
DTS	Data Transfer System
GUI	Graphical User Interface
HIL	Hardware-in-the-loop
IP	Internet Protocol
IPC	Inter Process Communication
LIN	Liner
LSA	Life Science Automation
OLE	Object Linking and Embedding
OCX	OLE Control Extension
PCS	Process Control System
PTP	Point to Point
RSL	Resource Specification Language

RW	Realistic Workstation
SAMI EX	SAMI Workstation EX Software
SDK	Software Development Kit
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol / Internet Protocol
US	Unvisited Set
VM	Virtualization Module
VR	Virtual Reality
WBS	Work Breakdown Structure
Wi-Fi	Wireless Fidelity
WMS	Workflow Management System
XML	Extensible Markup Language

Chapter 1 Introduction

1.1 Background of this Dissertation

Nowadays, highly developed automation improves the efficiency and accuracy for industrial productions and science experiments in Life Science Automation (LSA) [1]–[11]. It brings great convenience for scientists and engineers. Especially with Process Control System (PCS) developing, scientists get relief from heavy experiment works. They just need to design and schedule the workflows of experiments in PCS, and then the PCS will realize them by controlling the workstations [12]–[17].

However, when PCS drives automation devices working, the users have to stay at the laboratory to avoid the workflow wrong. In addition, all works about design testing, workstation display, and laboratory showing have to be done by PCS and the platforms set only in laboratory, other than in office or meeting rooms. That greatly limits researchers' works, especially when they do reports or show their automated workstations to customers out of the laboratories. The limitation makes it impossible to give any vivid demonstration. At the same time, running automation workstation frequently for testing designs induces high cost and some waste for human and materials resources. Additionally, automation laboratories have limited free area. If there are too many visitors, most of them could just watch a part of the experiment workflow, which makes them no comprehensive cognitive for the automation workstation.

These problems of PCS are also occurred in the laboratories of Center for Life Science Automation (celisca). There are many automation workstations which are also driven by some PCS, which are shown as Fig. 1.1 ~ Fig. 1.4. For example, Figure 1.1 shows an automated workstation composed of many automated devices for life science assays, including robot Motoman HP3JC, BiomekFX, BiomekNX-Span8, CytomatHotel, Cytomat6001, PHERAstar, Fluostar, ELx405, SIGMA, etc. All these automated devices are set in the Biomek workstation, and driven by PCS SAMI EX.

To solve the same problems as general PCS, the dissertation presents to make the experiment workflows virtualization in real-time for LSA workstations at celisca. The research result of the dissertation should make the experiment workflow virtualization in screen synchronously with the workflow data generated, and integrate the

virtualization steps as a whole experiment process, which could be shown wherever and whenever. It should also work in a flexible and controllable way, which reacts onto the control information for the workflow.

Figure 1.1: Motoman system at celisca

Figure 1.2: Reactor System at celisca

Figure 1.3: Zymark System at celisca

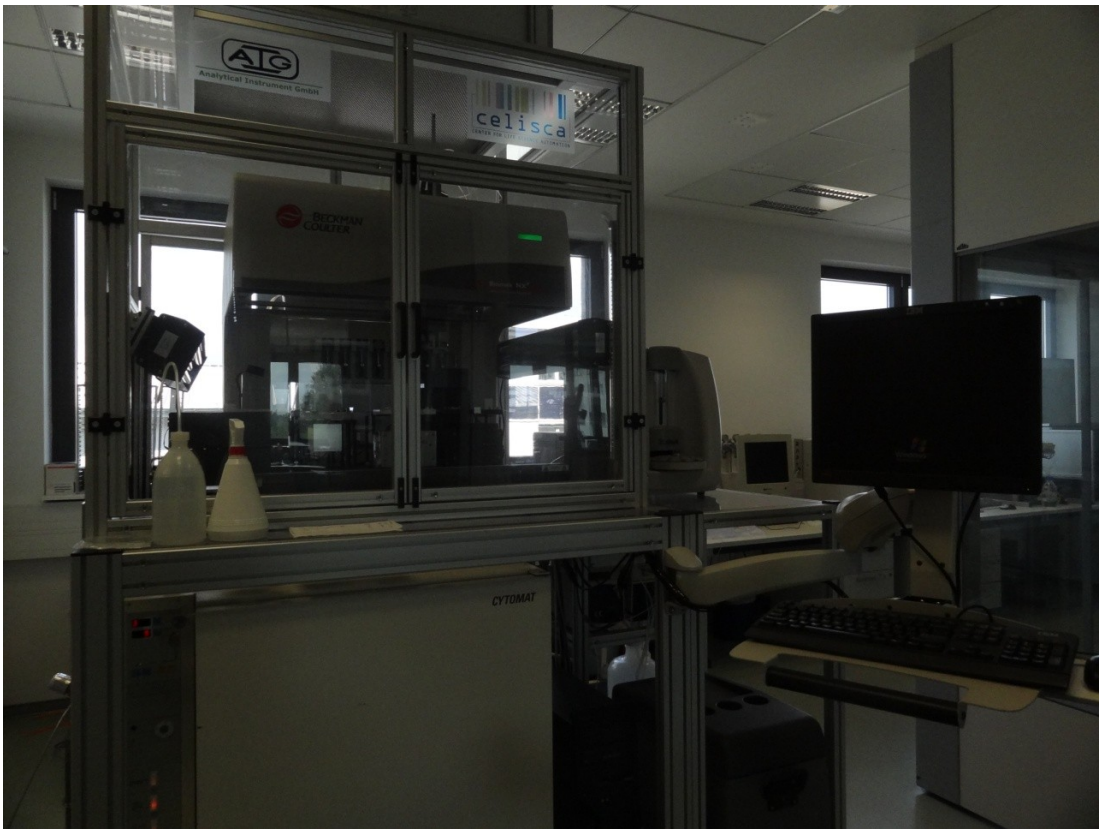


Figure 1.4: Cell Culture System at celisca

1.2 Literature Review

Virtualization for real-time experiment workflow of LSA refers to many fields, such as PCS, virtual simulation, data management, three-dimensional (3D) Computer-aided Design (CAD), four-dimensional (4D) simulation, etc. There are many relevant contributions from these fields be referenced and learnt in this dissertation.

1.2.1 Process Control System Workflow

Workflow is a depiction of a sequence of operations, which serve as a virtual representation of actual work [18]–[19]. In industry field and many automation laboratories, workflow is usually managed and defined by some workflow management system (WMS) or PCS within an organization to produce a final outcome [20]–[33]. The workflow data designed by the management or control system supplies a chance for researchers to do some virtual simulation.

Process Control System for Motoman System at celisca is SAMI Workstation EX Software (SAMI EX), which is utilized for developing scheduling, monitoring and running assays streamline operations on Beckman Coulter integrated systems [34].

SAMI EX software works with Biomek software and SILAS Integration system together for the whole workflow design and controlled operation [35]–[36].

SILAS Integration system works with SAMI EX as a communication router, which routes commands to and from the integrated devices. It allows SAMI EX to provide consistent dialog screen for device control, while invisibly translating commands and data between SAMI and the individual device's software [37]. In the SILAS system, instruments communicate with a module, which has an ActiveX software component (SILAS OCX) to connect itself with other modules. SILAS OCX control provides the writer of a new module with the means to communicate with other SILAS modules. It has built-in functions to create, place, send, request and save messages for later use. In addition, it includes an event that is triggered when a previously requested message arrives At that time, the programmer can get information out of the message using methods of SIALS OCX [38].

Therefore, it is feasible to register a module for a user to get the messages data from PCS via SILAS OCX. The function supplies the virtualization resource for the workflow virtualization of LSA experiments. That makes the virtualization possible to be realized.

1.2.2 Workflow Virtualization

At present, with WMS and PCS development, workflow virtualization becomes more and more popular in application fields [39]–[47], especially in business, medical and construction. It supplies virtualization effect for complex workflows. Its function of virtual reality saves testing and experiment cost for every application field, and brings much convenience for users to estimate work plans and demonstrate work ideals vividly. It is a flexible way to simulate workflow virtually in research or business occasions.

In business field, virtualization is widely used as a tool for business processes analysis and operational decision making [37]–[41], [44]–[48]. In the reference [49], M. Kovács and L. Gönczy presented a framework for the virtualization and formal analysis of workflow models, which are transformed into dataflow network models by BPEL language, and verified by SPIN model checker. A. Rozinat, etc., developed a simulation system for operational decision support by combining the workflow management system YAWL and the process mining framework ProM. [51]. In the IEEE conference COMPSAC'09, D. Eichhorn described his 3D simulation research

which added a third dimension into the graphical representation of process and data objects. The author referred future research on 3D virtualization and animation of other process objects (e.g. process metrics such as time, cost, etc.) [52].

In medical field, medical virtualization is a new method to facilitate skill training and assessment [53]–[60]. G. Bruinsma et al. proposed and demonstrate a method for simulating disasters for work and protocol optimization in disasters response (TAID), based on the multi-agent modeling and simulation language BRAHMS [61]. In the reference [62], the authors present a virtual imaging platform to facilitates the sharing of object models and medical image simulators, and provides access to distributed computing and storage resources. SA Schendel applied image fusion technology to increase further increased the importance and accuracy of virtual treatment planning [63]. A. W. Kushniruk et al. presented computer-based simulations that attempt to model human behavior [64], and simulations that are developed to test specific system components through health care information systems [65]. In clinic training programs, virtual reality (VR) tools become important in radiotherapy training for enabling students to simulate clinical situations without interfering with the clinical workflow, and without the risk of making errors. Immersive tools like a 3D linear accelerator and 3D display of dose distributions have been integrated into training, together with IT-labs with clinical software [66].

In construction filed, the technology of workflow virtualization is applied widely and developed in the leading. The researches on construction virtualization more relate to Computer Aided Design (CAD) and Computer Aided Engineering (CAE), which are developed vigorously. In the reference [67], by integrating lean principles and computer virtualization techniques, X. Mao and X. Zhang developed a construction process reengineering framework and methodologies, which classifies activities in the construction workflow to make it more effective in modeling workflow and virtualization. K. W. Chau presented a prototype four-dimensional site management model (4DSMM), which applied AutoDesk AutoCAD and ObjectARX development platform to simulate the construction process based on the scheduling data [68]. R. J. Scherer et al. designed a distributed multi-model-based Management Information System for virtualization and decision-making on construction projects based on ontology framework and Building Information Modeling (BIM) technologies [69]. M. Kugler and V. Franz developed a simulation system for the preparation work in building construction by Visual Basic (VB). The system provides a simulation editor which is integrated into a CAD system, and applies SQL database to manage the data

of process model [70]. In the publication [71], the researches mentioned and compared two approaches of virtualization techniques for construction field: one is rooted in scheduling, and involves linking activity-based construction schedules and 3D CAD of facilities to describe discretely evolving construction product virtualizations; the other one is rooted in discrete-event simulation, which concerns the virtualization of not only construction products, but also operations and processes in building courses.

1.2.3 4D Virtualization

With the development of the emerging technology Four-dimensional (4D), it is widely applied in virtual reality for several fields. 4D technology is a new virtualization method, which attaches time information to the traditional static 3D model, thus allowing planners to view workflow in a 4D environment [72]–[73].

In the researches of K. W. Chau et al. [68], [74]–[77], 4D technology is applied and extended into areas of resource management and site space utilization, in addition to planning of building construction solely; the papers delineate the development and implementation of a prototype 4D site management model (4DSMM) in a construction project. The prototype links a three-dimensional model and a construction schedule to furnish virtualization of the state of a site at any user-specified date. The system development referred Visual C++ and AutoCAD ObjectARX for programming, AutoCAD for 3D modeling, and the construction software GrandSoft CAD, as well as MS Project for construction workflow designing.

W. P. Segars et al. applied 4D technology for the development and improvement of some medical devices [85]. The excellent research achievement of the author is a 4D extended cardiac-torso (XCAT) phantom developed for multimodality imaging, which is a whole-body computer model of the human anatomy and physiology based on NURBS surfaces. In the phantom development process, x-ray projections of the 4D NCAT phantom were simulated using a cone-beam geometry and a standard x-ray energy spectrum [86].

C. Kim et al. used a 4D graphic simulation approach for analysis and modeling in a case study of cable-stayed bridge construction [87]. In the study, 4D CAD models were developed at levels as activity, discrete operation, and continuous operation. In J. Zhang's research [88], after comparing four virtual construction approaches, the author attempted to develop a 4D Virtual Construction and Dynamic Management

System, which integrates 4D technology, BIM and virtual construction technology to simulate and manage construction process dynamically.

In the reference [89], L. S. Kang et al. presented an information management methodology – a 4D simulation system, which uses Work Breakdown Structure (WBS) as an information center. In this research, the author supplies the same WBS codes for both scheduling and drawing information. The WBS codes are used as a library file, which is called and extracted throughout the whole process of the virtual simulation.

1.2.4 Real-time Virtualization

Real-time virtualization is related to the timeliness in the virtualization and application. It asks for not only high-performance computers, but also high flexibility and accuracy for virtualization tools, as well as the steady of the interface between reality and virtualization tool.

In the reference [90], a hybrid flow-battery super capacitor energy storage system (ESS), is studied by real-time hardware-in-the-loop (HIL) virtualization for being coupled in a wind turbine generator to smooth wind power. The prototype controller is embedded in one real-time simulator, while the rest of the system is implemented in another independent simulator.

K. Manoj et al. developed a distributed architecture for off-road vehicle dynamic models, 3D graphics virtualization and multi-rate model simulation to simulate various system dynamics with different integration time steps. The real-time simulation architecture includes three components: dynamic model simulator, virtual reality simulator for 3D graphics, and an interface to the controller and input hardware devices. Among the three components, the first one was developed by Matlab Simulink and SimMechanics for simulating dynamic models; the second one was realized by technologies of VR Juggler, OpensceneGraph and Extensible Markup Language (XML) data file; and the third one was developed by Visual C++, and applied Transmission Control Protocol / Internet Protocol (TCP/IP) socket technology for the communication between the first and the second components [91].

1.2.5 Discussions

Based on the literatures reviews, virtualization refers to technologies about database, CAD, programming, etc. LSA workstations at celisca have their separate PCSs. PCS works to control the realistic workstation, and it has data interfaces which could be

called by other platforms. PCS also supplies the data sources for virtualization. All the conditions show that it is feasible to simulate the experiment workflow of LSA by 4D virtualization. Through attaching time factors to 3D models and forming a 4D virtualization environment, the virtualization on real-time laboratory workflow of LSA has been realized in the dissertation.

1.3 Virtualization Strategy

1.3.1 Virtualization Ideas

"4D virtualization" is one of the computer-based process simulations. In the dissertation, "4D virtualization" is different from 4D (which also called "spacetime": space + time) in modern physics. It is defined according to Fig.1.5. Besides of the time factor added to 3D models, the "4D virtualization" includes 3D dynamic trajectories for all related components in every statement, and the links among statements to make the simulation coherent. The 4D virtualization system demonstrates the processes synchronously with the workflow data supplied. That's why we call it "on-line process". Due to the real-time data is generated with the realistic workflow running. However, virtualization is based on the workflow data. There is always time delayed for virtualization respect to the realistic workflow running. So we called the virtualization for real-time workflow data and realistic workflow as online virtualization.

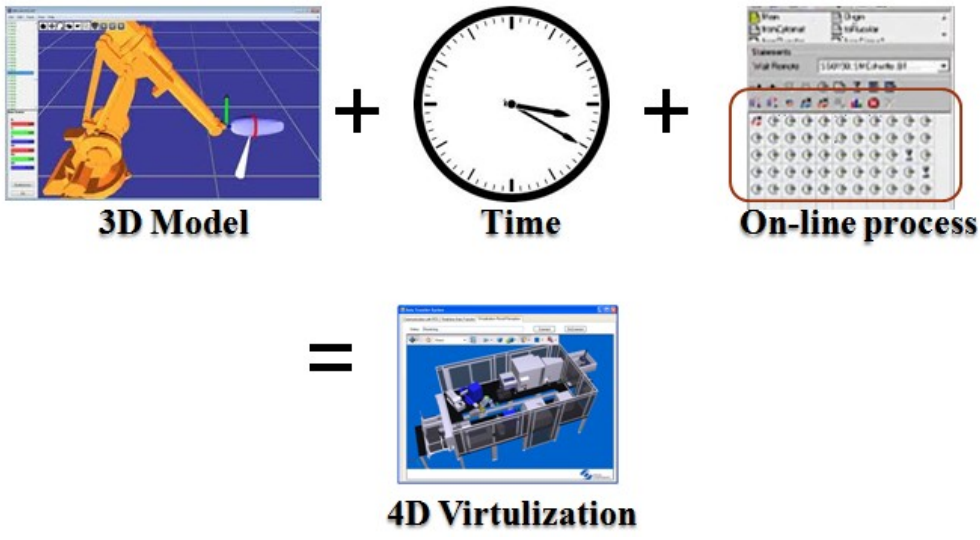


Figure 1.5: Definition of "4D Virtualization"

There are many ways to get workflow data from PCS, which designs and schedules experiment then realizes it in physical workstations [92]. The dissertation should choose suitable and strong 3D software to simulate the scheduling workflow in real time and high quality graphic display. Between PCS and 3D software, there should be an interface working as an intermediary to get data from PCS, assign the real-time data to 3D software and drive it to realize the synchronous 4D virtualization. In addition, the interface should have the function to drive the PCS to control physical workstations working based on the virtualization results. In summary, this interface is critical to integrate PCS, 3D simulation software and physical workstations as a whole Virtualization System (VS) for LSA laboratories at celisca. In this whole system, we defined the intermediary interface as Control System.

1.3.2 System Modules

Due to VS for LSA referring to various technology fields as PCS, workflow, database, CAD, dynamic simulation and programming technologies, the system could be mainly divided to four work modules: PCS, Control System (CS), Virtualization Module (VM) and Realistic Workstation (RW). The relationships among the modules are presented in Fig. 1.6. As Fig. 1.6 shows, PCS generates the scheduling data (real-time data in the figure) for CS. Based on the scheduling data, CS drives the VM to simulate the whole experiment process at once. Then CS receives the virtualization result from VM, and sends it back to PCS. Finally the virtualization process will be fully done with PCS driving RW to work.

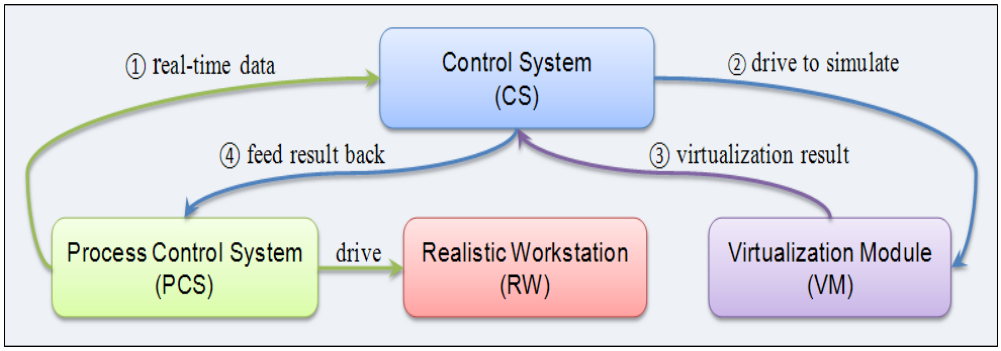


Figure 1.6: Relationships among the integrated system modules of the VS

As shown in Table 1.1, the dissertation applied some technologies and software to realize corresponding module functions of VS. Every module has their separated functions, which realize a part of the VS, and supply feasibility for other modules to realize the integration of them. Among the four modules, CS is mostly critical part, which connect other modules to an integration system and drives not only VM working but also the whole system running. Between PCS and CS, there is a data-transfer interface needed for their communication. In the dissertation, we call it as Data Transfer System (DTS). DTS calls SILAS OCX of PCS to get the workflow data, and applies TCP/IP Socket technology to realize communication with CS.

Table 1.1 Functions and applied technologies of the VS modules

Module	Functions	Applied technologies or tools
PCS	➢ Schedule methods/experiments; ➢ Generate scheduling workflow data; ➢ Supply OCX for communication; ➢ Drive RW to run as the scheduling methods.	SAMI EX integration system, Biomek software, SILAS OCX
CS	➢ Connect with PCS for data sending; ➢ Get workflow data from PCS; ➢ Process and manage data: extract, classify, save, .etc; ➢ Assign data to models in 3D simulation software; ➢ Control 3D software to realize online virtualization; ➢ Extract virtualization results from VM; ➢ Send virtualization results back to PCS.	Visual C#, TCP/IP Socket, database, COM API
VM	➢ Add models with behaviors and parameters; ➢ Simulate components movements synchronously based on the workflow data; ➢ Integrate all movements to an animation as virtualization result; ➢ Form fluent, nice graphics animation;	COM API, Python API, Visual C#, Python programming language, 3DCreate
RW	➢ Execute assay works for life science automatically based on the scheduling workflow.	Biomek integration technology, Automation and Control technology

1.3.3 Execution Methods

As Fig. 1.7 shows, to execute the virtualization idea of VS for the Biomek workstations at celisca, some interfaces are needed to connect the modules of the system: (1) SILAS between RW and PCS; (2) TCP/IP Socket between PCS and CS; (3) COM API and Python API between CS and VM. For SILAS, many devices modules in its system would be called for driving RW via PCS; for TCP/IP Socket, there are a Server and a Client for the data communication; for COM API and Python API, they are called to connect CS and VM for the driving and being driven in virtualization. To process and manage huge data, database is needed for PCS and CS. The database work could be assigned to the CS.

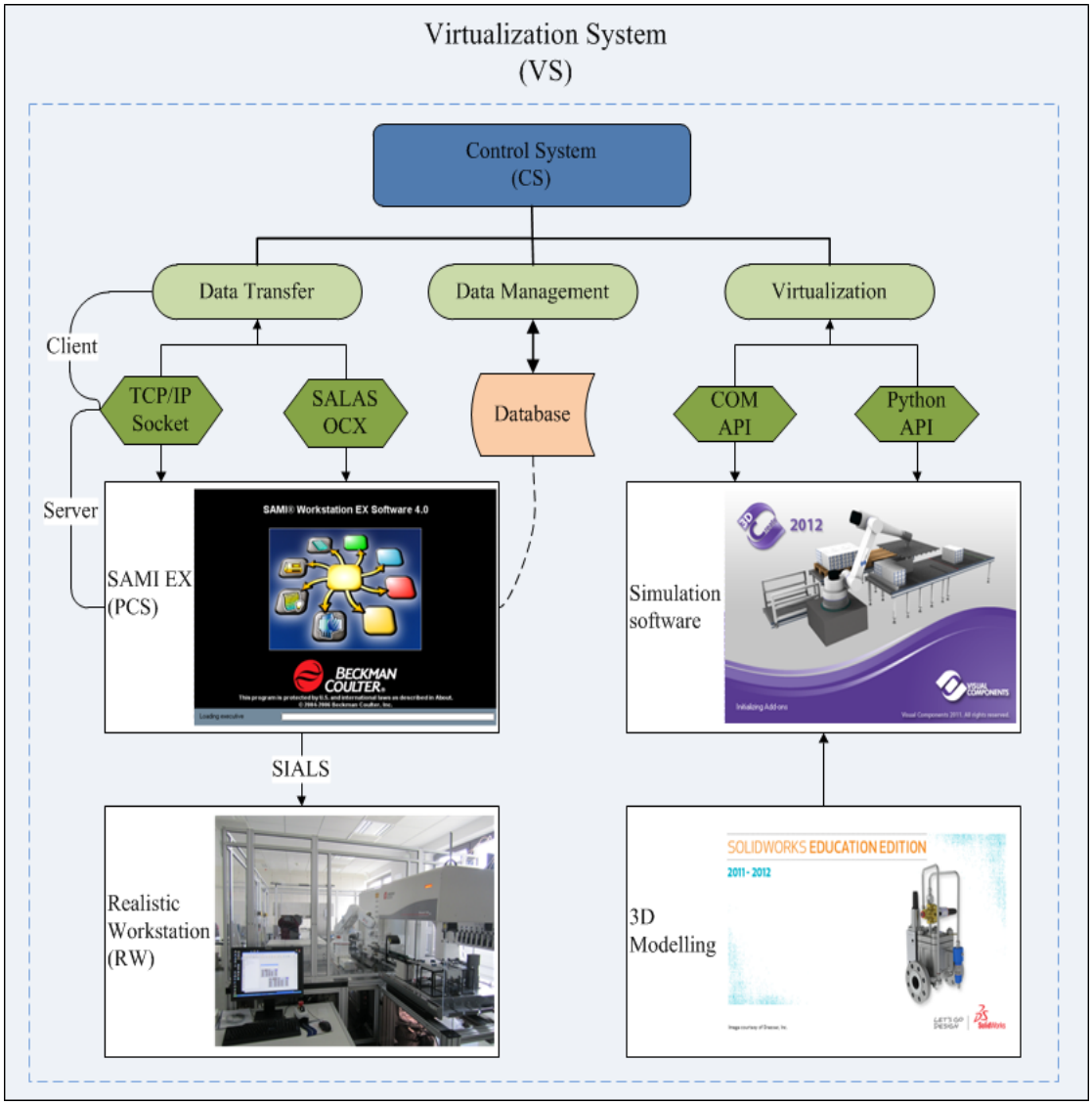


Figure 1.7: Working framework of the VS

For the critical module – the Control System, its functions could be assigned to three parts: data transfer, data management, and virtualization control. Anyone of the three parts is indispensable. Data transfer works for calling to get experiment scheduling data from PCS. In this part, the works for receiving data are finished by SALAS OCX, and the works for transferring data are done by TCP/IP socket. For the TCP/IP socket system, the interface in the PCS side works as a server, which is always waiting for others' calling, and prepares to send out the data; the interface at CS side works as a client, which calls for communication with PCS for receiving data when PCS is running. Data Management is in charge of saving and extracting important information from the scheduling data, which is send from PCS to CS. Due to huge amounts of experimental data especially for some complex experiments, here we use a database to save and manage the scheduling data. Either data transfer or data management is to make preparations for the virtualization. After CS received and extracted the experiment scheduling data, it will control and drive the simulation software to simulate the experiment process at once based on the data.

In the dissertation, the VS is supposed to be used for longer time periods. It should be upgraded flexibly. Compared to other developing languages, Visual C# has all advantages of them. What's more, Visual C# has an integrated development environment. It provides full COM / Platform support for integrated existing code. It also has easy and fast developing abilities, and the characteristics of upgrading packages. Thus C# has been used for system development in the dissertation.

The simulation software in the system is 3DCreate [93], [94], which has strong component object model (COM) application programming interface (API) and Python API for CS to call, and has strong 3D simulation functions as well as 3D rendering effects. When the controlled simulation is finished, the CS gets the virtualization result from 3DCreate and sends it to PCS. These processes are respectively finished by calling the COM API of 3DCreate and the TCP/IP socket.

The following chapters will analyze and expound the realization processes of every module and the integrated VS in detailed.

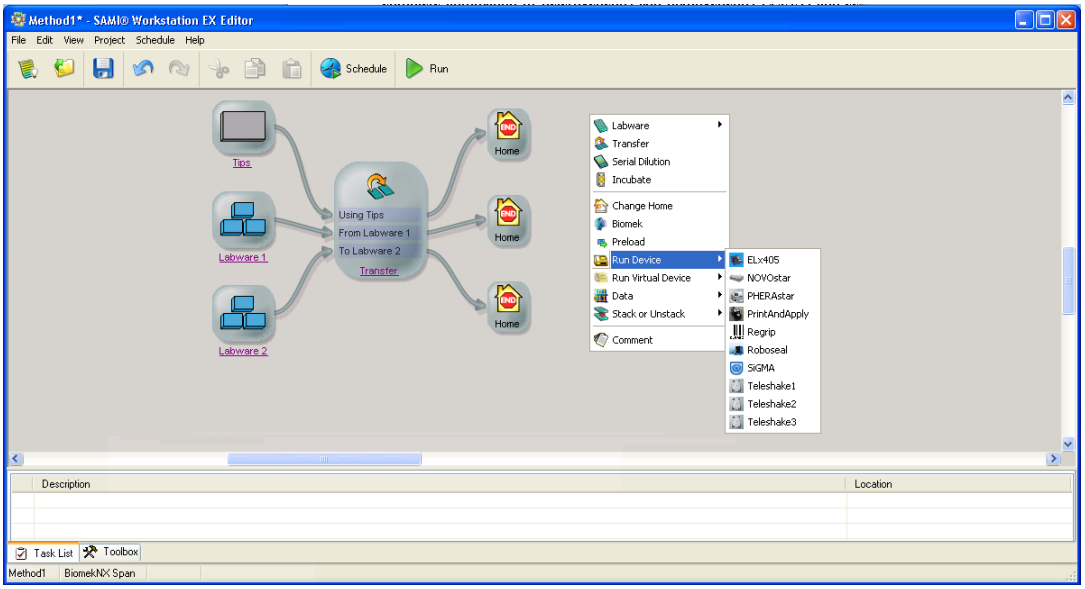
Chapter 2 Process Control System

2.1 Introduction

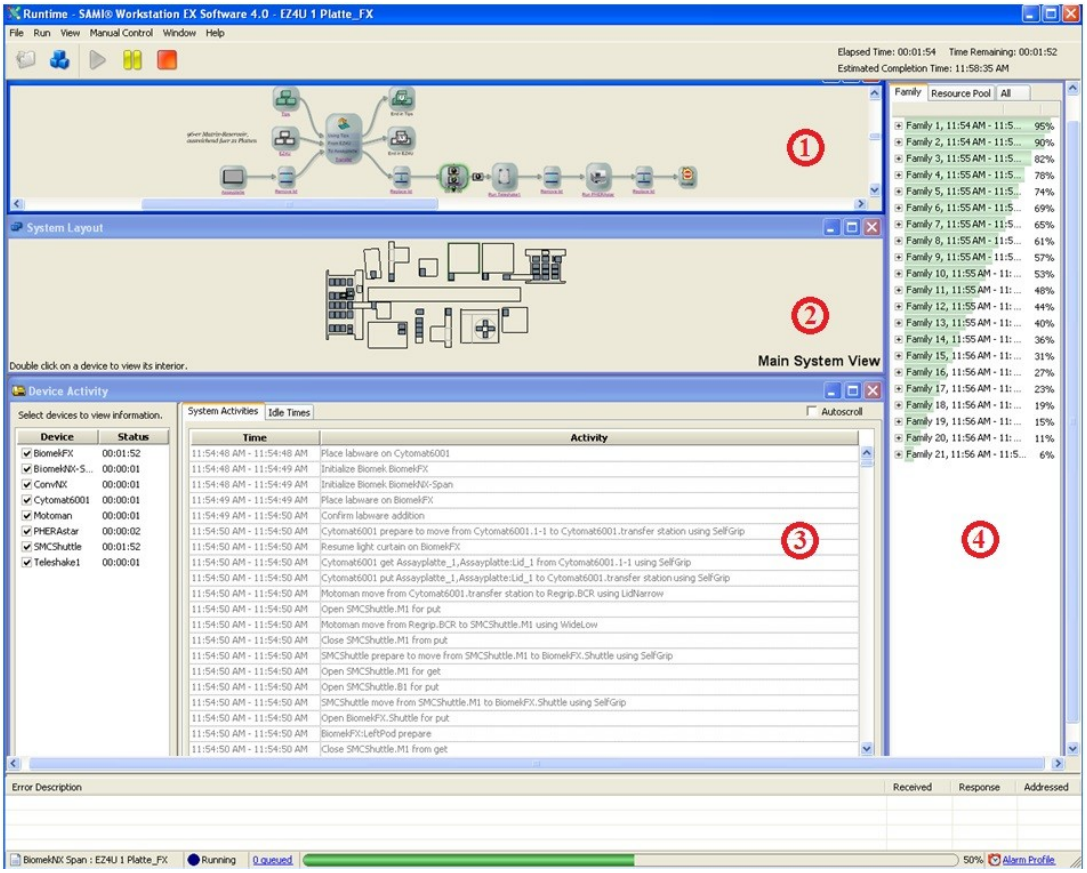
Process Control System (PCS) is very popular and important today in the Life Science Automation (LSA) field. It does life science assays accurately and effectively, which saves much time for scientists and laboratory assistants [95]–[99]. It frees scientists from the heavy assays, makes more time work on data and results analysis, rather than doing repetitive and dull experiments work [100]–[105].

At celisca, SAMI® Workstation EX (SAMI EX) software is an important PCS for its LSA workstations, which are called Biomek Assay Workstations. Both of the Biomek Assay Workstations and SAMI EX are developed and designed by Beckman Coulter, Inc. for LSA assay works. The Biomek Assay Workstation is a flexible assay platform designed to provide complete automation of heterogeneous and homogeneous enzyme-linked immunosorbent assays (ELISAs) and cell-based assays in a walk-away mode [106]–[112]. It is operated from the host computer using SAMI EX. SAMI EX provides a graphical interface to scientists and technicians for developing, scheduling, optimizing, running, and viewing automated assays on the Biomek Assay Workstation. It makes researchers describe complex assays flexibly in a straightforward and easy-to-understand way [107].

The SAMI EX software are shown as Fig. 2.1, which has two application interfaces for researchers: (a) Method Editor for creating methods and schedules – it's easy to develop an assay by creating many graphic nodes, behind which there are some dialogs for setting corresponding parameters; (b) RunTime for running schedules -- which has four parts for process information views: method view, device view, system view and labware view.



(a) Method Editor



(b) Runtime

Figure 2.1: SAMI EX software interfaces

2.2 Work Principle

To generate and receive workflow data from SAMI EX, it is necessary to know how SAMI EX works. As a PCS, SAMI EX does not work alone for driving automation workstation working. Behind its interface, there are Biomek software and SILAS which interact with SAMI EX to realize its controlling for the realistic workstation.

SAMI Workstation EX is tightly integrated with Biomek Software. The Software creates techniques, templates, labware and tip definitions in methods for SAMI EX. Particularly precise, complex, or linear methods that require actions be executed with rigid timing and using specific resources, such as deck positions, may be configured in Biomek Software and then run on SAMI EX.

SILAS is designed to simplify the integration of automated laboratory systems and to streamline the addition of new components. It works with SAMI EX as a communication router, routing commands to and from the integrated devices. It allows SAMI EX to provide consistent dialog screen for device control, while invisibly translating commands and data between SAMI and the individual device's software [112]–[114].

Figure 2.2 shows the working principle of SAMI EX, which also illustrates the relationship between SAMI EX and other modules. As Fig. 2.2 shows, SAMI EX acts on top of the Biomek software and uses it to drive some devices, such as the liquid handler. At the same time, SAMI EX uses SILAS for all of its IPC (inter process communication), which includes the Method Editor, Transportation, Executive / Scheduling, and all device modules.

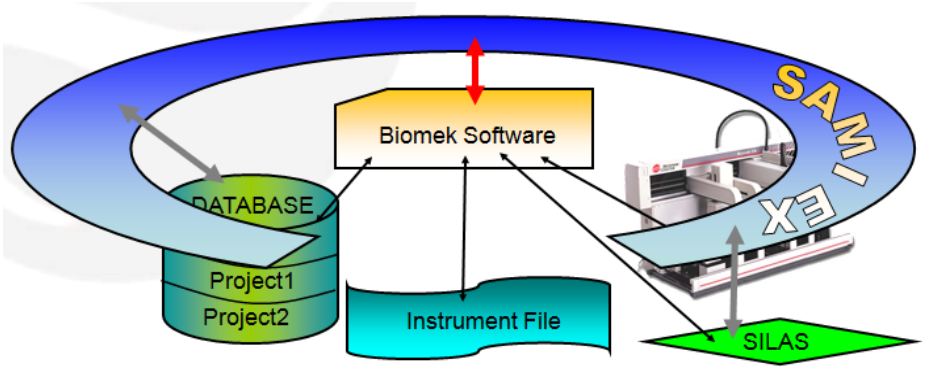


Figure 2.2: Working principle of SAMI EX [115]

In addition, SILAS steps operate devices on a Biomek deck during a method run by communicating between Biomek Software and the SILAS modules. In this module, SILAS stores all the information needed by an integrated system. The SILAS OCX provides the mechanism to attach modules to the rest of a SILAS system, and control resides in each module in the system that needs to communicate with other modules. So in the interface between PCS and Control System, the SILAS OCX is called to get workflow data from the PCS module.

2.3 Workflow Data

SAMI EX applies SILAS to generate and supply workflow data. SILAS provides a messaging protocol for laboratory integration based on ActiveX technology [38]. As Fig. 2.3 shows, there are three primary files with four items used in a basic SILAS system. The SILAS.exe file includes two core components of the SILAS system: Keeper and Router. The Router works to shuttle messages to the right places, and the Keeper tracks registration of modules in the system. The Keeper.rrg file holds the SILAS Keeper Registry, which stores all the information needed by an integrated system. In addition, message translations are also included in the Keeper Registry. For the MsgCtrl.ocx file, which works in the form of SILAS OCX, provides mechanism to attach modules to the rest of the SILAS system, and resides in every module for communication with each other in the system [116].

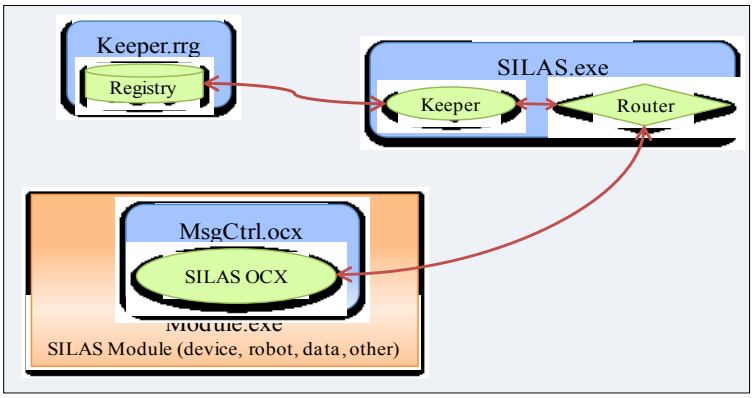


Figure 2.3: Communications in the SILAS system [116]

Figure 2.4 shows the SILAS working environment. From the Fig. 2.4, some hardware and robot connect with the SILAS system through their corresponding device modules in serial communication, other hardware is connected with SILAS via the third party controller software, such as Biomek software, and the database is connected to the Data Logger of SILAS via dynamic data exchange (DDE). Among the SILAS modules, the communication either between the Registry and the Keeper, or between the Router and the Keeper is internal. The SILAS modules, which are shown as mint and ellipse in the figure, are connected with the Router by SILAS messages, which are controlled by SILAS OCX.

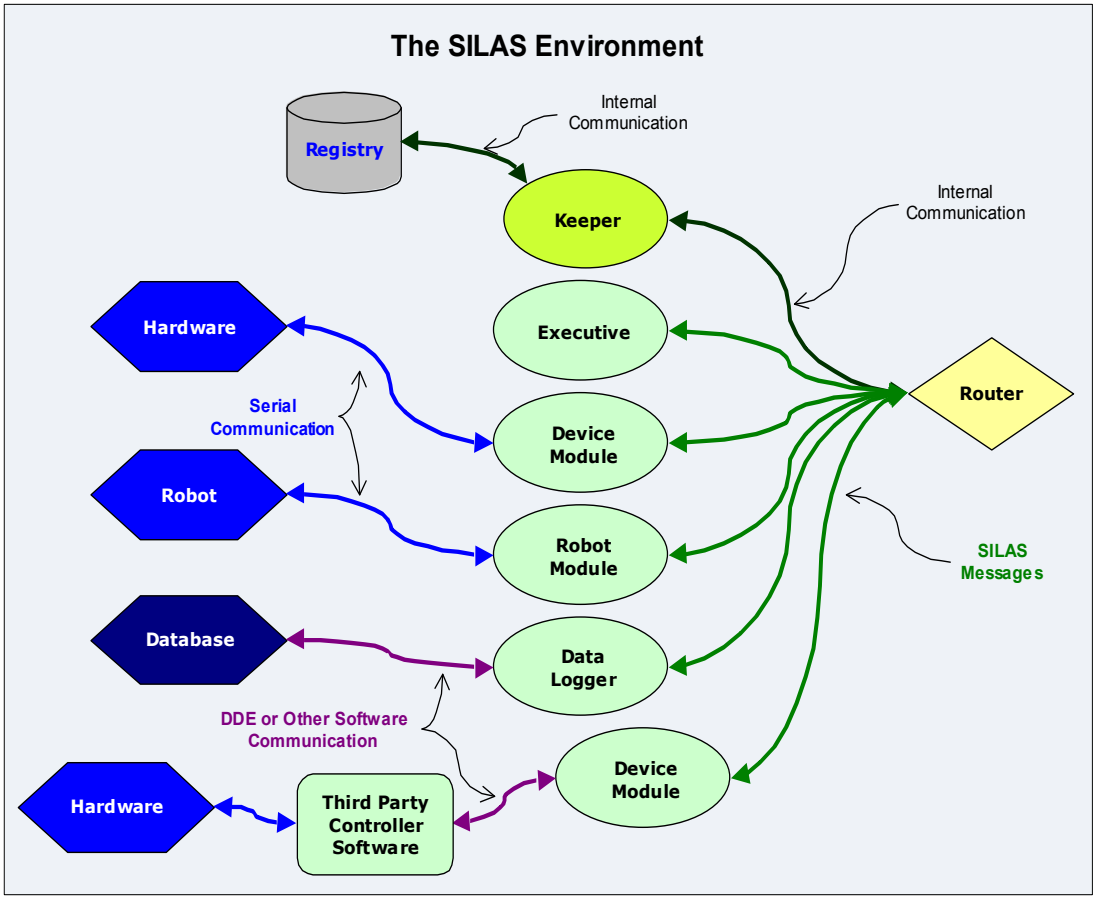


Figure 2.4: SILAS environment [116]

The SILAS modules, including device modules, robot modules, data logger, and SAMI executive, are registered in the Keeper Editor, which is shown in Fig. 2.5. For every registered module, there are many kinds of parameters and information inserted, which define messages the system should generate and transfer. Therefore, to get messages from the SAMI EX system, at first, there is needed to create and register a new module in the Keeper Editor, such as the module named “Workflow” in Fig. 2.5.

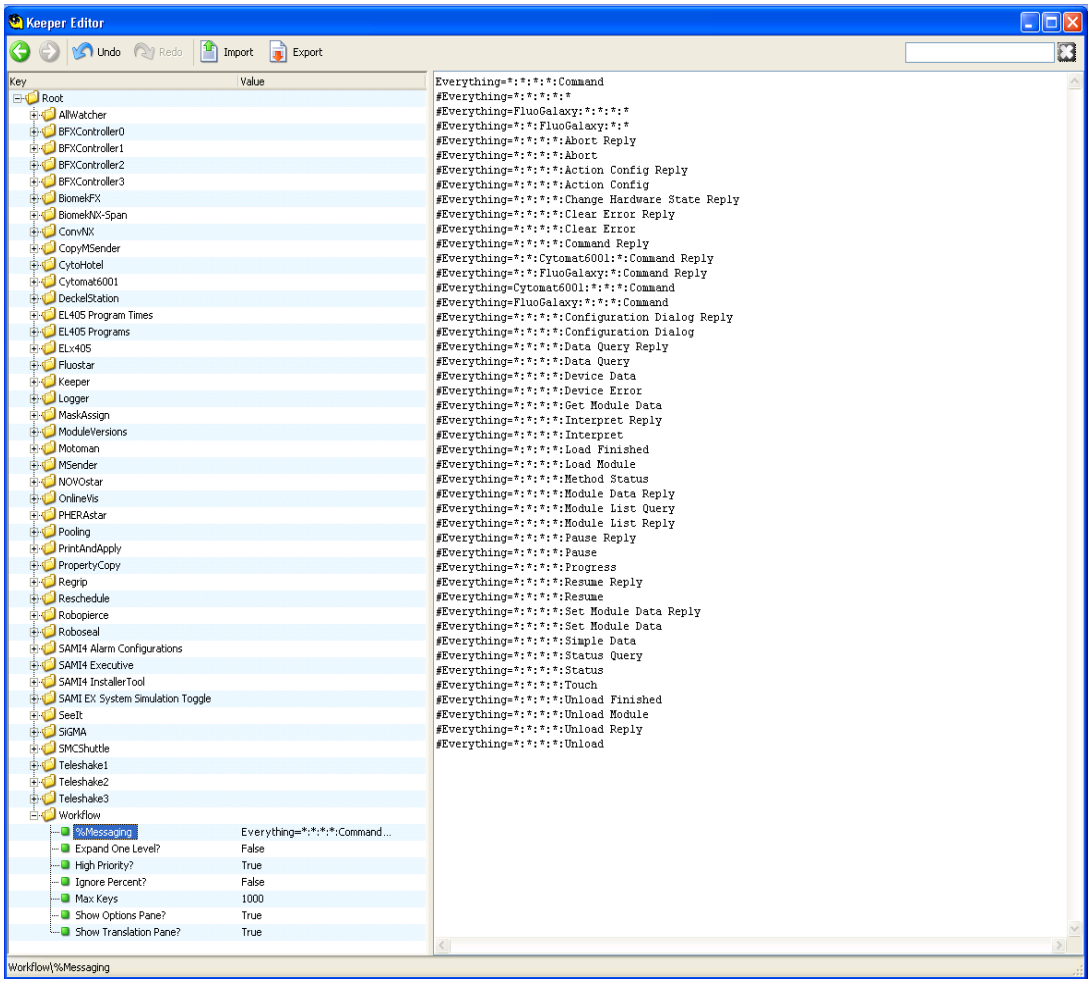


Figure 2.5: Keeper Editor

After scheduling a method in the Method Editor and running it in the Runtime of the SAMI EX software, there are a large number of assay messages generated.

SAMI EX system, The SILAS messages depict the scheduling workflow of LSA assays, which include all information the 4D virtualization needs. It could be shown as AllWatcher interface of SAMI EX system.

As Fig. 2.6 shows, the information in AllWatcher has characteristics as the followings:

(1) Tree-like structure;
(2) Messages can contain sub-messages and/or leaves;
(3) Sub-messages can contain sub-messages and/or leaves;
(4) Leaves contain strings of text or binary data.

From Fig. 2.6, it could be found that all parameters and actions of the workflow are included, such as time, source, destination, grip and action, .etc, which is critical for the virtualization. Every message could be accessed by calling SILAS OCX. In addition, the SILAS messages are also shown in the Device Activity in the Runtime interface of SAMI EX software, as Fig. 2.1 (b). In that part, the workflow data could be saved as .csv format.

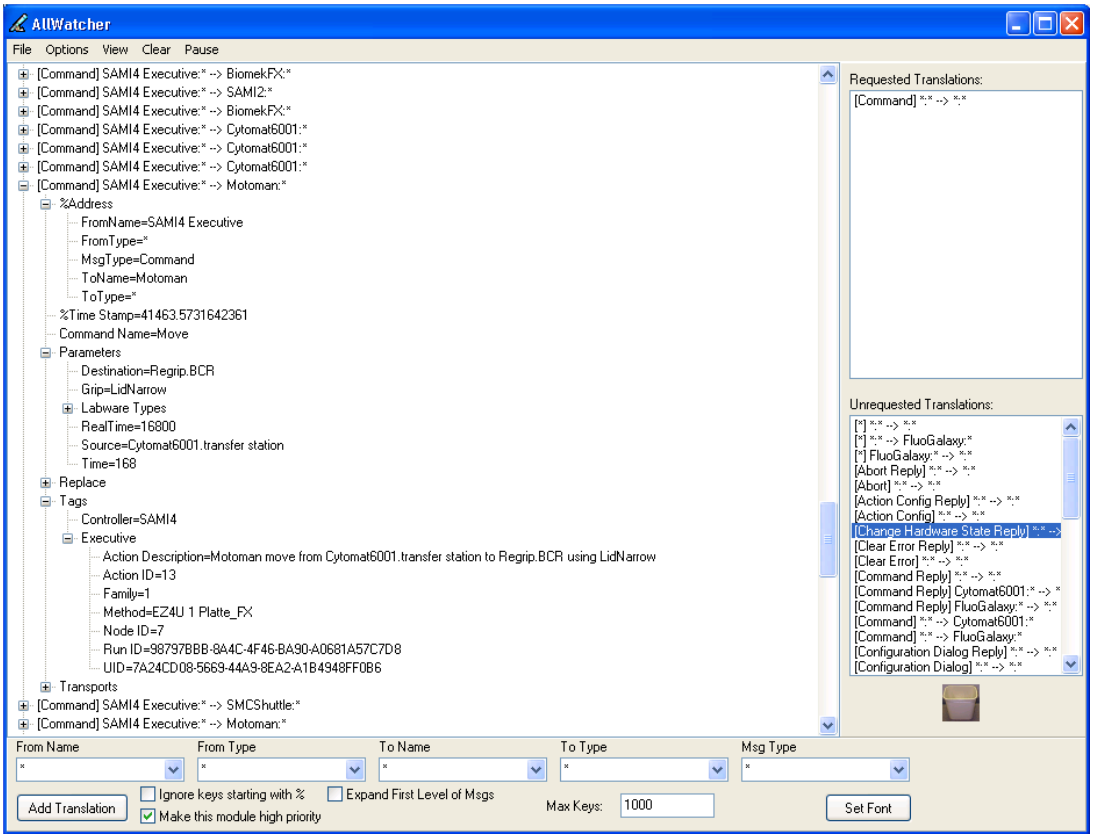


Figure 2.6: Meassages in AllWatcher

2.4 Discussions

To realize the 4D virtualization for a LSA workstation, the PCS should have a tool/system to generate and export process data designed. For the PCS SAMI EX, it is possible to realize SAMI EX workstation system virtualization, because of the

followings:

(1) It has strong process data system, and adequate OCX files for managing and supplying data. They supplies objects for 4D virtualization;
(2) It connects with the realistic workstation and has interface to control it working. It supplies functions for other system to call its OCX files. So SAMI EX software makes controlling the realistic workstation by other outer systems possible.

However, the SAMI EX system itself has no graphic interface to demonstrate virtualization results. Thus it is necessary to develop such an interface at the SAMI EX side for process designers to see the virtualization result directly. That is the Data Transfer System (DTS) detailedly depicted in the Chapter 3.

Chapter 3 Data Transfer System

To create communication between PCS and CS, there is an interface needed to link them. In this dissertation, as Fig. 3.1 shows, such an interface named Data Transfer System (DTS) is developed by Visual C#. The DTS works to get scheduling workflow data, send the data to CS and receive the virtualization result from CS. The system refers to technologies including SILAS OCX, Visual C# programming, and TCP/IP socket. Corresponding to its functions, the DTS is consisted of three modules: (1) Communication with PCS; (2) Data transfer for real-time workflow data; (3) Data reception for virtualization result.

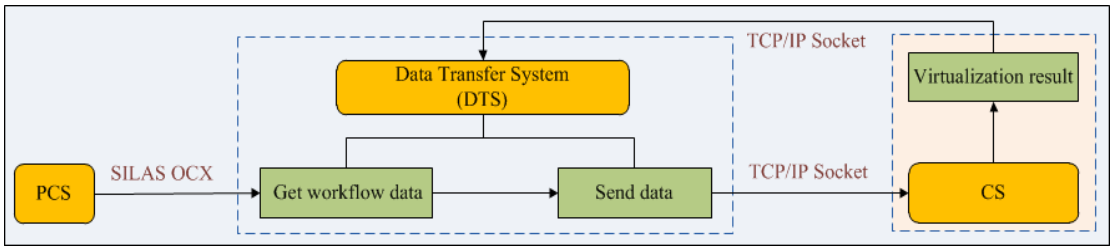


Figure 3.1: Workflow of the Data Transfer System

3.1 Communication with the Process Control System

When the scheduling method is run in SAMI EX, the module “Workflow” gets the workflow data one by one in the form of tree-structure as shown in the AllWatcher interface (Fig. 2.6). In the “Workflow” module, there are many messages in one assay project, and many sub-messages for every message. Take the message “[Command] SAMI4 Executive.*--> Motoman.*” in Fig. 2.6 for example, there are five sub-messages for this command, and under every sub-message, there are many sub-messages for themselves. Not all messages are useful for the 4D virtualization. The important ones are in types of “%Time Stamp”, “Time”, “Action Description”, and the “Status”, .etc. In them, the sub-message "%Time Stamp" means the start time of every process in the experiment workflow, the “Time” means the duration time for every process, and the “Status” is generated automatically while a process is finished. All the messages generated in SILAS system are in the form of String.

As Fig. 3.2 shows, after running the DTS, SILAS OCX is called to connect with the SILAS system of PCS. While the connection is successful, the module “Workflow” is registered in the Keeper Registry by calling related commands. Then DTS calls corresponding OCX commands to initialize the message controller, and request some message translations for receiving the corresponding messages the virtualization needs, such as the “Command” translation for workflow information. Once the scheduling method is running in SAMI EX, the “Workflow” module gets the SILAS messages. At the same time, the event called RcvMsg will occurs in DTS for extracting messages in the “Workflow”. In this event, DTS defines the types of messages and sub-messages’ which it would extract. To make sure there will be no data loss in the transferring even when the link is interrupted, after the data transmission is finished, the saving-message command is called in the RcvMsg event for backing up the received data. Finally, the message controller is cleared for depositing new experiment data.

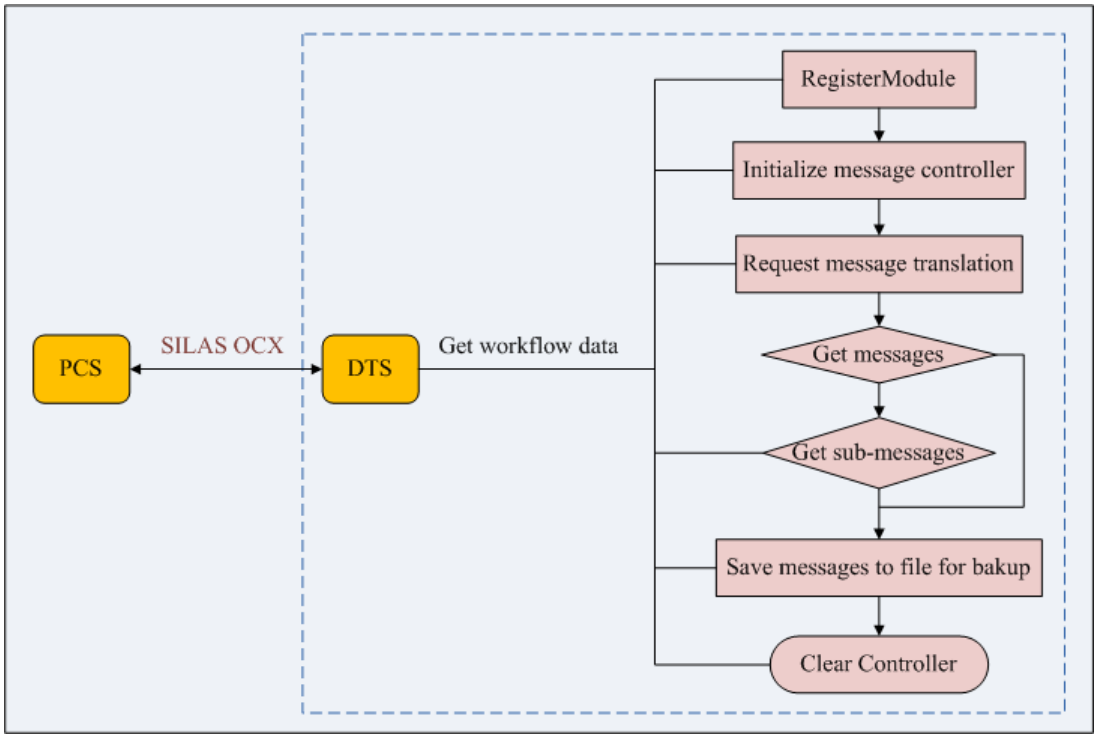


Figure 3.2: Workflow of the communication with PCS

In this dissertation, the messages with types of "%Time Stamp", "Time", "Action Description", and “Status” are enough to describe the experiment workflow. Therefore, DTS extracts these data from the SILAS messages.

Figure 3.3 shows the process of messages extraction. In the Fig. 3.3, the yellow modules are workflow data. When the "Workflow" module is registered, its "Command" translation is requested. Under the "Command" translation, there are many sub-messages and parameters. DTS calls corresponding commands of SILAS OCX to arrive at the target sub-messages and get the string of messages needed along the tree branches.

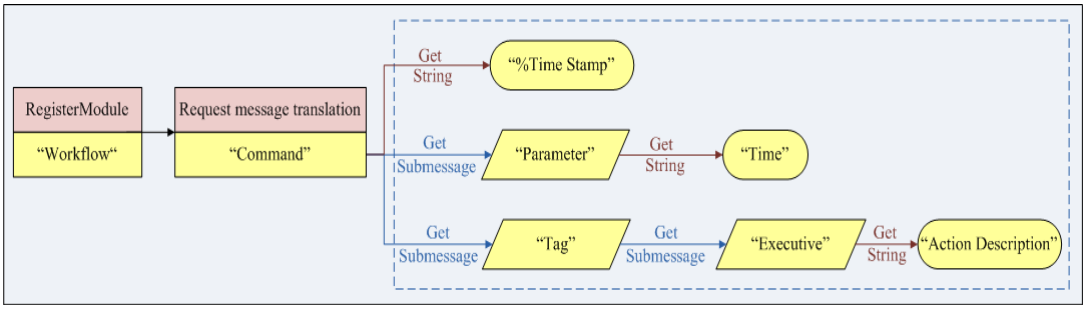


Figure 3.3: Message extraction in the DTS

The "Communication with PCS" module is developed as Fig. 3.4. Once SAMI EX Runtime is working and DTS is active, the textbox will show the received workflow data one by one. As Fig. 3.4 shows, every data has four parts of messages, which are all in string format.

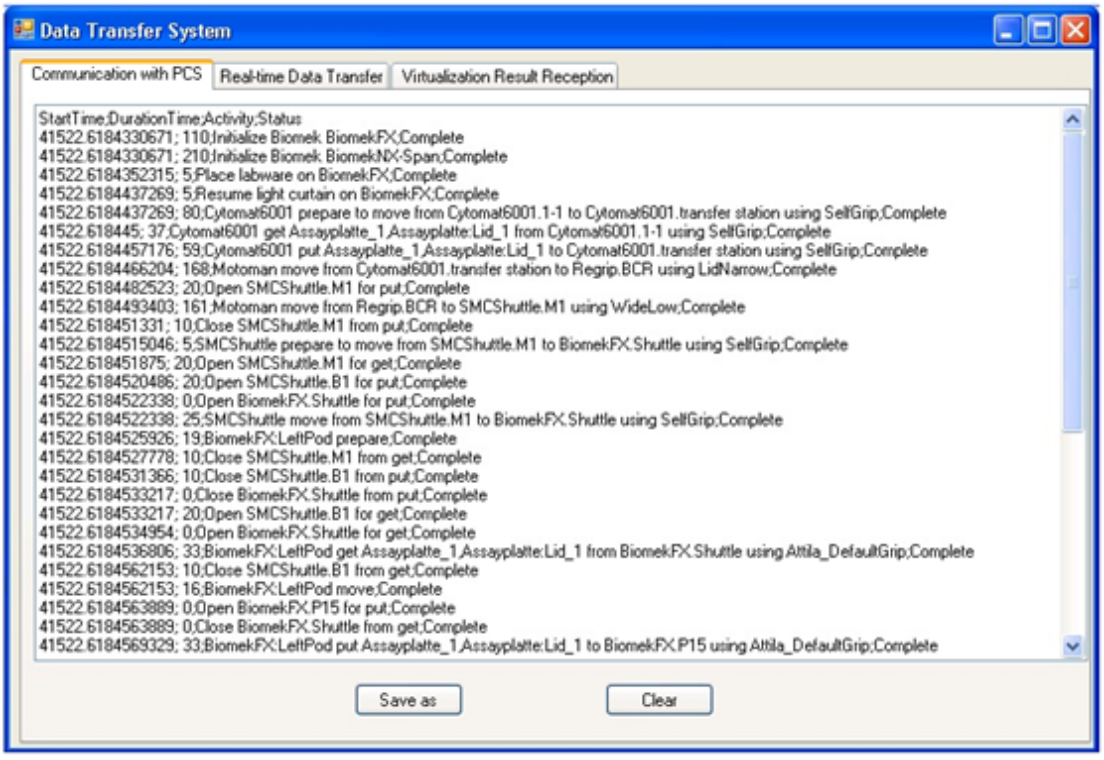


Figure 3.4: Communication with PCS

3.2 Real-time Data Transfer

The works of the real-time data transfer in DTS include two factors: one is for sending real-time workflow data to CS, the other one is for getting the data-received condition from CS. They are realized by TCP/IP socket technology. In the data communication with CS, the DTS socket works as a server, which always waits for being called in some specified net port as a watchdog.

For the workflow data communication, the server socket has two functions: data formats conversion and data transmission. On the one hand, since the data transmission works between two computers or two different ports in one computer, it is needed to convert data formats for users reading and computer recognizing. Due to the workflow data gotten from PCS is in string, and the data about received conditions from CS is in byte, there are corresponding encoder and decoder needed for data formats conversion. The encoder converts data from strings to bytes for computer understanding, and the decoder converts from bytes to strings for user reading. There are many trans-coding formats for data conversion, such as ASCII, Unicode, etc. To union the coding format in the whole data-transfer process, the dissertation applies the Unicode format. On the other hand, the server socket transfers the encoded workflow data and decoded data-received condition data with the client.

Figure 3.5 depicts the workflow of data transfer between DTS and CS. As Fig. 3.5 shows, in the function "SendOutData()", DTS creates an encoder to convert the strings data from PCS in the format Unicode. In this function, the encoder encodes the data in string to Binary bytes, which computer CPUs could recognize. Then the DTS socket applies TCP/IP technology to send the bytes data in real time to the client which requests communication through its port. Similarly, DTS applies TCP/IP technology to get the data about received conditions from CS through its port. These data are in byte format sent from the CS computer. To make the data readable, the socket creates the "DataReceived()" function to convert it to strings. In this function, the socket creates a decoder for decoding these bytes data to strings. It does just the opposite function of the encoder in this dissertation.

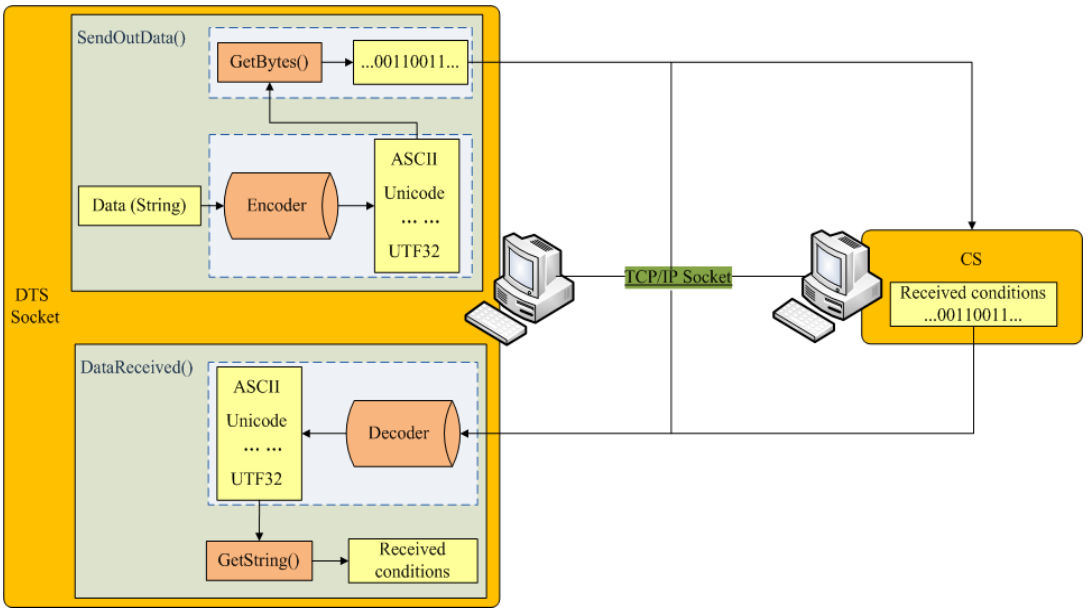


Figure 3.5: Workflow of data transfer between DTS and CS

Figure 3.6 shows the working interface of TCP/IP socket of DTS in two different statuses. As Fig. 3.6 shows, the DTS socket module has three functions: showing working status, communication test, and listing feedback information from CS. There are three statues for the system. When the DTS socket is successfully bind to a specified endpoint/port, the status is shown as “Waiting for a connection” at the port. Otherwise, the system throws the information as “Socket errors”. Once the socket is connected by a client successfully, the status box will show the connected information as well as the calling client IP address. There is one button for testing communication with client computer. This function aims at testing whether the data transfer process could be executed rightly. It is helpful to check the communication between DTS and CS before DTS gets workflow data from PCS. The textbox at the bottom of the interface lists the communication condition which is replied from CS. As long as the communication between DTS and CS is created, either for communication test or for data communication, the textbox always shows the receiving information from CS.

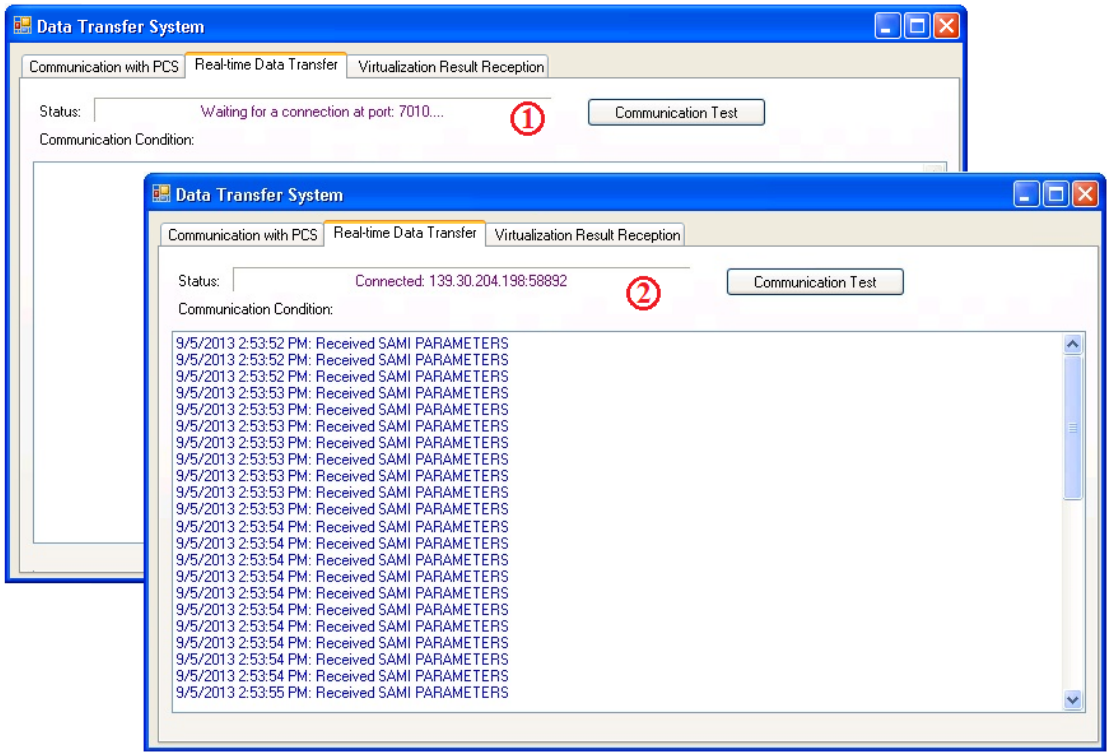


Figure 3.6: The TCP/IP socket interface of the DTS

Data transfer workflow in the DTS socket (server socket) could be presented as Fig. 3.7. At the beginning of this socket, the system binds the listening socket to a specified port as a watchdog, which always keeps at the port for any calling. When DTS is activated, the socket begins to wait at the port for some connection requests. An error warning is throw out if there is some problem to bind the socket to the port. But if the socket bind course is successful, once CS calls for connection with DTS, the socket system will start an asynchronous operation to accept the request, and send the encoded data to CS.

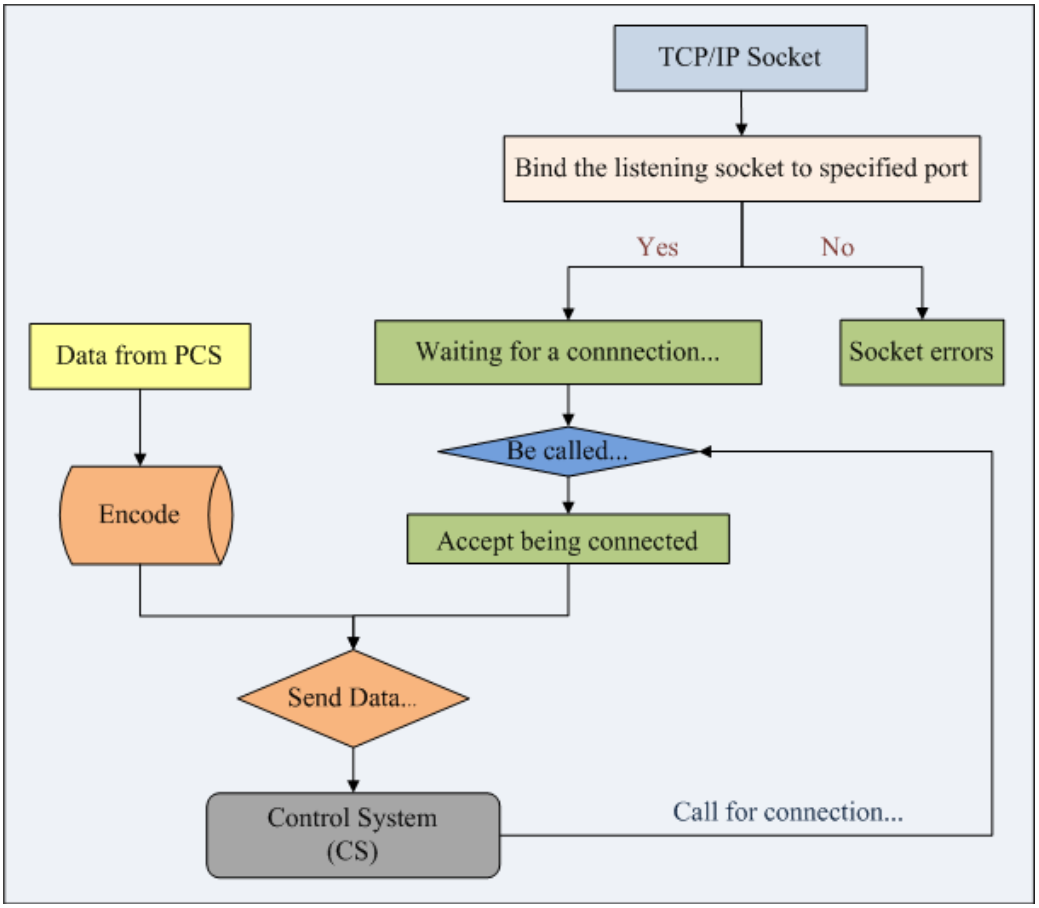


Figure 3.7: Workflow of the TCP/IP socket for Data Transfer

Algorithm 3.1 shows the running logic in sending data to CS. From Algorithm 3.1, the waiting work for communication begins with DTS being run. When the connection between DTS and CS is created, and the communication test is certified by the reply information from CS, it means data transmission channel is smooth. On this condition, DTS finishes its preparation for processing and sending workflow data to CS synchronized with receiving data from PCS. Once PCS runs scheduling method, DTS will receive the workflow data one by one in real time, and then encode and send the data in bytes to CS synchronously.

Algorithm 3.1 The Data transfer logic for TCP/IP socket of the DTS

Start: *Run DTS*
Step #1: *The DTS starts to listen.*
1: *Define IP address and server port.*
2: ***Try:*** *Bind the listening socket to the port.*
3: **if** *successful* **do**
4: *Listen…*
5: *Show socket status as "Waiting for a connection at port....".*
6: **else**
7: *Show socket status "Socket errors".*
Step #2: *Waiting for the request of connection.*
8: **if** connection is *successful* **do**
9: *Accept the request ;*
10: *Show socket status "Connected" and show the Client IP address & port.*
11: **else**
12: *Waiting…*
Step #3: *Wait for data from PCS*
13: **if** *data come in:*
14: *Encode data as bytes.*
15: *Send the encoded data to CS.*
16: **else**
17: *Waiting…*

3.3 Virtualization Result Reception

The Virtualization System (VS) realizes the virtualization for PCS. So the virtualization result should be shown at the PCS side. That is also done by TCP/IP socket technology. Due to the works of sending workflow data and receiving virtualization result independent, there should be another TCP/IP socket for the

virtualization result receiving. The socket is also embedded into DTS, and works as server in the data communication. As shown in Fig. 3.8, there is an Adobe Reader OCX embedded into the DTS interface, which could show the virtualization result directly for PCS workflow designers and visitors.

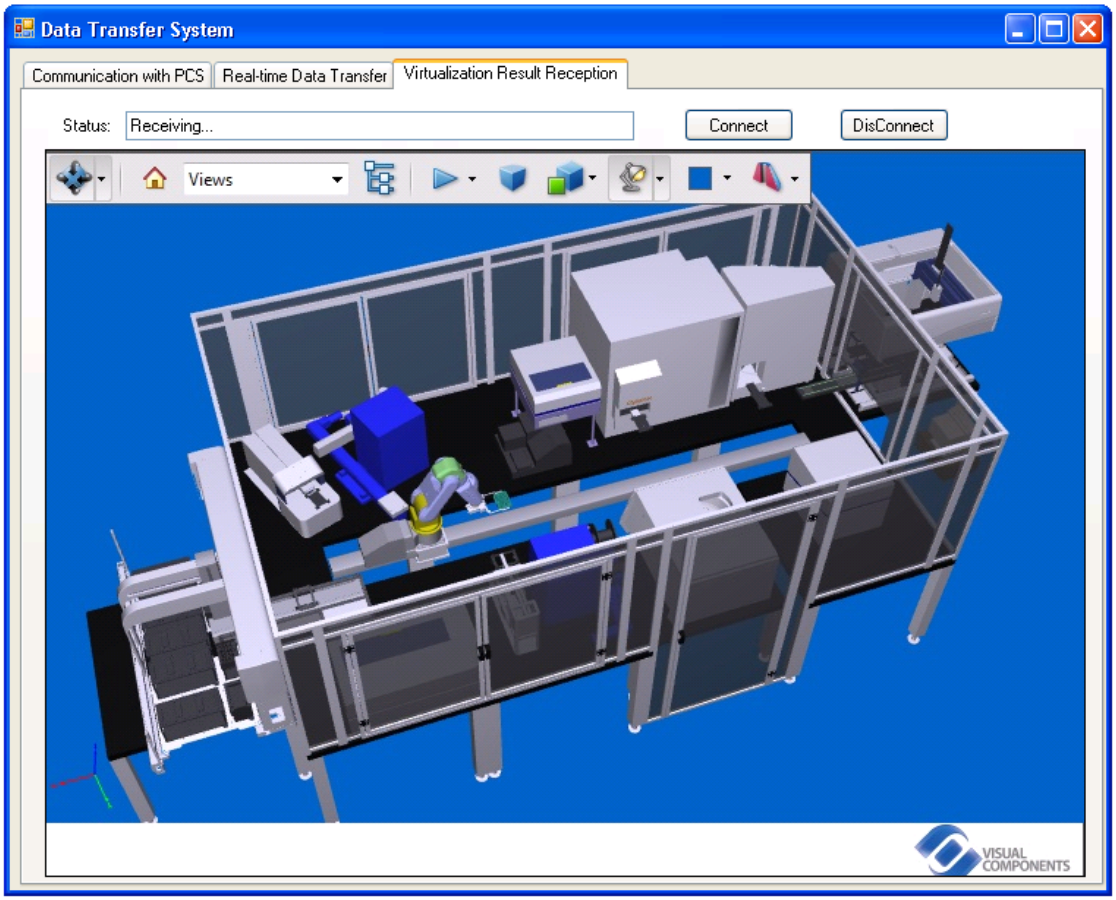


Figure 3.8: The Interface for virtualization result demonstration in DTS

In the online virtualization for real-time data, with the data being simulated one by one in CS, the virtualization result is saved as many animation parts in .pdf format. After the connection request from CS socket is accepted by the DTS socket, the communication for virtualization results between DTS and CS is started.

In the process of the virtualization result reception, the DTS socket receives the data stream of the virtualization results from CS socket one by one. The socket decodes the data stream back to .pdf file in Unicode format. Then the .pdf files are backed up in the DTS side and demonstrated synchronously in the DTS interface one by one. As for the historical data virtualization, since the virtualization result is also separated to many parts and sent to DTS in data stream, the DTS socket works as the same as the real-time data demonstration.

When there is some interruption for the virtualization result communication, the DTS socket begins to wait for being called again. However, the animation parts go on showing in the DTS interface until there is no one left. Once the communication is created once more, the DTS socket receives and saves the coming animation files from the last interrupt point. At the same time, the socket shows the new files in the GUI one by one.

As the TCP/IP socket for workflow data transmission, while the DTS socket receives one data stream of animation from CS, the socket feeds the reception information back to CS socket at once.

3.4 Discussions

The Data Transfer System works at the side of the PCS. It supports the PCS in realizing the data transmission with the CS. It also supplies an institute visualization result for process-designers to check his design, rather than to expect the designed process could run successfully in the workstation.

Because the performance and transmission forms of workflow data are generally different for different PCS. So within the DTS, the module for communication with PCS has a bit difference for different PCS. For a new PCS, based on its data format, the system developer needs to add a corresponding module into the DTS for data processing and transmission.

Chapter 4 Control System

In the virtualization process, the Control System (CS) is the most critical and important module, which links PCS, DTS and 3D simulation module as a whole, and realizes online virtualization for experiment workflows in LSA flexibly. The VS functions are mainly reflected to users by the CS interface.

The Control System has four modules to realize its control functions:

1. TCP/IP socket module for the communication with DTS;
2. Data processing;
3. Virtualization control;
4. Post process of the virtualization.

The relationships among these modules are shown as Fig. 4.1. The TCP&IP socket works for communication with DTS to get the real-time workflow data, and to feed back its data-received conditions to DTS. The socket also backs up the real-time data for historical data virtualization. In CS, there are two data processing modules for preparation of different type-data virtualizations. For either the real-time data or historical data, the CS module could realize the virtualization for the experiment workflow. That module links with the virtualization module (VM) by calling its COM API, and assign the data to related components in VM by calling both of COM API and Python API. CS drives the VM to create a series of movements as the data depicts, and makes VM to form the movements to an animation. At last, the CS calls COM API of VM to get the animation as the virtualization result, and sends the result to DTS via TCP/IP socket.

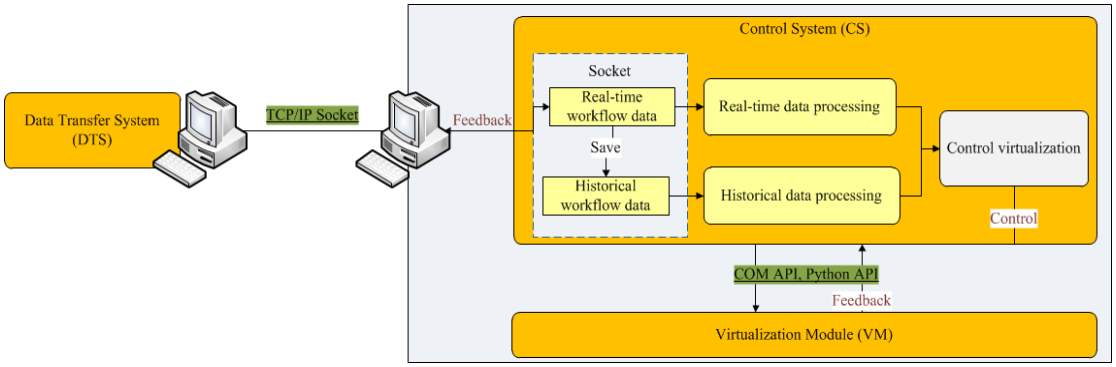


Figure 4.1: Workflow of the TCP/IP socket for Data Transfer

4.1 Communication with the Data Transfer System

The TCP/IP socket in the CS works as a client to request communication with DTS. As Fig. 4.2 shows, at first, the client CS calls the server -- DTS socket for connection (step ①). The calling has two results: accepted or failed. If it is accepted, the CS would wait for step ②; if it is failed or rejected, the CS socket will call repeatedly until it is accepted. As for the step ②, the DTS encodes the workflow data to bytes and sends them to CS via its TCP/IP socket, and correspondingly, the CS waits for and receives the data in byte via its own TCP/IP socket. Once the CS receives the data, its socket would reply the data-received conditions information back to the DTS (step ③).

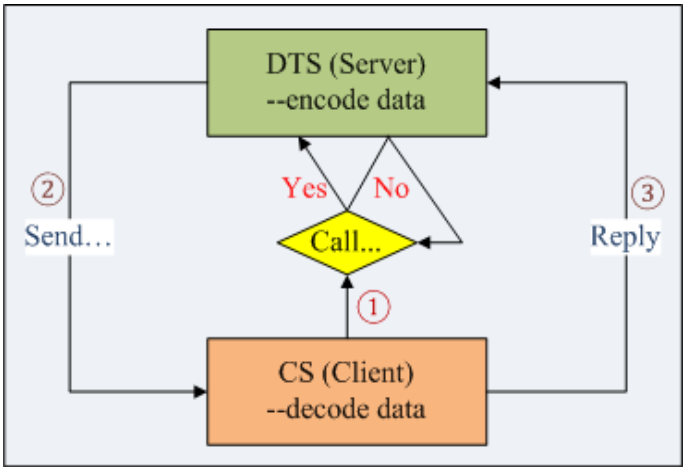


Figure 4.2: Data Transfer between the server and the client

Besides of the functions for connection and data transmission, the CS socket also does some parts of data processing works. That is for data format conversion. It is somehow similar but in opposite direction as the functions of the DTS socket. As described in the previous chapter, the DTS socket encodes the workflow data to bytes for computer to recognize. So when the computer with the Control System receives the bytes data, they should be decoded back to string format for users to read.

As showing in Fig. 4.3, the CS socket firstly creates a decoder, applies Unicode format to convert the bytes data to strings. Those works are realized in the function "ShowReceivedData()". That is the original works for real-time data virtualization. For the feedback step of CS socket, there is another function "SendOutData()", which is in charge of sending data-received conditions to the DTS computer via TCP/IP socket. The "SendOutData()" function works oppositely with the "ShowReceivedData()" function. It creates an encoder for converting the feedback data in string to bytes, which could be recognized and transferred between different computers.

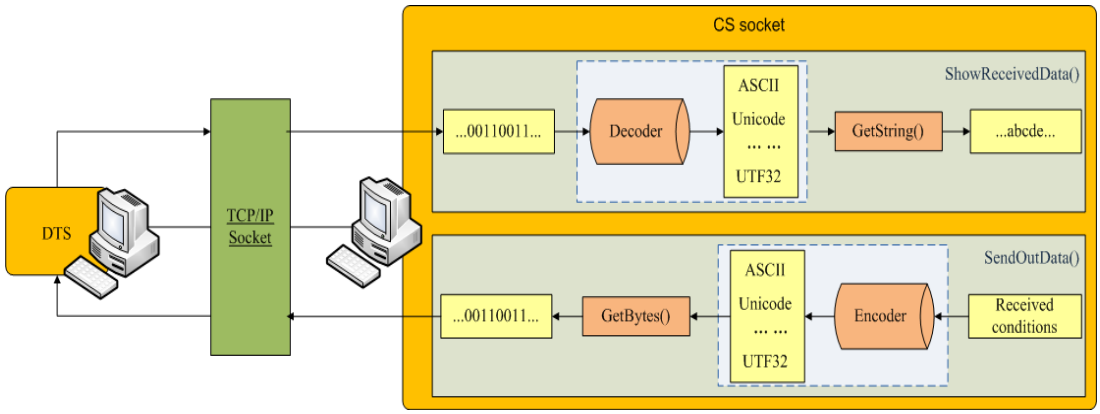


Figure 4.3: Data conversion in the CS socket

Figure 4.4 shows the internal functions structure of CS socket. In the CS module, there is a sub-class "ParameterSocket" created to realize the communication functions of TCP&IP socket. For functions in the main class "ControlSimSystem" and the sub-class, they call related functions from each other. The left box lists the functions of TCP/IP socket in the main class named "ControlSimSystem", and the right-bottom one contains the functions of the class "ParameterSocket", which is defined specially for socket connection and communication. Among the functions of these two classes, when the commands are run to some stages, some functions in one class will call desired functions from the other class.

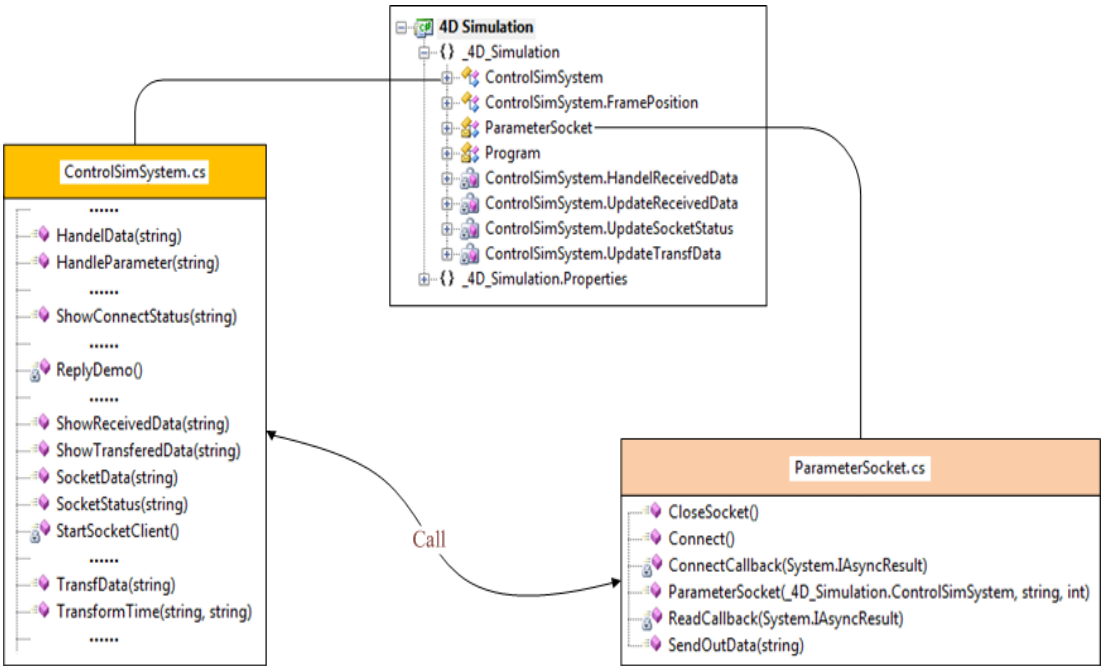


Figure 4.4: Internal functions structure of the CS socket

The complex nested and inter-call workflow is shown in Fig. 4.5. When the connection request is generated, the function "StartSocketClient()" is activated to connect with the DTS socket. In this function, the class "ParameterSocket.cs" is visited for calling its function "Connect()", which tries to connect the specified address and corresponding port of the server by calling the client socket function "BeginConnect()". In "BeginConnect()", the function "ConnectCallback()" is nested. It calls "ShowConnectStatus()" from the class "ControlSimSystem" to show the connection status. In addition, the "ConnectCallback()" calls the function "BeginRead()" when the connection is successful. By embedding another function "ReadCallback()", "BeginRead()" calls three functions from "ControlSimSystem.cs": ShowConnectStatus(), ShowReceivedData() and HandleParameter(). As for the function "HandleParameter()", it calls "SendOutData()" from ParameterSocket.cs" indirectly to feed the received conditions back to the server -- DTS.

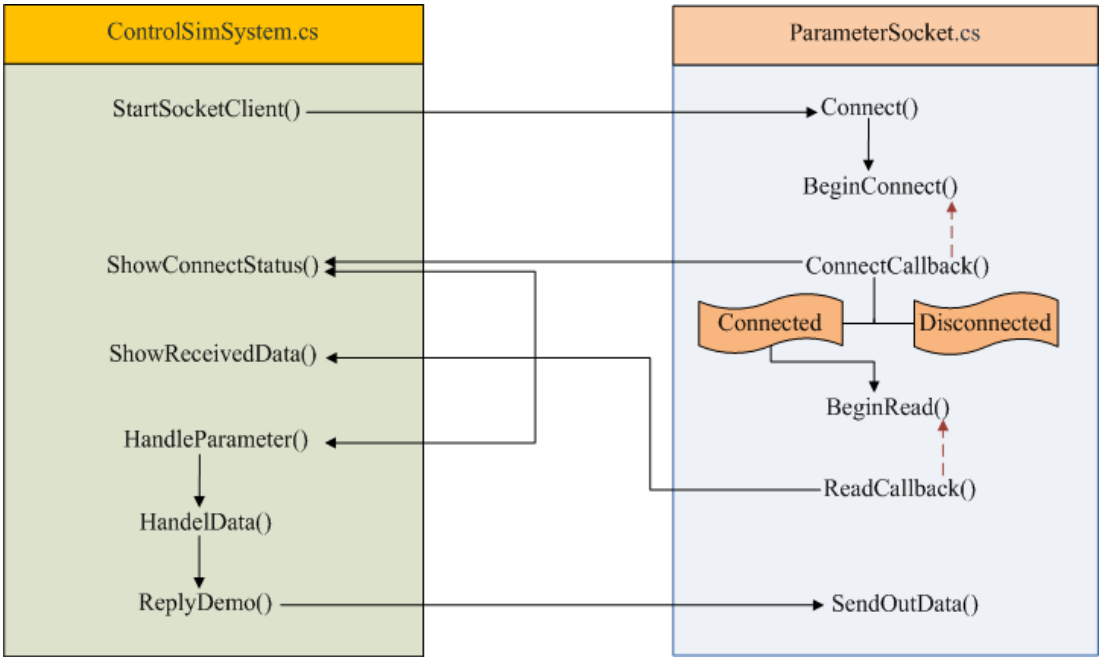


Figure 4.5: Workflow of the CS socket (black arrow:be called; red arrow: be nested)

Table 4.1 introduces the related functions of TCP/IP socket in the CS.

Table 4.1 Functions of the TCP/IP socket class in the CS

Functions	Description
Connect()	Try to connect with the server.
BeginConnect()	Begin an asynchronous request for a remote connection with the server DTS socket.
ConnectCallback()	Start “BeginRead()” when the status is “Connected”.
EndConnect()	Asynchronously accepts an incoming connection attempt.
GetStream()	Return the System.Net.Sockets.NetworkStream used to send and receive data.
BeginRead()	Begin an asynchronous read from the System.Net.Sockets.NetworkStream.
ReadCallback()	Decode and show the received data, feed received conditions back to the server.
GetDecoder()	Obtain a decoder that converts encoded bytes into characters.
EndRead()	Handle the end of an asynchronous read.
GetString()	Decode a sequence of bytes from the specified byte array into a string.
GetChars()	Decode bytes array in the internal buffer into the specified character array.
ShowConnectStatus()	Show connection status in main interface of CS.
ShowReceivedData()	Show received data in CS interface after decoded.
HandleParameter()	Feed received conditions back to the server.
SendOutData()	Try to encode feedback data and send it to the server.
GetBytes()	Encode all the characters in the specified System.String into a sequence of bytes.
NetworkStream.Write()	Writes data to the System.Net.Sockets.NetworkStream.
CloseSocket()	Close the current stream and releases any resources associated with the current stream.

4.2 Data Processing

4.2.1 Real-time Data Processing

The real-time data CS originally gets is in bytes which could be recognized by computers. The CS socket receives and converts these data to strings, and shows them one by one with the PCS running. The converted data should as the same as the data in Fig. 3.4. To demonstrate the data-received process more clearly, the real time for data reception is displayed before every data. The received data with local time information are shown as the Part I in Fig. 4.6.

In case there are some checks needed after the experiment, for the original-received data shown one by one in the Part I of Fig. 4.6, the CS saves them as a file in .txt format without any change. The file is named by the local time automatically. As Part I of Fig. 4.6 shows, every data has been converted to strings. That is required for the device activity description, rather than for the start time and run time of every activity. In the strings, the duration time is shown in unit ms, which should be transformed to the unit *s*. So to make the received data more readable, the system converts the time data in string to time format, and combines the data to the form as Part II in Fig. 4.6 shows. The time conversion is realized by the function TransformTime().

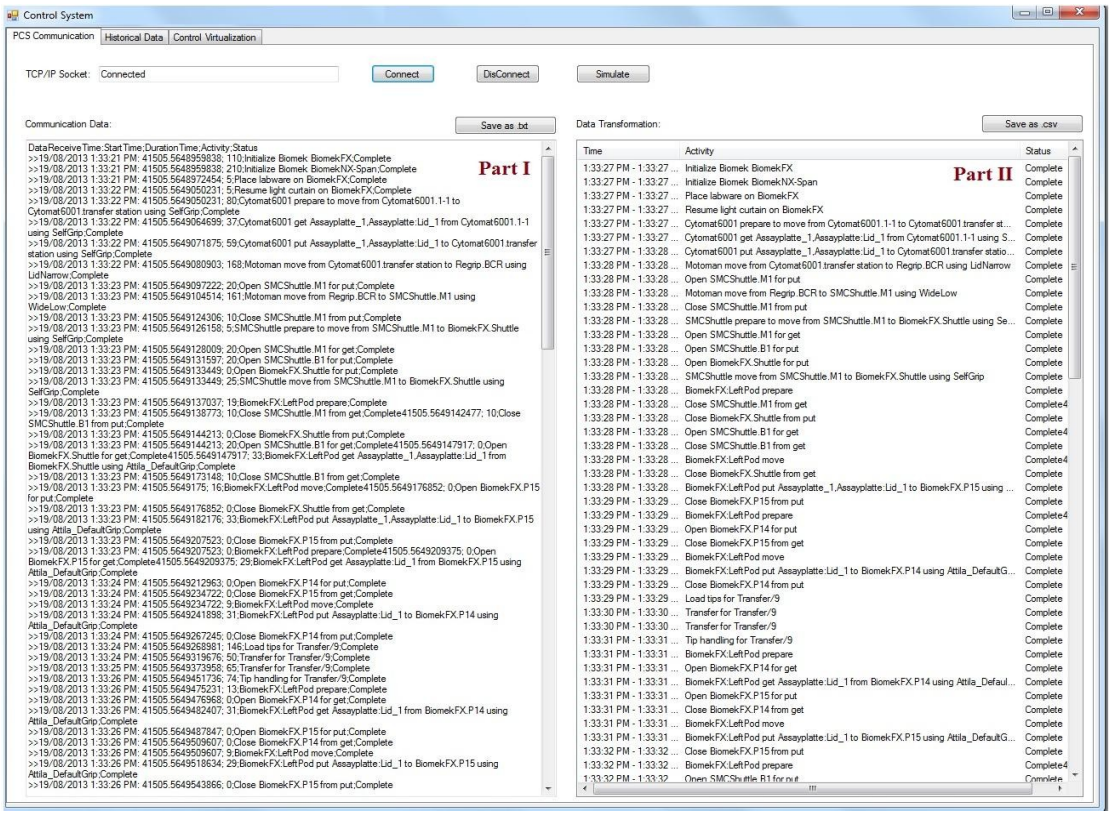


Figure 4.6: Receive and process real-time data – the CS communication interface

The workflow of the function is shown as Fig. 4.7. After transformed, the time factor is converted to the time form with start time and end time. So the original-received data is separated to three parts: Time, Activity and Status. Correspondingly, for the processed data, the system saves it as .csv file, which is also named automatically by the local time.

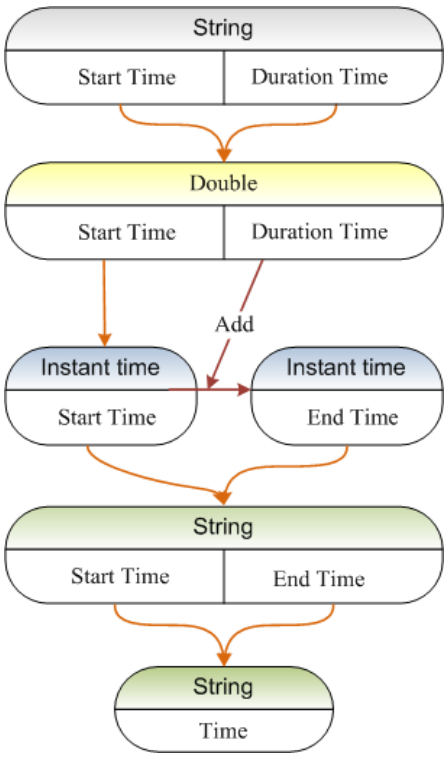


Figure 4.7: Workflow of the function TransformTime()

With the data received and transformed in real time, the CS simulates the workflow data one by one. There is a button "Simulate" in the interface to trigger the next function module – "Control Virtualization" (CV), which converts the data in string to 3D trajectories. At the same time, the button triggers many functions of the CV module, including defining workstation, loading workstation layout, assigning data and listing components' names.

4.2.2 Historical Data Processing

Besides online virtualization for real-time data, at sometimes virtualization for historical data is necessary. For instance, when we need to show the advanced devices and work environments of the life science automation to customers out of the laboratory, it will be much better to show vivid virtualization of LSA experiment workflow in screen, other than to depict them in oral or draw in blackboard. So to

simulate historical workflow data (as whole data virtualization) visually is flexible and helpful for the demonstrations of LSA laboratories.

As Chapter 2 depicts, the workflow data could be saved as .csv in PCS. As Fig. 2.1 shows, the data includes three factors: time, activity and status. The data is generated with the scheduling method running in the Runtime module of PCS. It is complex and lengthy. In this dissertation, not all of the activity data are necessary for the virtualization, e.g. initializing internal parts of devices. So the "Historical Data" module in the CS develops functions to extract and process key information from the historical data in the .csv file (See as Part II and Part III in Fig. 4.8).

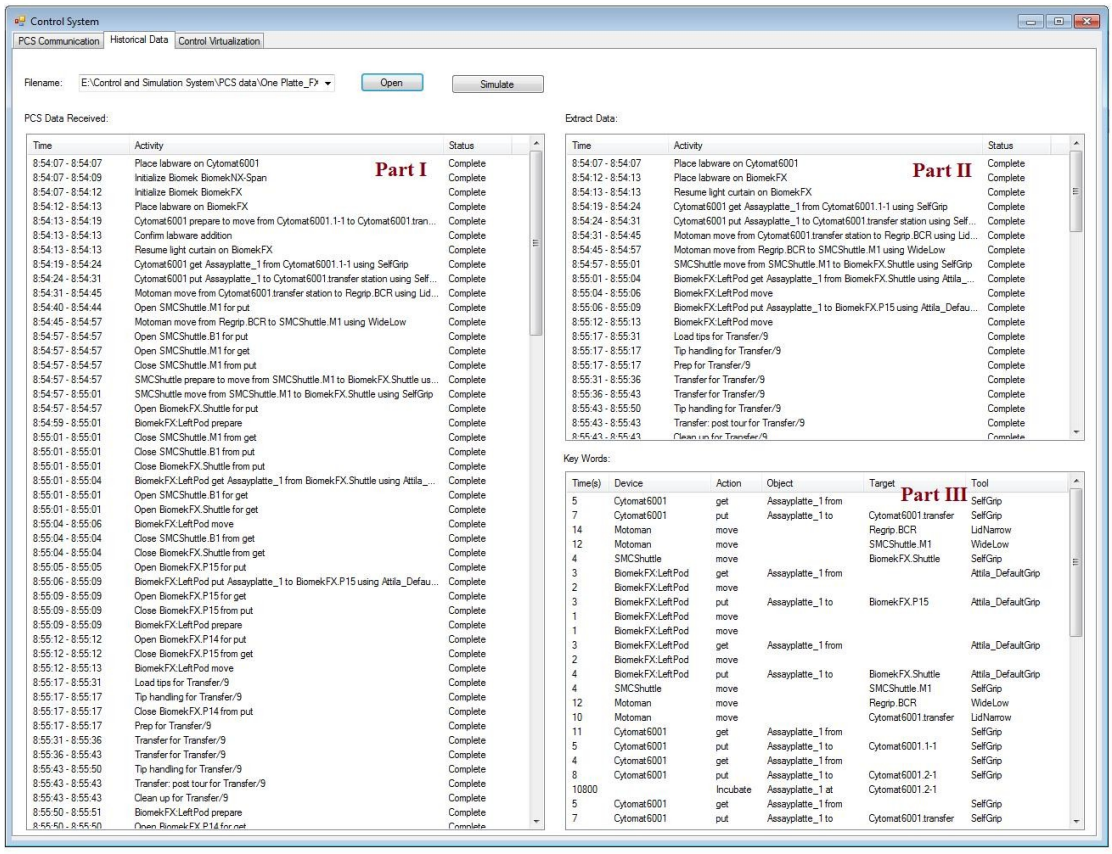


Figure 4.8: Historical data processing

As Fig. 4.8 shows, there is a drop-down box listing the names of .csv files from a defined folder automatically while the CS is started. That is the module for users to choose the historical data file. When the simulated file is open and loaded into the interface, there are three parts to extract and process the imported data. Part I works for showing the integral data in list; the Part II works for extracting useful data from Part I; and the Part III works for extracting key words from Part II. The final key words extraction is for simulating the workflow in shorter time. Except for useless

information and unimportant words, the system could save more time for data assigned and virtualization control in VM. As shown in Part III, to meet the needs of fast virtualization, the key words includes:

1) Time (s) – to get the action duration time (unit: s) for the working device;
2) Device – find the active device in VM directly based on its name;
3) Action: teach the device which kind of actions it need to do, such as get, put, and move, etc;
4) Object: teach the device which object it should work on;
5) Target: teach the device where it should go in the activity (Note: the last target is the beginning position of the next activity);
6) Tool: teach the device which tool it needs to activate and use. Owing to the above key words, the system could easily and fast assign the data to the corresponding device.

After getting the key words, there is a button “Simulate” for toggling to the tab page “Control System”, which works for connecting with the VM, and driving the VM forming the series of movements as the workflow data depicts. The button triggers the functions of the Control System for historical data, including link with the simulation software, load workstation layout and assign the key data to the related components.

4.3 Control Virtualization

Control Virtualization is the most critical module to work for controlling and realizing the virtualization. Behind the module, the 3D simulation software 3DCreate is applied by calling its APIs. CS realizes the virtualization for the workflow data by controlling 3DCreate working. After that, the module sends the virtualization result to PCS via corresponding TCP/IP socket.

Figure 4.9 shows the workflow of the CV module. On one hand, CS calls COM APIs of 3DCreate to load the workstation layout into the embedded-in GUI, list the names of components of the workstation layout, and assign the workflow data to the related components for creating most of their behaviors. On the other hand, the system calls Python APIs of 3DCreate to compile behavior properties of devices, trajectories of robots, and so on. At last, the system creates movement sequences for the components in the layout. All these behaviors, trajectories and sequences form whole workflow movements, which could be saved into a layout as .vcm, or recorded as .pdf animation. Finally, the CV module feeds the virtualization result in the form of .pdf back to the PCS via TCP/IP socket. The result is the basis for the PCS users to decide whether to drive the workstation running in the LSA laboratory.

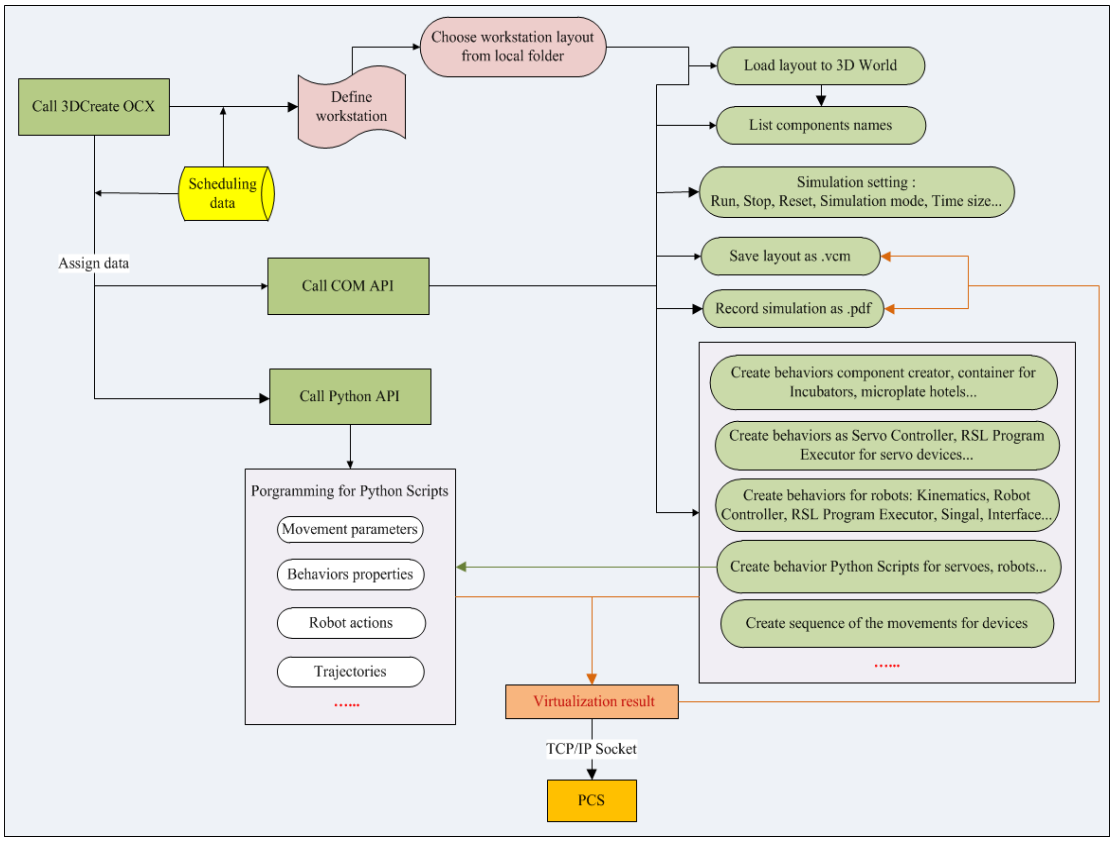


Figure 4.9: Workflow of the module CV

Control Virtualization module is shown as Fig. 4.10. There are six modules in the interface:

I. Virtualization demonstration;

II. Import workstation layout;

III. Simulation setting;

IV. Save layout;

V. Send virtualization result (animation) to DTS;

VI. List components' names. In the module I, there are three key functions referred: link with the VM, assign data, and show virtualization result. The module is shown in the form of the embedded VM GUI.

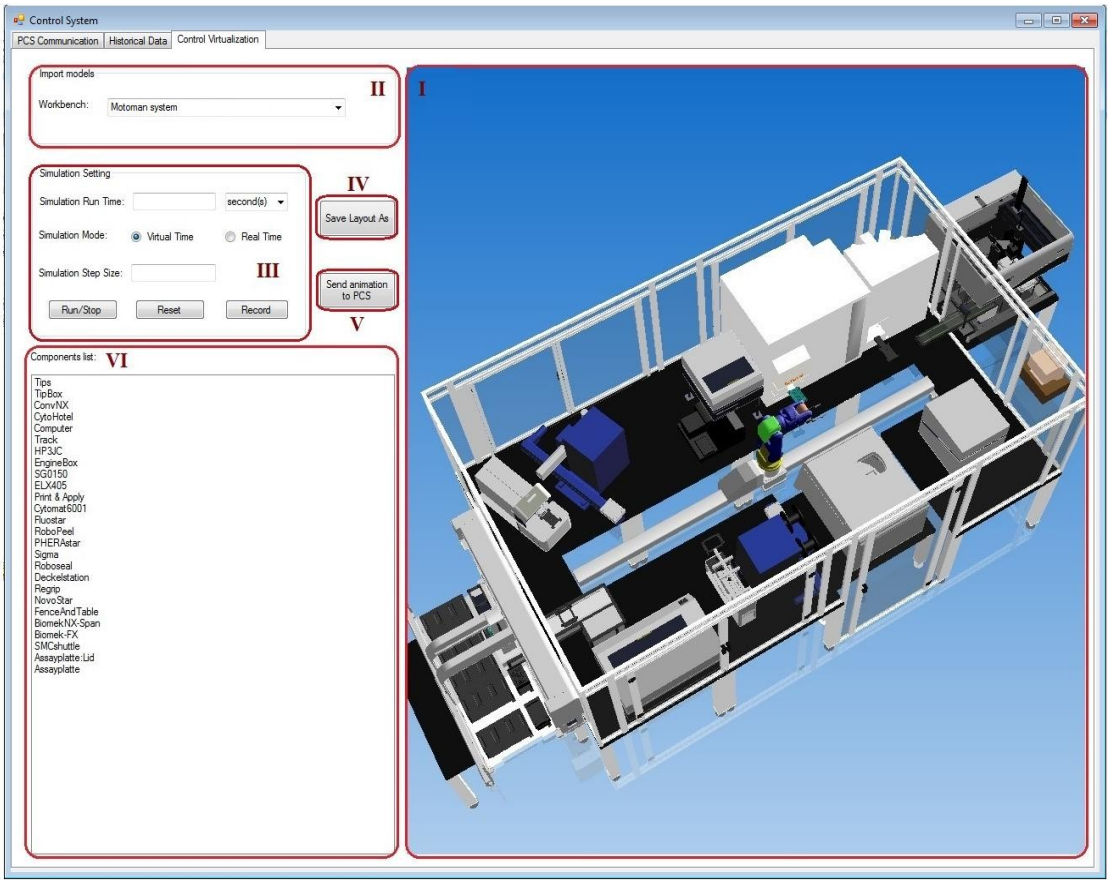


Figure 4.10: Control Virtualization

4.3.1 Link with the Virtualization Model

Either for virtualization on real-time data or historical data, once the tab page "Control Virtualization" is triggered, the Control System will connect with the simulation software – 3DCreate at soon. As Fig. 4.11 shows, when the data processing module sends the request of virtualization, the CV module is activated. Then the module tries to start 3DCreate program by calling its COM APIs.

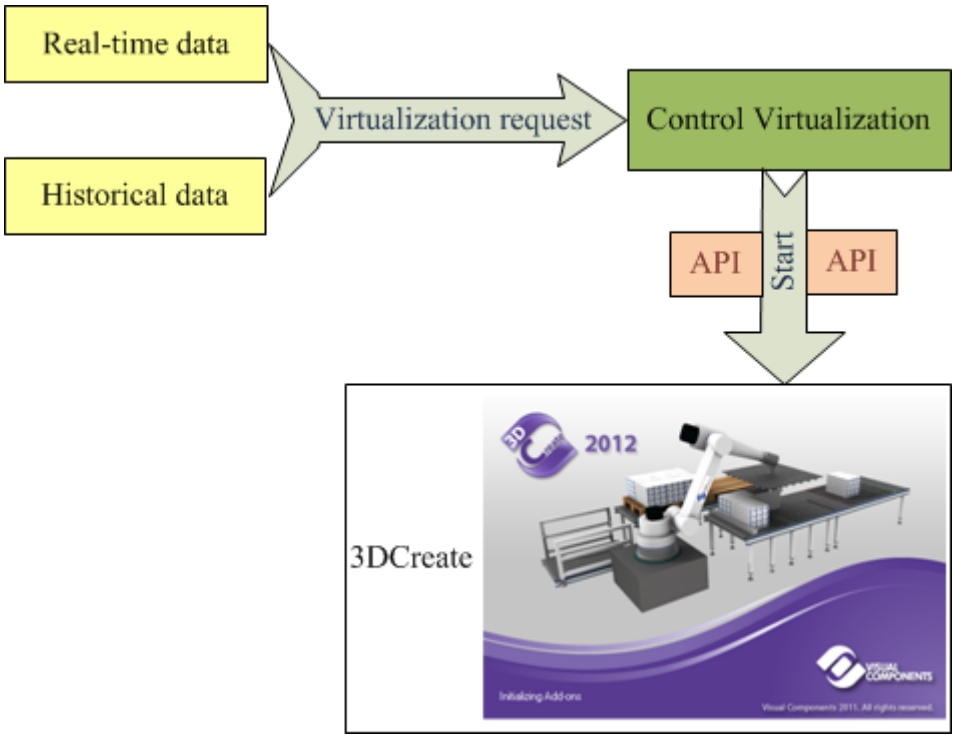


Figure 4.11: Link with the VM for virtualization request

There are two methods to start the 3DCreate program via its APIs: One is to open and expand the software fully just as it runs generally; the other one is to run the software by embedding its GUI into the system. The first one is realized by calling corresponding program running commands of COM API, and the second one is by calling a COM component of 3DCreate. At the stage of generating and checking trajectories in 3DCreate, as well as the system-test stage, the first method is applied frequently. For the finished system, the second method is applied for system simplifies and more vivid virtualization. The embedded GUI, which is also called 3DWorld in 3DCreate, is presented as the Part I of the "Control Virtualization" interface in Fig. 4.10. It is realized by calling the application component OCX of 3DCreate. On the back of the 3D World interface, there are strong functions of 3DCreate.

While the system links to 3DCreate and jumps to the control system interface, 3DCreate OCX is called, and the 3D World is embedded into the Demo Platform. The related layout is loaded to the platform. Once experiment data is assigned to the devices in the layout, the platform can show the virtualization result directly while the simulation setting module is triggered. The platform makes the virtualization on real-time data faster and more vivid.

To make sure no exception in the controlling process, there is just one 3DCreate program permitted to run. So before the software is started, the system searches from the computer processes whether 3DCreate program is running. If it is, the system will force to end the software process, and restart it by calling its COM API. As shown in Fig. 4.12, the judgment process is finished by the function IsRun(). For the GUI embedded method, while the CS interface is closed, the 3DCreate process will also be ended by the corresponding COM API command.

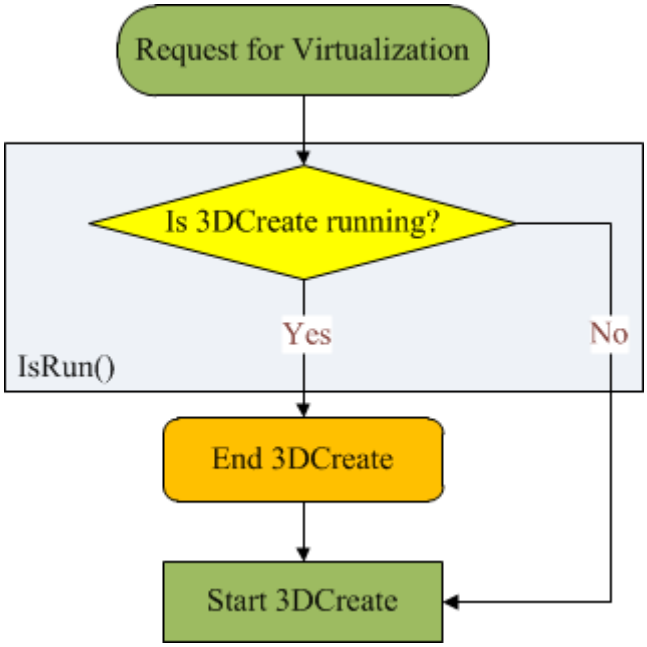


Figure 4.12: Workflow of the reaction for virtualization request

4.3.2 Import of the Workstation Layout

After open 3DCreate or embed its COM component for application window in CS, the system works following with the workflow data. Based on the extracted data, firstly, the system judges which workstation the workflow data works on. Then it searches the workstation layout file from a specified computer disk, and loads it into the 3DCreate GUI.

The workflow for importing the workstation layout is shown as Fig. 4.13.

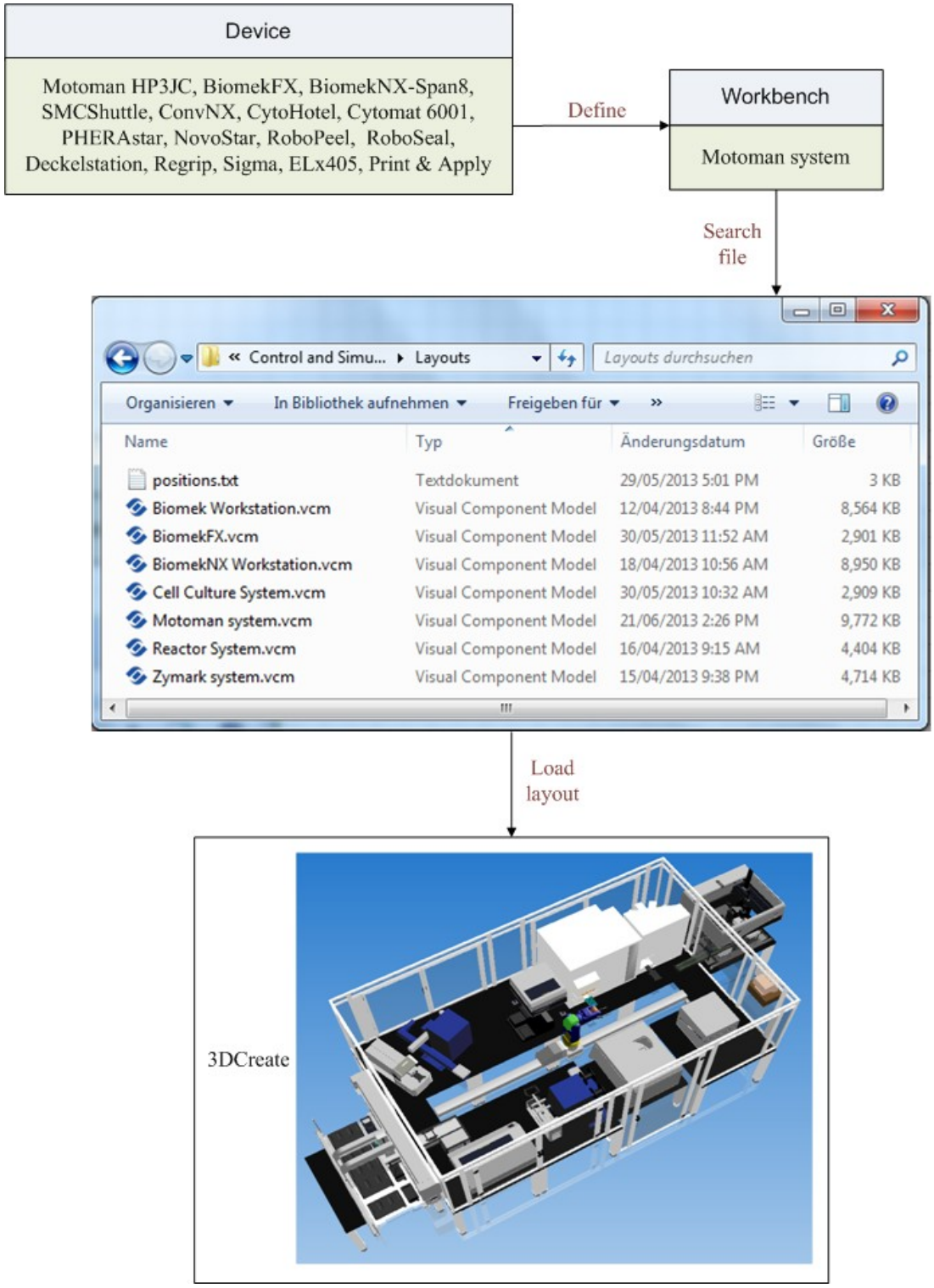


Figure 4.13: Import workstation layout

(a) Define workstation

There are many specific devices in every LSA workstation. Table 4.2 lists the workstations at celisca and their corresponding devices. Based on the devices information, it is easy to define which workstation the workflow runs on. For example, if there is the word “Motoman” in the parameter “Device”, it is surely the working workstation is “Motoman system”. So the workstation layout CS should load could be determined by the key word “Device” in the workflow data.

Table 4.2 Workstations and their devices at celisca

Workstation	Device
Motoman system	Motoman HP3JC, BiomekFX, BiomekNX-Span8, SMCShuttle, ConvNX, CytoHotel, Cytomat 6001, PHERAstar, NovoStar, RoboPeel, RoboSeal, Deckelstation, Regrip, Sigma, ELx405, Print & Apply
Zymark system	Zymark XP, Adapter f Turbovap, Analysenwaage, Autodose, Büchi Syncore, HPLC, PAL, CEM Discover, CTC Analytics
Cell Culture System	BiomekNX-Span8, Vi-Cell® XR, Zeutrifuge Vspin Velocity11, Cytomat 6001, FX Device Controller, Port Selection Valve, Cooling box, MasterFlex ® Console Drive
Reactor System	Biomek 2000, HPMR50-96, ORCA robot, HPLC system, Time-of-Flight mass spectrometer, CTC Analytics

(b) Search layout file

There are many workstation layouts saved in a specified folder. They are the original layouts with stationary components, which compose to a workstation and have no any trajectory.

When the workstation name is defined, the system searches its layout file at once from the specified folder, and shows the name in the system. For 3DCreate, the layout file ends in .vcm as its extension.

(c) Load layout into VM

Before loading a workstation layout into 3DCreate GUI, CS checks whether it has any component. If it has, the system clears it. When the GUI is surely vacant, CS executes the definition and search functions, and loads the layout into 3DWorld of 3DCreate.

4.3.3 List Components' Names

The module VI in the Fig. 4.10 is for listing the names of all components in the layout. While the workstation layout is loaded into the GUI, all names of components are listed in the box. Whenever there is any change in the GUI, the contents in the listing box will be refreshed.

The module is realized by the COM API command, which gets the "name" property from all components. In this process, CS gets the component one by one, and at the same time, to get the name property from the component properties.

4.3.4 Assign Data

The assign data module works to transform the workflow data in strings to trajectories of 3D models in VM. It is the most critical step in the data virtualization process. In CS, the "Assign Data" works as an invisible module behind the Part I in Fig. 4.10. Once the "Simulate" button in either "real-time data processing" module or "Historical data processing" module is clicked, the assign data module is called and triggered with the Part I activated.

Although it works for data assigned literally, it is a complex process to create behaviors, parameters and trajectories in 3DCreate by programming. Some of these tasks are developed by calling COM API, such as joints, features and parameters created; and some are written to the Python script behavior by calling both COM API and Python API, such as behaviors, trajectories created. The first development way applies Visual C# to call relative methods and properties of 3DCreate COM API, which includes type libraries as vc3DCreate, vcCOM and vcCOMecat, etc. The second way applies Visual C# to create behavior Python script by COM API, and python language to add movement parameters, behaviors properties, robots actions and trajectories, etc. into the Python script by calling Python API.

When one of the data processing modules triggers the CV module, CS finds and loads the workstation layout into 3DWorld. After the layout is loaded into 3DCreate GUI, the data assignment is started automatically. In this course, the system assigns the processed data to the corresponding components of devices in the workstation. For the real-time data, the system assigns them one by one with the data received, generates components' trajectories and run them in 3DCreate in real time, and finally forms a whole series of movements until the scheduling experiment is finished. For the historical data, the system assigns all the data to the corresponding devices in one time,

and then generates the whole virtualization when the data is assigned fully.

4.3.5 Data Analysis

Generally, there are four factors in one workflow data: run time, working device, object, and activity. As for the "activity", it refers to some factors as action, start point, destination, tool and purpose, .etc.

To assign the activity of every data to devices, the system needs to separate it to many factors and actions, which the related 3DCreate API could be called to generate. Table 4.3 shows some workflow data of LSA experiment.

Table 4.3 Parts of the workflow data of a LSA experiment

No.	Time	Activity
1	8:54:07 - 8:54:07	Place labware on Cytomat6001
	……	
2	8:54:31 - 8:54:45	Motoman move from Cytomat6001.transfer station to Regrip.BCR using LidNarrow
	……	
3	8:54:57 - 8:55:01	SMCShuttle move from SMCShuttle.M1 to BiomekFX.Shuttle using SelfGrip
4	8:55:01 - 8:55:04	BiomekFX:LeftPod get Assayplatte_1 from BiomekFX.Shuttle using Attila_DefaultGrip
	……	
5	8:55:17 - 8:55:31	Load tips for Transfer/9
	.....	
6	8:55:31 - 8:55:36	Transfer for Transfer/9
	.....	
7	11:57:55 - 11:58:53	Issue command to Teleshake1 (1:00)
	.....	
8	11:59:08 - 11:59:12	Open PHERAstar.R for put
	.....	
9	11:59:17 - 11:59:28	Motoman move from ConvNX.outer to PHERAstar.R using WideReverse
	.....	
10	11:59:28 - 11:59:32	Close PHERAstar.R from put
11	11:59:32 - 12:00:42	Issue command to PHERAstar (1:47)
12	12:00:42 - 12:00:47	Open PHERAstar.R for get
13	12:00:47 - 12:00:59	Motoman move from PHERAstar.R to ConvNX.outer using WideReverse
	.....	
14	12:00:59 - 12:01:03	Close PHERAstar.R from get

Take the data No. 1 for an example, the activity “Place labware on CytoHotel” should be separated to many actions (as Fig. 4.14 shows). It includes five steps of actions, which refer to different commands, such as import, translate and rotate the component. For the data No.2, No.3, No.9 and No.13 in Table 4.3, they all have factors as device, action, start position, destination and tool.

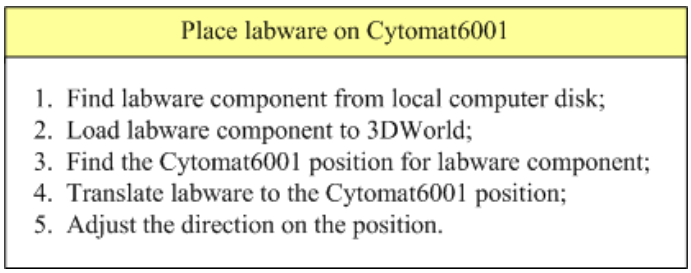


Figure 4.14: Actions contained in the workflow data No.1

Figure 4.15 shows the factors separated from the data No.2. From Fig. 4.15, the activity depicts that the “Device” Motoman takes an action “move”, from the position Cytomat6001.transfer, to the position Regrip.BCR. In this course, Motoman applies its tool -- gripper with the condition “LidNarrow”.

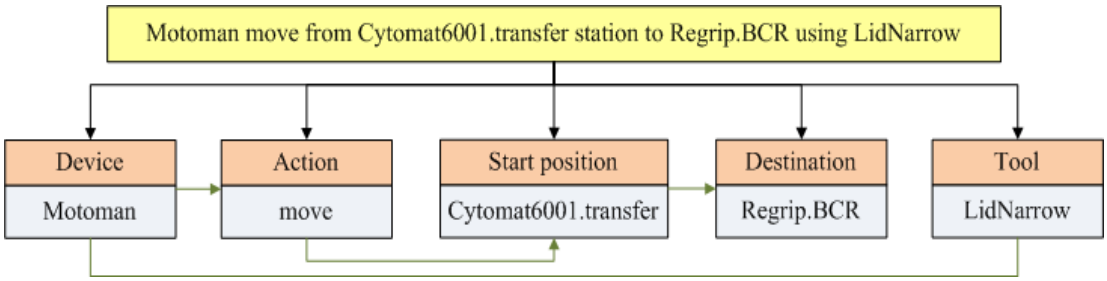


Figure 4.15: Factors in the workflow data No.2

4.3.6 Information Mining and Collection

Just knowing the objects of the factors is not enough to realize the data assignment. More detailed information is needed for the CS to start assigning data, such as the motion method of the robot, the coordinates of the start position and destination, etc.

In fact, there is a large number of information implicit in every factor of the data. Parts of it are the ones that data assignment needs. They could supply sources and basis for the data assignment. They are critical in the conversion process from strings data to 3D motions. So before assigning data to the VM, it is necessary to mine and collect the important information from the factors of every data.

Table 4.4 shows a case to mine some important information from the factors of the data ② in Table 4.3. From Table 4.4, for the factor "Device" Motoman, the robot original position, tool center point (TCP) coordinates, robot joints value, and the kinematics of the robot, etc could be mined for next system work; for the "Action", many motions could be designed for the robot; as for the factors "Start position" and "Destination", from the names of the devices, the target frames on them could be mined, and at the same time, the respective coordinate values of the frames in the robot parent coordinate system could be extracted for the robot moving; the factor "Tool" supplies important condition for the tool of the robot. It tells the robot Motoman which gesture its tool should apply to work on the target object. All these information could be extracted from the related 3D components in the layout.

Table 4.4 A case of information mined from workflow data

Factor	Content	Information referred
Device	Motoman	Original position of the robot, Tool Center Point (TCP), Tool direction; joints' values, kinematics…
Action	move	Many motions with tool targets should be created.
Start position	Cytomat6001.transfer	the first place robot Motoman should go; the coordinate information of the target position; the frame of the tool target; second action for Motoman: pick up …
Destination	Regrip.BCR	the destination position robot Motoman should go; the coordinate information of the end position in the activity; aimed frame for tool; third action for Motoman: put down
Tool	LidNarrow	The working condition of the tool Gripper

After CS mines all information the assignment process needs from one data, the system collects them together for the related device. The full-collected information tells CS which functions it should realize in VM. The collection work does preparation for the full assignment to the device, which will be taught to convert all the mined information to some motions.

4.3.7 Data Assignment

The trajectories of devices are generated in the simulation software when CS sends the data to VM and drives it to create corresponding movement routes via the APIs of VM. This will be done by the third tab page “Control System”.

This is an API programming and application process to develop the simulation software 3DCreate secondarily. When the CS receives the full information virtualization required for every data, the system starts to assign the information to the related components in the layout. In this process, based on the information, the module calls and programs COM API of 3DCreate to generate corresponding 3D motions for the components, and form 4D trajectories which could be run in VM.

To shorten the trajectories generation time for the virtualization on real-time data, every possible trajectory is created in 3DCreate. When the data is separated to detailed information, the system calls corresponding trajectories to realize its 3D simulation. So in the data assignment, the system just needs to get and separate data in real time, and then based on the extracted workflow in the data to call its trajectories in 3DCreate. This concept of data assignment greatly saves time in trajectory generation, and ensures the virtualization speed.

In the data assignment, at first, the CS searches the component from the layout according to the “Device” name. Then the system assigns the corresponding commands to the component. If the component is a robot or a servo, the CS finds its executor and controller, and creates or calls motions for them. If the component is stationary, the system set its corresponding properties as the data requires.

Figure 4.16 shows a case to assign a data to the related components in VM. For the data "Motoman move from Cytomat6001.transfer station to Regrip.BCR using LidNarrow", at first, based on the factor "Device", the system finds the component "Motoman" in the GUI of 3DCreate. Then the system extracts the component's original status information (including original position, tool gesture (narrow or wide), tool location, configuration method, joints' conditions, etc), and the start position for its first movement. Between original position and start position of the tool frame, there are many motions created for the tool frame of Motoman. When the tool frame in Motoman arrives at the position "Cytomat6001.transfer", the system goes into the component Motoman to check whether its tool gesture is right for next movement. In this data, the tool should be as "LidNarrow" for grasp the labware. So if the tool is not in gesture "LidNarrow", the system adjusts its gesture and position to narrow at the lid height. That is realized by remote calling the routines of the gripper SG0150. The SG0150 routines are the ones having corresponding motions, which change the gripper sizes, gestures and directions to grasp or release object. Then, the CS calls the action statement "pick" to get the labware on Cytomat6001.transfer, and creates a motion for the labware "up". All the above motions, remote routines and grasp actions form a workflow trajectory "from_Cytomat6001.transfer" of Motoman. After that, a routine "*NarrowtoRegrip*" of Motoman is called to realize the action "move" to the destination position "Regrip.BCR" in the narrow gesture of the gripper SG0150.

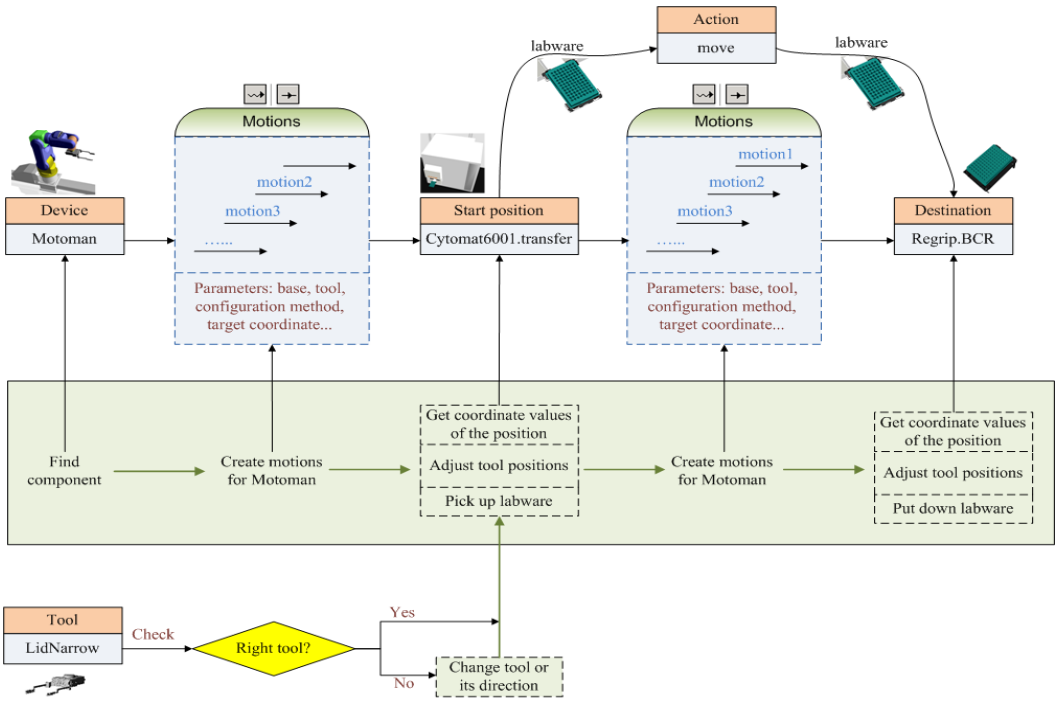


Figure 4.16: Workflow of the data assignment

Algorithm 4.1 A case of data assignment in the CS

Data: Motoman move from Cytomat6001.transfer station to Regrip.BCR using LidNarrow

1: *Find component "Motoman"*
2: *Find executor and controller of "Motoman"*
3: *Find the main routine in the executor*
4: **Statement #1:** *Define tool location for "Motoman"*
5: *Search the action in "actions"…*
6: **if** *the action is "move"* **do**
7: *Search the start position in the "fromDevice" group…*
8: **if** *the start position is "Cytomat6001.transfer"* **do**
9: **Statement #2:** *Call routine "from_Cytomat6001.transfer"*
10: *Search the tool in the "tool" group…*
11: **if** *the tool is "LidNarrow"* **do**
12: **Statement #3:** *Call remote routine "SG0150_ForNarrow" to from_Cytomat6001.transfer".*
13: **Statement #4:** *Create motion for tool frame to Cytomat6001.transfer.*
14: **Statement #5:** *Create action statement "pick" to "from_Cytomat6001.transfer".*
15: **Statement #6:** *Call remote routine "SG0150_Narrow" to "from_Cytomat6001.transfer".*
16: **Statement #7:** *Create motion "up" from Cytomat6001.transfer.*
17: *Search the destination in the "toDevice" group…*
18: **if** *the destination is "Regrip.BCR"* **do**
19: **Statement #8:** *Call routine "NarrowtoRegrip" to the reach position near Regrip*
20: **Statement #9:** *Create action statement "put down" to "NarrowtoRegrip".*
21: **Statement #10:** *Create remote routine "SG0150_ ForNarrow".*
22: **end if**
23: **end if**
24: **end if**
25: **end if**
return *a 3D trajectory*

(a) Assign real-time data

The assignment for real-time data in the CS is triggered by the command "Simulate" in the "PCS Communication" tab. When the PCS workflow data enters into the CS, the system assigns every data to the workstation layout for corresponding trajectories generation one by one, and shows the 3D movement in its interface synchronously. In the virtualization process, it is feasible to add new trajectories into the movement, and show the new ones with the moving going. If there is some interruption in the data transmission, the virtualization for the received data will not stop until there is no data

left. Once the data is gotten again, the data assignment and virtualization will go on from the breakpoint. When the data is received totally, the trajectories will also be generated in a specified sequence. After that, the virtualization is finished, and the whole movement for the experiment workflow is generated.

In a word, the real-time data assignment works in real time, and its virtualization result is also generated and shown with it in real time. The full virtualization result could be gotten until all the data is received and assigned. The processes for real-time data transmission and virtualization work as Fig. 4.17.

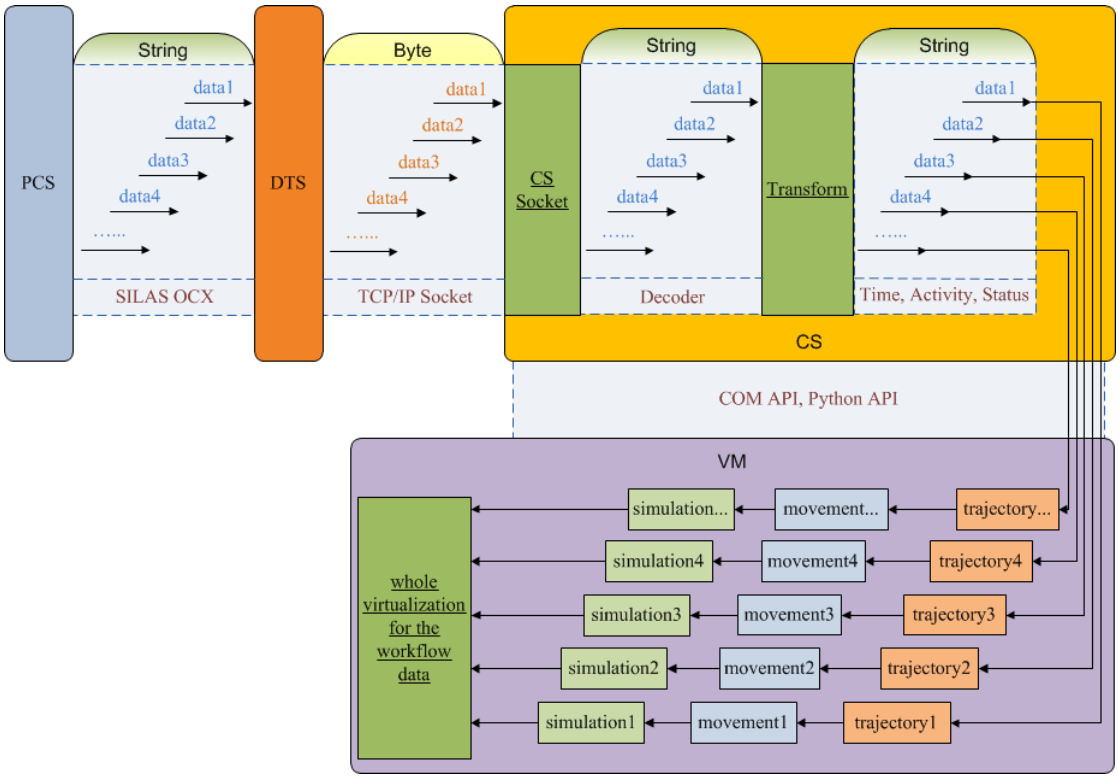


Figure 4.17: Real-time data transmission and virtualization

(b) Assign historical data

The assignment for historical data is triggered by the command “Simulate” in the “Historical Data” tab. For the historical data, it is a full workflow data for a whole experiment. So the assignment for historical data is done in one time, and the full virtualization result could be generated after that.

The assignment process of historical data also runs in the order of the data one by one, which makes decision of the movement sequence. Finally, all the motions for the data virtualization form an integral workflow layout, which can run by the simulation

setting module of the Control System. The processes for historical data virtualization work as Fig. 4.18.

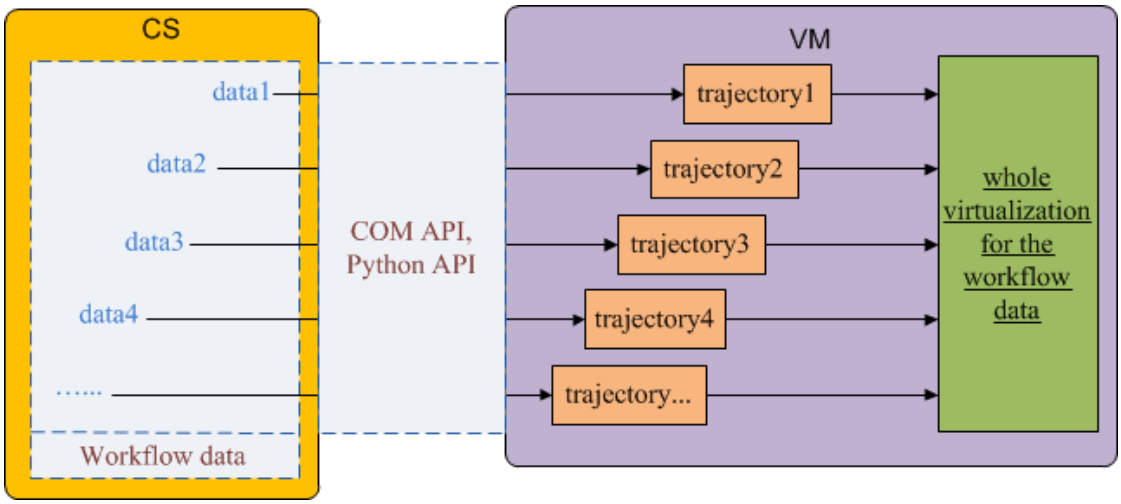


Figure 4.18: Historical data virtualization

4.3.8 Motions Sequence

The workflow data implies information about the sequence of device motions. With the motions for one data being generated, the CS creates sequence for it and the following one data. After the sequence for all data is created, a complete and coherent virtualization for the experiment workflow data is produced.

The sequence of components' motions in 3DCreate is generated by calling remote routines. Correspondingly, the CS calls the COM API to create new statements for remote routines. Taking the virtualization on Motoman system as an example, the CS defines the robot Motoman as the main moving robot. When there is a motion of other component generated, the Motoman robot controller will create a statement to call the remote routine of the motion, and another statement to wait for the motion being finished. After that, the call for remote routine is over, and then following with it, the next motion of the Motoman is generated and added to the sequence. With the whole data is assigned, all the motions of other components are inserted to the Motoman movements. At last, an orderly motions sequence is generated in the Motoman executor. The motions with their sequence constitute an integrated 4D movement virtualization on the workflow data.

4.3.9 Simulation Setting

To control the virtualization demonstration, it is more flexible if there is a simulation player to operate the virtualization process. The module “Simulation Setting”, which is shown as the Part III in Fig. 4.11, is just the one developed as a player. The module has all functions for simulation setting and controlling as 3DCreate. It is also realized by calling the corresponding COM API of 3DCreate. The functions are depicted as followings:

(1) Simulation run time: There is an input box for users to set the simulation time length. Many time units could be chosen, e.g. hours, minutes, seconds or years, months, days. The default unit is second(s). The module works to set the property “SimulationRunTime” in 3DCreate.

(2) Simulation mode: There are two simulation modes supplied for users to choose: virtual time mode and real time mode. The former is dependent on the computer speed, and the later means the time in the simulation is the one in real time.

(3) Simulation Step Size: Both of the time modes can be accelerated and decelerated by setting the step size for the simulation.

(4) Run/Stop: The module works to switch the functions between start and pause the simulation. It is realized by setting the property “simulation running” in 3DCreate.

(5) Reset: The module resets the simulation to its initial state. It starts the corresponding simulation command of 3DCreate COM API.

4.3.10 Post Processing

When the virtualization on experiment workflow is finished, the CS needs to back it up and feed the virtualization result back to the PCS. The related post processing for the virtualization result includes the followings:

(1) Save layout as 3DCreate format

The module saves the layout with the simulation movement as a 3DCreate file (*.vcm), which is backed up for being reused and modified. In this module, the CS gets the command from COM API, and then executes it in 3DCreate.

(2) Record as animation file

In 3DCreate, it is feasible to record the virtualization result as an animation in .pdf format, which could show the animation vividly in flexible 3D views and sizes in Adobe Reader. The module could not only save historical data virtualization result into a whole animation, but also save the online virtualization into many parts, which could be sent and shown in PCS synchronously. The function is also realized by setting corresponding property of the 3DCreate COM API.

(3) Send animation to the DTS

In the online virtualization, there is a requirement to send the virtualization result instantly back to DTS, which works at the PCS side. As the above context depicting, in the online virtualization process, the virtualization result is separated into many parts of .pdf animations step by step. So in this module, the CS applies the TCP/IP socket technology to send these .pdf parts one by one to the DTS with their gradual generation.

As Fig. 4.19 shows, once the animation file is generated, the CS socket gets an encoder in Unicode form to covert the .pdf file into datastream for transmission from the CS to the DTS. While there is an interruption in the communication, the CS socket will try to call the DTS socket again. Until the communication is created once more, the CS socket starts to send the animation files from the breakpoint.

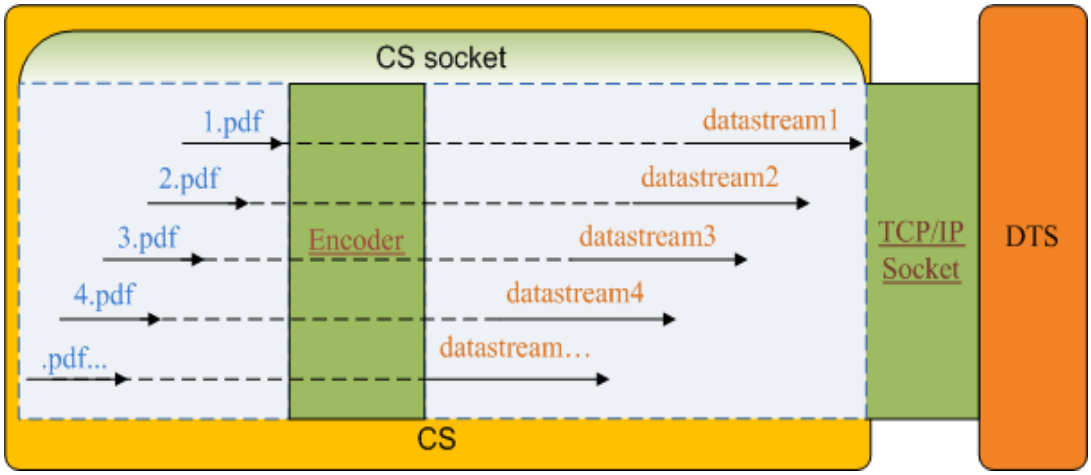


Figure 4.19: Virtualization result transmission

4.4 Discussions

The Control System realizes the data conversion from text to three-dimensional kinematics and dynamics in real time. It links all other modules as a whole

virtualization system, and controls its working for realistic LSA workstations and laboratories.

However, the CS itself has no simulation functions just by programming. So it requires a third 3D simulation tool to demonstrate the converted kinematic and dynamic data graphically, and control the tool to form a virtualization result in 4D. The selection and applications of the 3D tool are presented in detail in chapter 5.

Chapter 5 Virtualization Module

5.1 Introduction

The Virtualization Module is the last and very important module in the virtual virtualization for the LSA workflow. It should prepare the workstation layout for the VS, and supply strong 3D simulation functions for nice virtualization effects. In addition, the module should have the feasibility for CS to control and drive. To meet the requirements of the VM, the dissertation applied 3D virtualization technology. The technology refers to some 3D simulation software, which has strong virtualization functions and strong API for CS calling.

Currently, there are lots of popular 3D CAD software and animation-making software. In the dissertation, besides of the general animation functions, the simulation tool should have strong functions to create kinematic trajectories and dynamic actions for mechanical models. From the demands of the virtualization, some related mature 3D simulation tools are compared in Table 5.1.

Table 5.1 Comparisons of 3D simulation software [117]-[122]

	Solidworks	**3DCreate**	**3ds Max**	**Maya**	**Easy-Rob**
Main application areas	mechanical engineering	Digital simulation	Games	Film	Robot simulation
Rendering speed	good	Excellent	slow	slow	slow
Animation tools	good	Excellent	Very good	Excellent	simple
Modeling	Excellent	week	Excellent	Very good	week
CAD data communiation	Excellent	Excellent	good	good	Not too much
API & supported language	API; Visual Basic for Applications (VBA), VB.NET, Visual C#, Visual C++	COM/Pyt-hon API; Visual Basic, Visual C#, Visual C++ ,C++ Builder	SDK; Visual C++	API; Visual C++	API; Visual C++
Multi-kinematics	No	Yes	Yes	Yes	Yes
Robot libraries	No	Yes	No	No	Yes

Integrating factors as virtualization object, graphics, rendering speed and application areas, .etc, we chose 3DCreate for the 4D virtualization. The software 3DCreate is the premium package of the Visual Components' software family. It could work for multi-robots kinematics simulations synchronized, and supply very good graphics and high speed rending [123]–[125], which is necessary for online virtualization. The software users could create new simulation components from existing 3D CAD data by adding custom functionality with behaviors and parameters, and simulate complete factory layouts. What's more, the software supplies strong COM API and Python API [126]–[128] for application developers. Fig. 5.1 is a case of 3D manufacturing virtualization with 3DCreate. The process for creating a 3D virtualization using 3DCreate is as followings:

i. Prepare 3D models for 3DCreate;
ii. Create component in 3DCreate;
iii. Teach robots how to work.

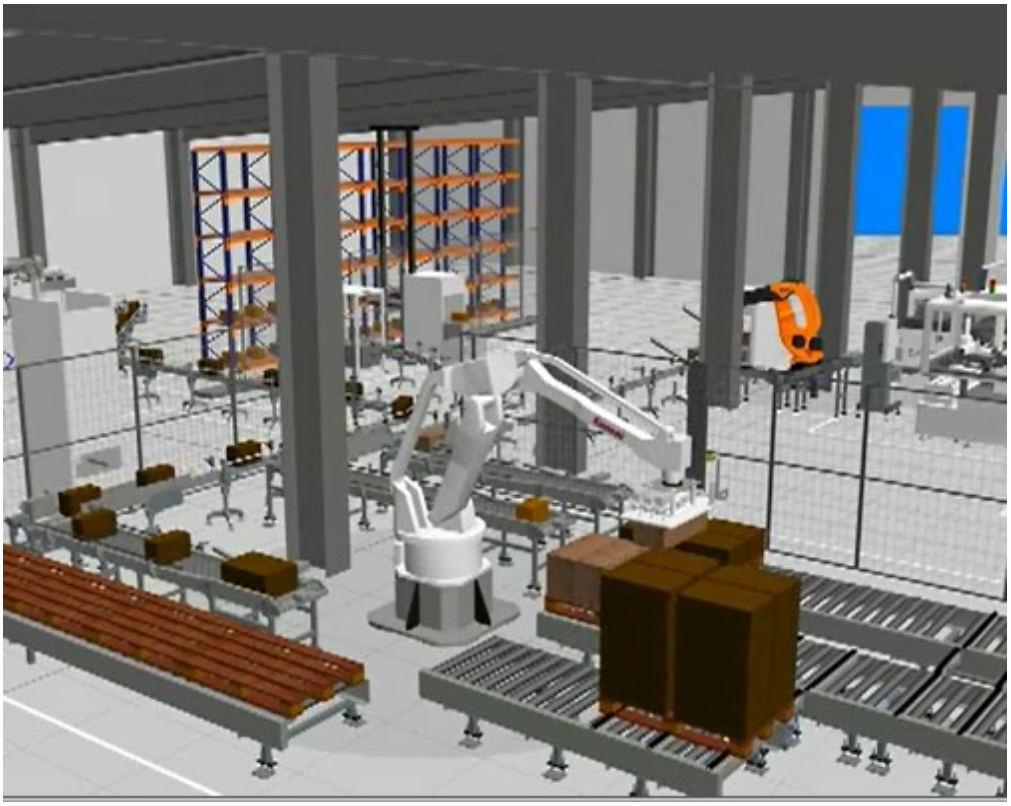


Figure 5.1: 3D manufacturing virtualization with 3DCreate

5.2 Modeling

Due to 3DCreate is simulation software with week 3D modeling functions, there is other 3D CAD modeling software needed. In this dissertation, SolidWorks, an outstanding 3D CAD tool, is utilized to build the 3D models of devices mounted in the workstations at celisca, and to describe their mechanism characters. That is the preparation of models for components creation in 3DCreate.

The models created by SolidWorks are in real sizes and structures. They do not have characters of joints, interfaces among models, behaviors, and movement trajectories, etc. They are exported as step format, and transferred to components in 3DCreate by adding kinetic properties and behaviors.

5.3 Create Components

To make a 3D model moving and working together with other models in 3DCreate, the model should be converted to a component, which is a 3D graphical representation of a machine/product with simulated behaviors.

5.3.1 Component Structure

Technically, a component is a "container" of different virtualization objects, including frames, features and behaviors, as well as their relations. Some of the objects define "the looks", while others are the behaviors and interaction with other components in the virtualization.

The relations among the objects form a tree structure, which is shown as Fig. 5.2. As Fig. 5.2 shows, a component is composed of nodes and parameters. For the nodes, they have three factors: features, interface and behaviors. They form the kinematics characters for the component. As for the features, they consist of geometry factors of the 3D model, such as points, lines and faces [129].

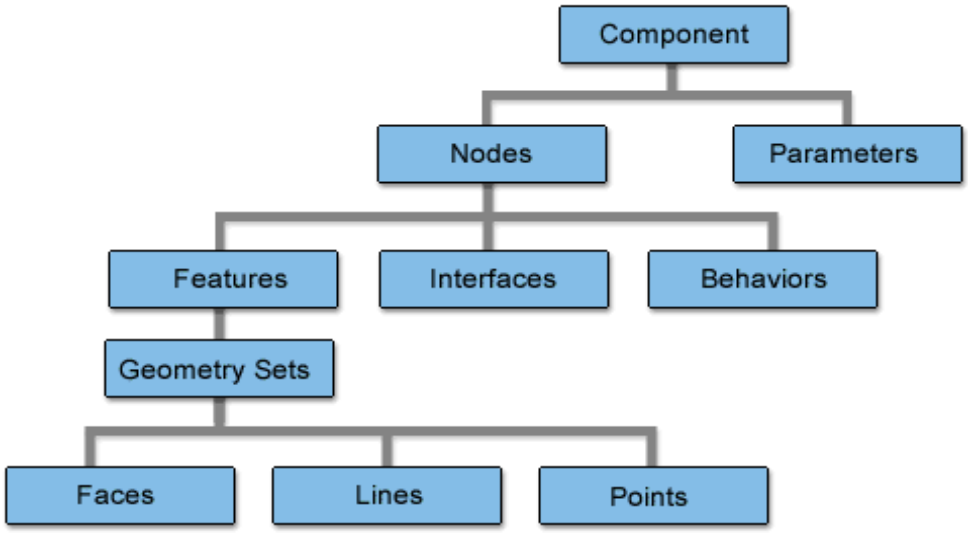


Figure 5.2: Internal frame of a component in 3DCreate [129]

5.3.2 Organizing the Geometry

Generally, a component includes many moving parts, which are called nodes. The features and behaviors of a node are based on geometries. Therefore, when a 3D model imported into 3DCreate, the first work of component creation is to break the geometries of the model into many logical features, and then organize the features to their corresponding nodes.

Take a 6-axis robot for example, due to there are six joints and one plate for tool setting, the robot component has seven nodes under the root node (as shown in Fig. 5.3). The root node is the base of the robot, and every sub-node has its own physical joint and kinematic parameters. In the Fig. 5.3, the front six sub-nodes are corresponding to the six axes of Motoman HP3JC, and the last sub-node "mountplate" is the one to connect a tool into the robot system.

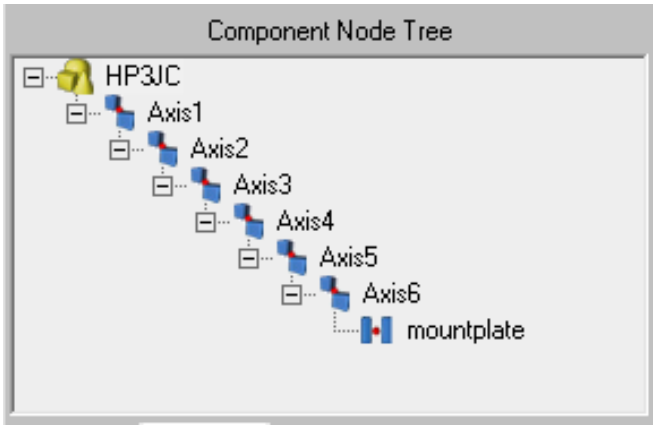


Figure 5.3: Node tree of a 6-axis robot

Besides of the physical features, frames in coordinate system are necessary to create for the definition of component location and movement positions. That is important for assembling layouts and creating trajectories.

5.3.3 Add Behaviors

Behaviors are the definitions of kinematics, characters, tasks and object links for components. Without behaviors, a component is just a stationary and isolated model. So it is necessary to add many kinds of behaviors for components, especially for servos and robots, which have moving parts in it.

5.3.3.1 Robot Behaviors

To make a 3D robot model working, the behaviors such as moving joints, kinematics, executor, etc, should be created for the robot component.

Take the robot Motoman HP3JC in the Motoman system workstation for example, as Fig. 5.4 shows, to make it work as a real robot, it requires behaviors including kinematics, controller, signals, executor, interfaces, Python script, etc.

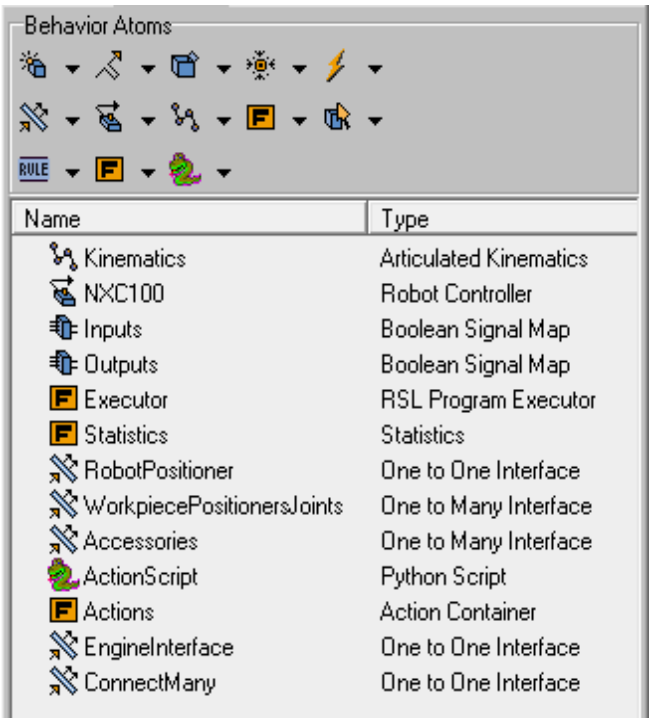


Figure 5.4: Behaviors of motoman HP3JC

(a) Robot Controller

Robot Controller is a behavior to control motions of independent joints using forward and inverse kinematics. The joints are defined in the controller with their type and properties, such as limit values, maximum speed and acceleration. They are corresponding to the kinematic properties of the real robot joints. For the Motoman HP3JC, it has six rotate joints for its six axes movements.

In the robot controller, basements and tool frames, which are similar to ones in real robots, are defined to make it easy to program robot movements -- node movements in robots and other inverse kinematics supported components. The initial basement and tool before moving are defined in the controller for the robot initial condition.

Similarly, the controller defines the kinematics behavior being used when calculating the inverse kinematics. Once inverse kinematics is defined, it will provide its related properties to the robot controller.

In addition, the robot controller attaches the joints to corresponding nodes for the definitions of their kinematic characteristics (moving type, limit values, .etc).

(b) Articulated kinematics

The articulated kinematics provides additional properties related to kinematics, including joint length, angle, positioning, coupling, configuration and tolerance levels dealing with robotic movement.

The most common robot kinematics in LSA is shown as Fig. 5.5. This solver can calculate the forward and inverse kinematics of a robot that has 6 rotational joints in the following order: RotZ, RotY, RotY, RotX, RotY, RotX [129]–[131]. In the Fig. 5.5, the parameters of LinkLength1 - LinkLength5 are the lengths of the joints, and the ones of JointOffset1- JointOffset3 are the distances among axes.

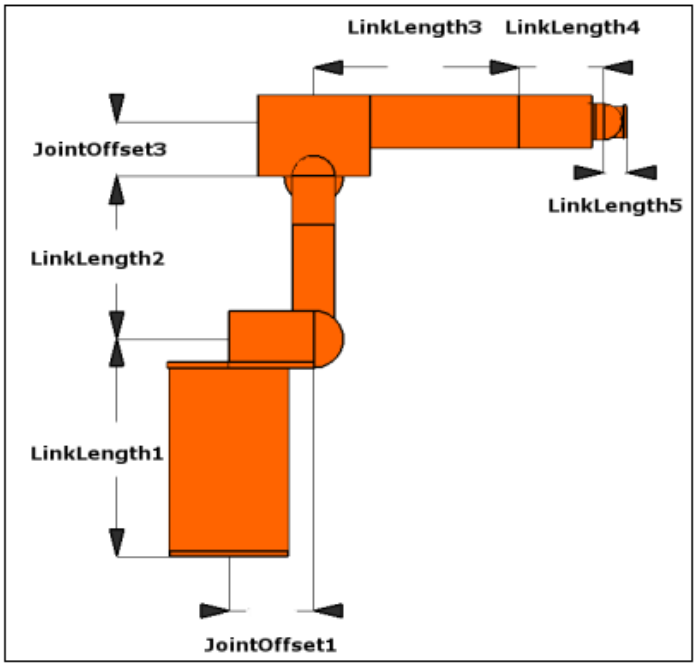


Figure 5.5: Articulated kinematics for a robot with 6 rotational joints [129]

Motoman HP3JC is a typical 6-axis robot in Motoman system workstation at celisca[132], [133]. Its dimensional parameters for kinematics are shown as Fig. 5.6. Once the robot nodes are attached with these kinematics parameters and the right arm configurations, all nodes could work together coordinately as the real robot.

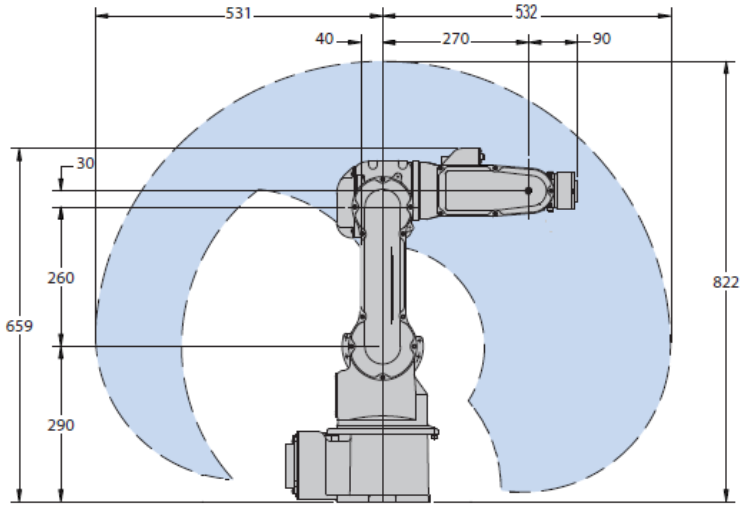


Figure 5.6: Articulated Kinematics of Motoman (Unit:mm) [134]

(c) RSL program executor

The RSL program executor executes RSL language sequences for robots and manipulators. RSL is a simple language for programming logics for robots and other Components [135]. RSL language consists of 3 levels: program, sequence and a statement. As long as the executor is created, the robot could be taught with motions, which is created in statements and sequence.

The RSL program executor attaches other behaviors such as robot controller, signals, and handlers to it. So it could call the properties of these behaviors when it teaches the robot moving. In addition, the action mode could be setting in the executor for the component to work together with other components. This is useful for peripheral components such as grippers and fixtures. If the component is connected to other components through interface that connects the RSL publisher field, the sequences of the RSL program can be launched from other components with help of “Remote Routine Call”.

(d) Interface

Interfaces are special behaviors used to make components to work together without exposing the internal details to other components, promoting component reuse [129].

It could connect different components together either in shape or in communication.

There are two kinds of interfaces for connecting components: one to one interface, and one to many interface [125], [136]. The first one is typically used for material flow, component attachments, signal communication and RSL execution. For example, when there is a tool required to set toward a robot, both of the robot and tool need such an interface for their connection. The second one allows connecting with multiple other interfaces using abstract connection. It is typically used with "remote" RSL execution. That is the critical technology for components working synchronously.

(e) Python script

The behavior atom Action Script is the one that uses a python script editor to perform different actions with robots and their tools, such as grasp/release, trace, mount and unmount actions.

Python script customizes component behavior by controlling other objects. The Python script execution may use signals and other events to control script execution [137]–[139].

In the behavior Python script, every kind of properties and motions of a component could be created and set. Parameters, behaviors, properties of a component, as well as its trajectories could all be created into the Python Script by Python API [125], [131], [140]. The script is programmed by Python language. In the virtualization of the VS, the Python script behavior is created for defining the physical characteristics for components, as well as properties of actions, and so on.

(f) Jog information

The JogInfo behaviors are created for independent moving nodes. It defines the degree of freedom (DOF) of the node, and attaches it with a joint for getting its type and properties.

5.3.3.2 Servo Behaviors

The servo behaviors are created for movements of mechanisms. To make a servo or its nodes moving, generally, the behaviors Servo Controller, Python script, and RSL program executor are necessary.

(a) Servo Controller

Servo Controller controls the motion of independent joints. It supports forward kinematics only, and cannot be used in robots. It is typically used in external axis systems, grippers, fixtures, weld guns and other simple mechanical structures.

The Servo Controller is "a container for joints". Each joint has a type, either rotational or translational, and properties such as limits, maximum speed and acceleration. The Servo Controller can calculate the execution time of a motion based on the slowest joint and synchronize the other joints so that it takes them the same amount of time to execute the motion. It also defines the root node and flange node for the servo.

(b) Python script

The motions, parameters and properties of a servo could all be written into Python script. They are defined in the script python code as text, which can be edited in the separate python editor.

(c) RSL program executor

To teach a servo motion as well as connect it to other components, the RSL program executor is required. When the action mode is set to true in the executor, the sequences with different motions could be remotely called by other component, which has connected with the servo executor.

(d) Interface

To connect a servo executor to other components, the interface behavior (“one to one interface” or “one to many interface”) is required for their connection. That makes multi-movements possible since the components could be connected by the interface, and the routines of the servo could be called remotely by other connected components.

Taking the workstation “Motoman system” for example, except for the regrip and the Motoman robot, as well as the BiomekNX and the BiomekFX, all the other components could move as servo. So for each of those components, servo controller is created for its nodes moving. In addition, for other components to remotely call its motions, the RSL program executor and interface are added into its behaviors. The action mode of the executor and the IsAbstract property of the interface are the critical factors in remote-routine calling.

5.3.4 Make Component Parametric

For a component, there are many factors parametric in its creation.

(1) Geometry parametric

The geometries of a component are made parametric by using parameters to control the properties of the features [129]. There are two approaches to the geometries parametric: (a) control the properties of the primitive features directly with a parameter; (b) do the desired parametric manipulation with the Transform feature.

(2) Behavior parametric

Some of the behaviors of a component are parametric for its conditions and properties setting.

(3) Parameters

Besides of the geometries and behaviors parametric, there are many parameters created for component parametric. As Fig. 5.7 shows, there are six parameters created for setting properties of a robot component. Among the parameters in Fig. 5.7, the parameter "Configure" in SignalActions is for setting output signals [147]–[148]. Firstly, all robots have a built in functionality to grasp and release components. By default grasp is done by setting any of the outputs 1- 16 that matches the tool number to true and release is done by setting the same output to false. For every output signal, when the tool is chosen, the parameters about detection volume size in axis X, Y and Z could be set for grasp accuracy. Secondly, a tool frame can draw a trace, which is turned on and off with a signal. The output signals 17 - 32 are by default mapped to tools 1 - 16, with true turning the trace on and false turning it off. Thirdly, the robot can mount or dismount a tool by setting a signal. The output signals 33 - 48 are by default mapped to tools 1 - 16, with true mounting the tool and false dismounting the tool [140]–[143].

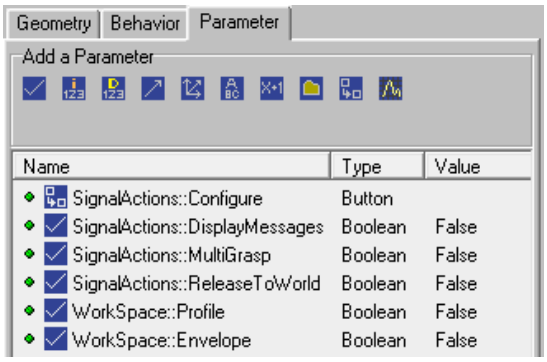


Figure 5.7: Paramters created for the robot Motoman HP3JC

5.4 Teach Components

The module “Teach” provides the functions to teach and program components that have servo or robot controllers. It teaches robots and servos motions and actions for their trajectories in the movement.

In 3DCreate, to teach a component with controller, statement and sequence are referred in creating motions:

1) Statements appear in the RSL programs in the RSL executors, and control the function of robot controllers. There are many kinds of statements to depict the motions and actions of robots and servos [144]–[147], such as “Linear (LIN)” or “Point to Point (PTP)” motions, grasp or release actions, executor delay, etc. Each statement has a list of properties, which can also be available through the COM and Python APIs.
2) A sequence is a series of statements in order. It is executed by logic executors. The main sequence holds the default storage area for initial positional data. Sub-sequence represents an action or a trajectory, which depicts a process that the component moves from one position to another target position [148]–[149]. All sub-sequences could be called by each other or the main sequence. The call sequence statement is used to execute other sequences in the same RSL program. The sequence specified in the statement is executed synchronously. It is completely executed before the next statement is executed.

In this dissertation, to save time for online virtualization, sub-sequences of all possible trajectories are created in advance for every component with controller. When some trajectories are referred to the experiment workflow data, corresponding sequences will be called and added into movements of a specified robot by programming 3DCreate API, which includes both COM API and Python API.

In the following, the Motoman system is taken as an example to explain the teaching process in the dissertation. In the workstation, except for the regrip, all the other components have their own movements. So motions and actions should be taught to every component to realize their movements.

5.4.1 Teach Servo

Generally, motions of a servo component (component with servo controller) are depicted as moving from one frame to another frame, and actions of that are depicted as open, close, issue, or incubate. For example, there are three possible actions for the component "PHERAstar": open, close, and issue command. Correspondingly, three sequences should be created for them.

As Fig. 5.8 shows, there are three statements for the "open" sequence. The first motion P1 is keep the all of the PHERAstar joints as original conditions. Then, the second motion P2 is created to change the value of joint2 from 0° to -180°, which makes the door of PHERAstar from close to open. After that, the motion P4 changes the value of joint3 in P3 from 0mm to 104mm. That makes the door keep open, and the position for labware putting being pulled out. It is fully the open condition of PHERAstar to prepare getting a labware. As for the "close" sequence, its statements are just in opposite order of the "open" sequence. For the issue command, all of its actions and motions take place in the inner of PHERAstar. So it is unnecessary to simulate these invisible works. The "issue command" is just consisted of the "Delay" statement with the same issue time.

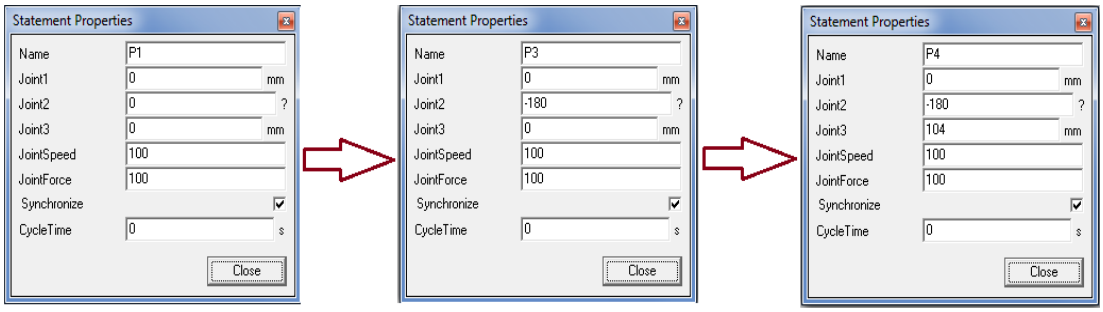
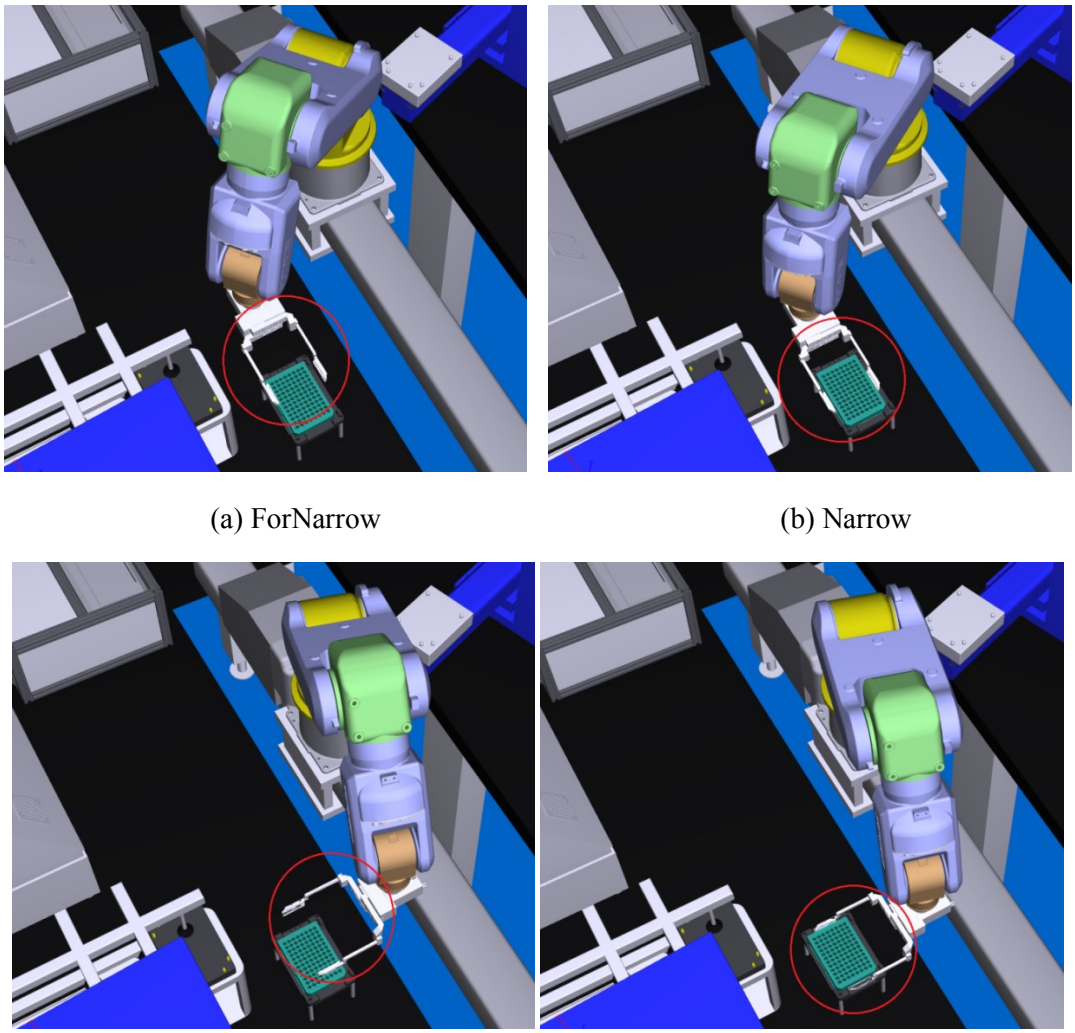


Figure 5.8: The statements of sequence "Open" in teaching PHERAstar

For the servo SG0150, which is the gripper tool of Motoman HP3JC, it is some different with other translate-moving servos. As normal gripper, it grasps or releases things. However, what's more for SG0150, it works with different sizes for different target positions. That is depicted as "using Narrow" or "using WideLow" in the workflow data. These two conditions work correspondingly to different planes of a labware.

As shown in Fig. 5.9, due to the gripper needs to change its condition and position time by time, there are four statements created for its corresponding conditions: Narrow, WideLow, ForWide, and ForNarrow. When the robot needs its gripper to grasp a labware using Narrow, the gripper should operate the "ForNarrow" sequence at first to make itself wider than the Narrow condition. That is the preparation for grasping in Narrow. Conversely, when the gripper puts a labware down on some position in the Narrow condition, the sequence "ForNarrow" should also be operated for releasing the labware. So the four sequences are corresponding to different sizes and angles of the gripper.



(a) ForNarrow (b) Narrow

(c) ForWide (d) WideLow

Figure 5.9: Sequences of the gripper SG0150

5.4.2 Teach Robots

A robot could have many different actions, such as move, grasp, release, etc. All the actions are taught firstly to robots before they could work. Generally, the teaching process for a robot is complex. It will take lots of time for developers to teach the robots, especially when there are many robots working in one case. Therefore, the dissertation presents a flexible method for robot teaching: once a robot component with many taught sequences is created, other robots with similar structures could copy the characters of the defined robot, and then make some changes for characters of the new robot. The method saves a lot of time and is very convienient for system developers when some new virtualization has to be created for other workstations.

The Motoman system is a typical modern LSA workstation. The following sub-sections will intepret the teaching process for the three robots (Motoman HP3JC, BiomekFX and BiomekNX-Span8) in Motorman system in detailed. Based on the teaching information, developers could create a new robot component with similar structures easily.

5.4.2.1 Preparation

Before teaching robots to work, some factors should be created and set:

(1) Frames and robot positions: All robot positions are represented by robot position frames in 3Dworld. That is the same to target positions for tool grasping. So many frames are created in every component for robot moving and working. In creating a new motion for the robot, it just needs to snap corresponding frames in the trajectory, and makes the end frame as the tool target.
(2) Base: It is a coordinate system that the robot positions (motion statements) are relative to. When the base moves, all the motion statements referencing the base also move. It's especially useful in repetitive transportations.
(3) Tool: It is a Tool Center Point (TCP), typically relative to the robot flange plate. There is usually no need to change the tool value programmatically. However, the interpolation mode (IPO mode) for either a base or tool can be set to inverse motion targeting and use an external coordinate system, i.e. from External TCP to Base target solution.
(4) Output signals: It defines actions for a robot, such as grasp/release, trace, mount and unmount a tool. Different signals have different variables and parameters. They are corresponding to different actions for robots. It could be defined or set

in either the Action Script editor (python) or an Action Map editor that is automatically created by the Action Script behavior.

(5) External TCP: An External TCP is a tool that is not attached to the robot, but another object (typically a static object, but can be something moving as well). Setting the IPO mode to TCP in either a Base or Tool sets ExternalTCP to True. In contrast, the IPO mode set to Base sets ExternalTCP to False. By default, the IPO mode is set to zero for all bases and tools which allows a user to turn the ExternalTCP on or off from via the Teach tab.

(6) Configuration: It is an alternative way to reach the same goal position. Configurations are used only if the motion interpolation type is set to Joint (point to point). In Linear motion interpolation the closest configuration is automatically selected. The number of configurations depends on the robot type.

With the above factors, the teach module in 3DCreate could call them to teach a robot detail information about target positions, configuration mode, etc for motions generation. There are three robots defined in 3DCreate for the Motoman system workstation: BiomekFX, BiomekNX-Span, and Motoman HP3JC. The teaching processes for their motions are shown as followings.

5.4.2.2 Teach Biomek FX

Biomek FX is a laboratory automation workstation, which is composed by deck, towers, bridges, Automated Labware Positioners (ALPs), multichannel pods and heads (or Span-8 Pod and its liquid system), as well as a pair of grippers [150]–[153]. Its structures and characteristics are shown as Appendix A.

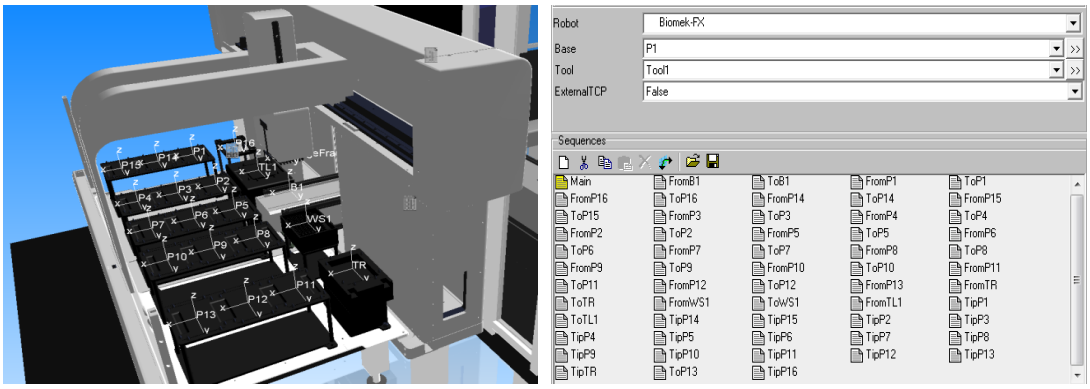
In the Motoman system workstation, the Biomek FX has the structure with multichannel pod. It has six joints for its movements and works. Among the moving and working components, the bridges could hold the pod, heads and grippers together to move along its rail for defining positions (shown as Fig. A.3). It has a joint with an X-axis linear DOF. The multichannel pod, which holds the heads and grippers, has two joints for Y-axis and Z-axis linear movements separately. As for the grippers, to grasp labware flexibly and avoid affecting the head's works, there are three joints to ensure its motions in X-axis and Z-axis directoins. The limit values for these joints are based on the movement regions of the real components.

There are 21 frames on corresponding ALPs (see as Appendix A.4) of Biomek FX for locating labwares onto target positions in movements. The location could be for either

grasping or pipetting, which is executed by grippers or tips in mandrels. So there are two tool basements created for Biomek FX: one tool base for grippers, the other one for tips.

The following works could be done by Biomek FX workstation in a LSA assay: location for labware, load and unload tips, aspirate/dispense liquid, wash tips, drain and refill a reservoir. In the location process, the joints of bridges and pods are assigned to the specified value of the target position. Then based on the task, the tool basement of grippers or mandrels is moved to the target frame. At last, the grippers change their distance through setting their joints' values to prepare for grasping or releasing, or the tips set in mandrels pipette liquid. If the workflow refers gripper actions, the output signal of the grippers' tool should be set correspondingly for grasping labware from the target ALP or release it to the ALP. In the loading and unloading tips process, at first, the tool basement of mandrels should be located to the TL1 frame, and then the output signal of the mandrels' tool is set for loading or unloading tips in the TL1 ALP. When the pipetting work is referred in the workflow data, tips should aspirate or dispense liquid in some specified ALP also through setting their output signals. In all, every work of Biomek FX refers to location tool basement to target frame, and then call corresponding output signal of related component to realize the task in the workflow data.

Figure 5.10 shows frames created in Biomek FX ALPs, and parts of the sequences taught to it. As Fig. 5.10 (b) shows, in the VM of VS, all possible locations and actions for different ALPs are defined to their corresponding sequences. Every sequence has its own statements in order, which include those motions and actions for its task. For instance, the sequence "FromP1" means to get a labware from the P1 ALP. It includes statements of location the tool basement of grippers to the frame P1, and set the joints of grippers to the labware width, and then set its output signal as "pick up". As for the sequence "ToP1", except for the same location statement, other statements are all opposite with the "FromP1" sequence. For the "Tip-" sequences, they have actions insist of locating mandrels tool basement, loading tips, aspirating liquid from a specified ALP, and dispensing liquid to target ALPs, etc.



(a) Frames on Biomek FX (b) Seuqnces for Biomek FX actions

Figure 5.10: Frames and sequences of Biomek FX

5.4.2.3 Teach Biomek NX Span-8

The Biomek NX is a multiaxis instrument designed with an open architecture to allow expandability of the system [154]–[155]. As shown in Fig. B.1 in Appendix B, the main-function components of the Biomek NX Span-8 in the Motoman system at celisca are deck, towers, and bridges, Span-8 Pod with probes, grippers, ALPs.

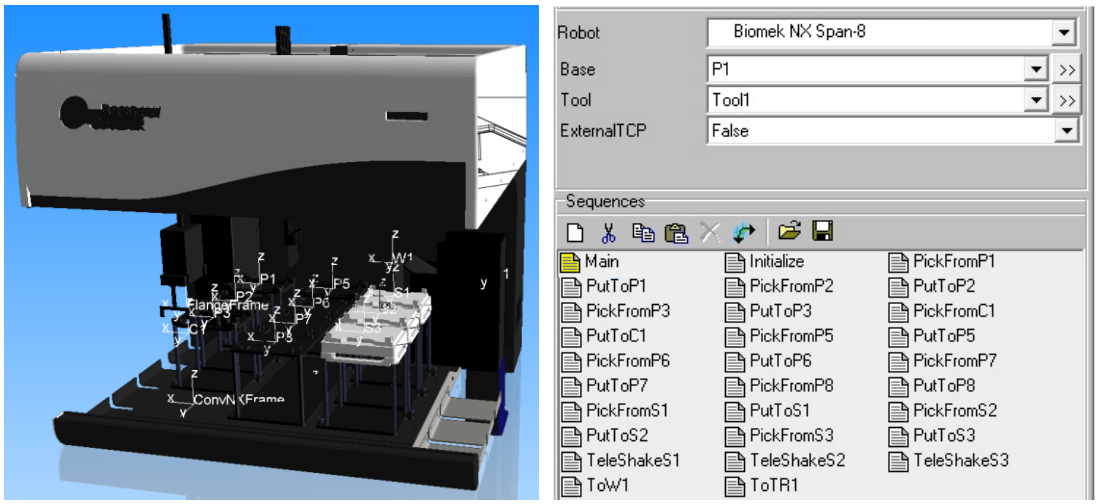
There are two groups of bridges installed in the Biomek NX Span-8 workstation [156] –[157]. One has a joint with X-axis DOF for holding the gripper to move along the X-axis rail of the workstation. The other one also has a joint for X-axis movement of the Span-8 Pod and its components. To make the gripper work, there are four joints created for its Z-axis movement (up and down), Y-axis movement (left and right, one joint for each finger), and C-axis movement (rotate with Z-axis). As for the Span-8 Pod, it has one joint for its eight probes moving along Y-axis simultaneously, 8 joints

to move every probe in the Z-axis independently, and 8 joints for probes to pipette in the D-axis with the assistance of the pumps independently, as well as 8 joints to make the span between probes expand and collapse.

There are three devices named Teleshake installed in the right ALPs of Biomek NX Span-8 workstation. In the Motoman system workstation at Celsica, the Teleshakes hold microplate and spin with it around the center axis of the corresponding ALP. So there is one rotate joint for every Teleshake created when the component is generated in 3DCreate.

To define target positions for the tool basements of grippers and pod in virtualization, there are 11 frames created on ALPs (see as Appendix B.4) of Biomek NX Span-8 for snap and definition. When a frame definition is required for the gripper or the pod, corresponding joints of bridges and their own components take movements along their moving axes and make the tool basement coincided with the frame. Then the motions of grasp, release, or pipette could be executed by corresponding components.

Figure 5.11(a) shows frames created in Biomek NX Span-8 ALPs. Except for them, a ConvNXFrame is created for plug and play connection with the component ConvNX; FlangeFrame is created to grasp for the gripper; eight frames are created for eight probes separately, so that the probes could aspirate or dispense liquid with specified quantity. Figure 5.11(b) shows parts of the sequences taught to the Biomek NX Span-8. The sequence "Initialize" works for setting all components to their initial conditions. The sequence named as "PickFromP1" include motions of definite the gripper tool basement to P1 frame, get the gripper fingers down and wider to prepare grasp labware and adjust the gripper fingers as grasp condition, as well as an action of pick up the labware, which is realized by an output signal. As for the sequence as "PutToP1", it has opposite order of motions and an action for putting the labware down to the P1 ALP. For the components Teleshake, besides of the sequences for gripper motions to pick up or put on their ALPs, there is another kind of sequence which works for shaking the labware on them. For example, the sequence "TeleshakeS1" includes motions to shake the components by setting the values of the Teleshake joints.



(a) Frames on Biomek NX Span-8 (b) Seuqnces of Biomek NX Span-8

Figure 5.11: Frames and sequences of Biomek NX Span-8

5.4.2.4 Teaching Robot Motoman

The robot Motoman HP3JC is the main transport robot in the Motoman system. It moves along its track and shifts the assay labware from one device position to another one. In the virtualization, the Motoman not only does its own transportation tasks, but also links the movements of all components together and forms a sequence for them

to generate a full virtualization for the experiment workflow.

Among the actions of the Motoman in the virtualization, the works about its own transportation and calling other components' movements are set as the sequences in the simulation layout beforehand, which is shown in Fig. 5.12. They will be called when there are related processes in the experiment workflow data. As for the sequence of all movements in the virtualization, it is generated in the CS with the sequences being called.

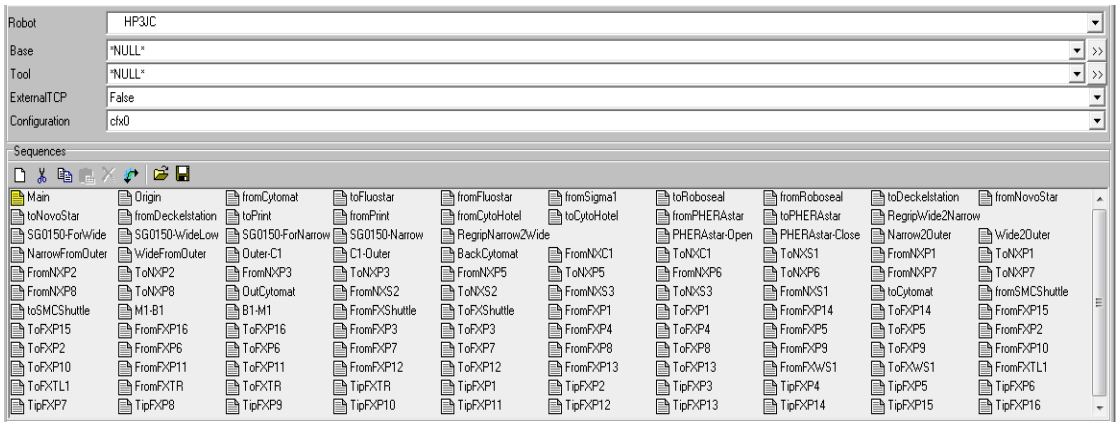


Figure 5.12: Sequences of Motoman HP3JC

(1) Origin and tool location

The sequence "Origin" has motions to drive the Motoman back to its specified original position. It is created mainly for backing up the original positions information of its all nodes. When there is some collision or error as run out of limit values, it could restore the robot to its normal state.

The robot itself has a tool frame in its tool interface. It is a Tool Center Point (TCP), typically relative to the robot flange plate. When the gripper SG0150 is installed into the robot in the tool interface, a new tool frame is created for grasping labware using the gripper.

As Fig. 5.13 shows, the tool frame is set in the middle of the two fingers of the gripper so that it could easily snap other target frames for grasping or releasing. That is done in the CS at the beginning of the data-assign stage. After that, in the motion creation process, all robot target positions could be set for the tool frame of the gripper SG0150.

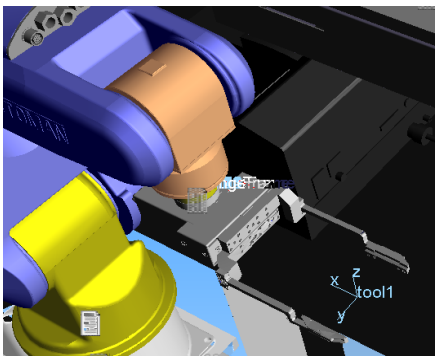


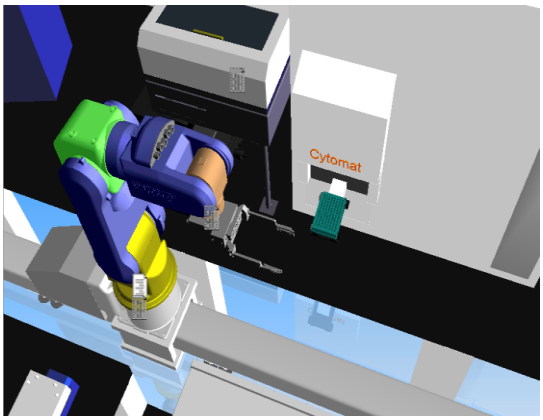
Figure 5.13: Tool frame of the robot gripper

(2) Pick labware

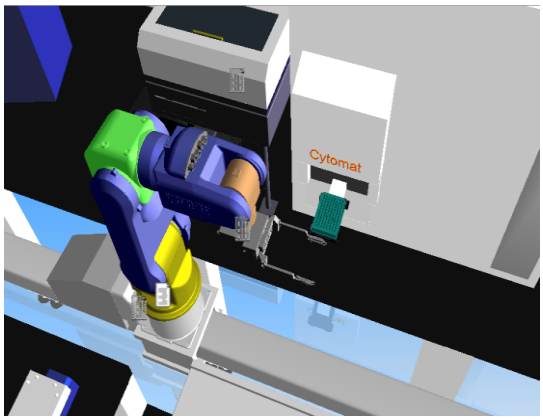
There are many possible positions the robot Motoman could reach in the workstation. When the workflow data refers actions of the Motoman to get labware from some position, a sequence named “from” with device name (such as “fromCytomat”) is called by the CS.

For example, the motions in the sequence “fromCytomat” are all based on the frame tool1 (as Fig. 5.13). There are three steps in the statements of the sequence: 1) move the robot to the target position near the Cytomat6001; 2) adjust the gripper to the right working condition; 3) pick up the labware from the Cytomat6001 ALP. To drive the robot to the target position, many point to point (PTP) statements are created for joints’ movement of the robot. In every PTP creation, the tool, base, ExternalTCP and configuration are defined and chosen, and the target position values in the six DOF are also defined. Beside of them, the value of ExternalJoint is important for the position definition of the robot base.

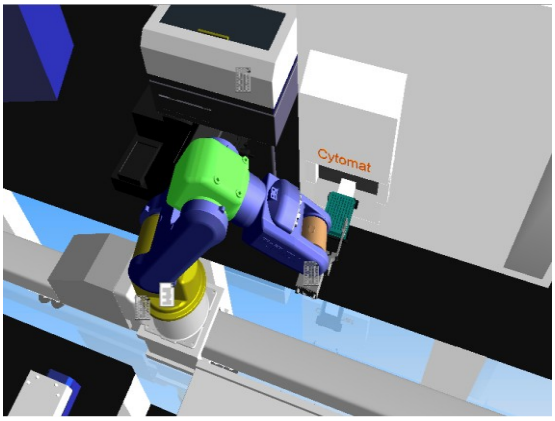
The External Joint is set on the position near Cytomat (Fig. 5.14 (b)), which makes the movements of the seven joints and the action “pick up” possible. After the robot arrives at the position where it could touch the Cytomat frame, it changes its joints values and adjusts its tool (the gripper) condition to the right direction and gesture (Fig. 5.14 (c)), which could press the labware from both sizes in its middle height. These actions need to remotely call corresponding sequences of SG0150 to adjust it to its narrow grip condition. After that, an output signal is called and set as the action “pick” (Fig. 5.14 (d)). At last, a PTP motion is created to raise the gripper from the Cytomat frame (Fig. 5.14 (e)), and another one is generated to make all nodes of the robot back to their conditions as the one when it arrived (Fig. 5.14 (f)).



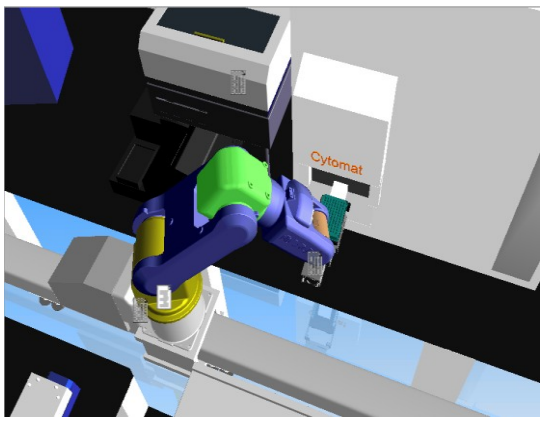
(a) Origin



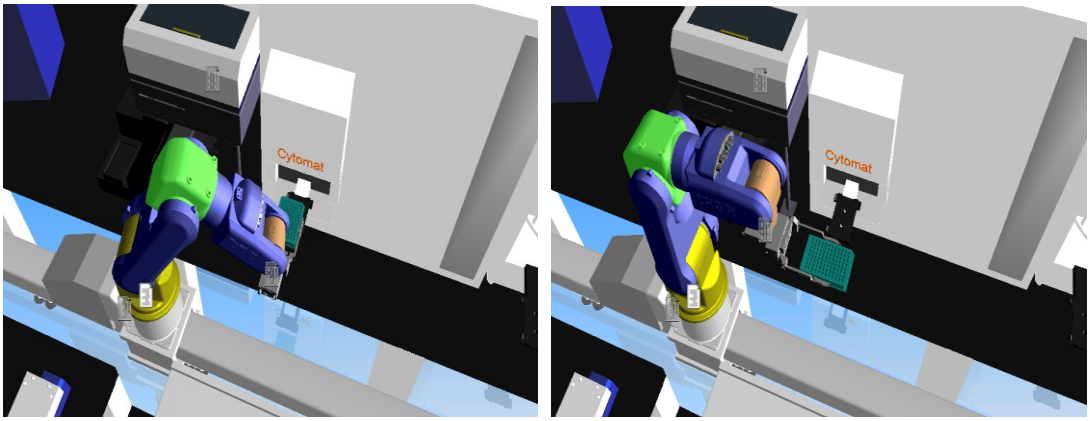
(b) Near Cytomat



(c) Adjust tool



(d) Pick/grasp



(e) Get up (f) Condition back

Figure 5.14: Get a labware from Cytomat

(3) Put down

The sequence “To” with device name (such as “toCytomat”) has statements in the opposite order of the “From” one. Take the sequence “toCytomat” for example, it also has three steps in its statements: 1) move the robot with a labware to the target position near Cytomat6001; 2) adjust the gripper to the right working condition; 3) put the labware down to the Cytomat6001 ALP. Among them, the statement for the action “put down” calls an output signal of release. After that, sequences of SG0150 are remotely called to make the distance between its two fingers wider so that it could get up freely.

(4) Call Remote routine

As the above depicted, when the robot grasp or release a labware, some sequences of the gripper SG0150 should be run to get a right gesture for the action. That is realized by calling remote routines of SG0150 in the Motoman statements. It is also the same to generate a full sequence for the experiment workflow.

To call the remote routine of other components, it is necessary to create many connections. In the dissertation, every component has to connect with more than one component so that it could call or be called with many components. So for every component, a behavior “One to Many interface” is created. The behavior allows connecting with multiple other interfaces using abstract connection. It uses one template interface section to describe the elements that are connected. Template Interface Section has user definable Interface Fields, which are used to define what is interfaced when components are connected. The “One to many interface” behavior is used with the "remote" RSL execution, because it is abstract in nature (not bound

physically as material flow or hierarchy attachment).

The process of calling the remote routine has two steps: one is to start a remote routine, and the other one is to wait the remote routine finish. The Start Remote Routine statement signals another RSL executor to start the execution of a specified sequence. The remote RSL executor has to be associated with this executor through an RSL field, and its ActionMode property must be set to true. Remote routines are executed asynchronously, so execution continues immediately after this routine is done. The Wait for Remote Routine Statement is used to wait for the remote routine to complete.

5.5 Creating Layouts

When all components are created with their geometries, parameters and behaviors, they are assembled together to a workstation based on their relative positions in the realistic workstation, and formed as a layout in 3DCreate.

Figure 5.15 shows the workstation layout of the Motoman system. All the components dimensions and their distances are in actual size. There is a basement frame for every component to be installed to the workstation platform. In the layout of Motoman system, all possible motions and trajectories have been created and kept into the layout for being called by the CS.

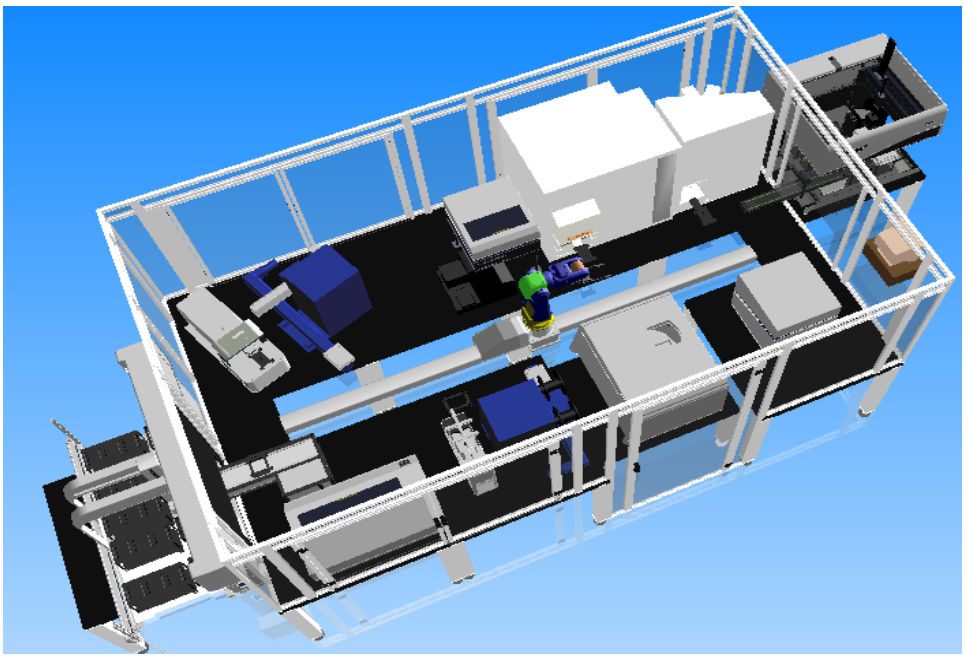


Figure 5.15: Workstation layout of Motoman system

5.6 Discussions

The Vitualization Module prepares kinematic components for the CS to call and drive. It supplies a platform to generate virtualization results for the 4D virtualization system.

Since there are different devices in different LSA workstation, to make a new workstation virtualization, developers should create new models, components, and layouts in the VM module in advance for the CS. The dissertation presents a convenient way to make the following creation of components easier. Developers just needs to copy existing component to the new model with same structures, and then make some small changes for the new model to get its component in the VM. The work in the dissertation to create components for devices and robots in Motoman System, takes much convience and saves much time for the components generation of the other workstations.

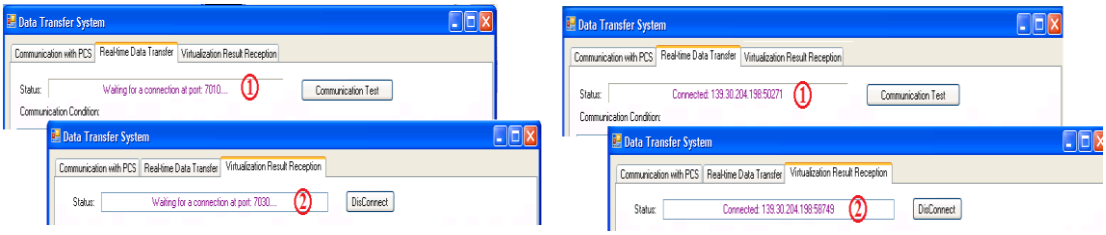
Chapter 6 System Test and Application

The modules of theVS are connected together all by interfaces, including the SILAS OCX between the PCS and the DTS, the TCP/IP socket between the DTS and the CS, the COM and Python API between the CS and the VM. When there is an interrupt in any interface, other modules still go on work based on existing data. To demonstrate and verify the VS for life science applications, this dissertation uses a whole application case for example. The case is executed as steps as followings. The test and application results confirm that the system could simulate the experiment workflow of LSA not only in real time but also in history well and smoothly.

6.1 Connections among Modules

Before running the PCS to get original experiment workflow data, the connections among modules in the VS should be created. Once the DTS is running, it activates the SILAS OCX at once and waits for the data generation. While the connections between the CS and the DTS are created by two TCP/IP sockets for the transmissions of workflow data and .pdf file separately, the DTS shows the IP address and the separate transmission port at the side of the client CS in its interfaces.

The connection statuses of the two TCP/IP socket are shown as Figure 6.1. The Fig.6.1 (a) shows the port-bind statuses of the two different modules in the DTS; Fig.6.1 (b) demonstrates the socket connection statuses of them.



(a) Waiting for connections (b) connected

Figure 6.1: Connection statuses of the two sockets

6.2 Method in the Process Control System

In this case, at first, a method for one-plate assay is designed in SAMI EX Editor of the PCS. As shown in Figure 6.2, in SAMI EX Editor, the workflow for this experiment is demonstrated in the form of icons. The method of this case contains parameters and activities of components in the LSA workstation -- Motoman system.

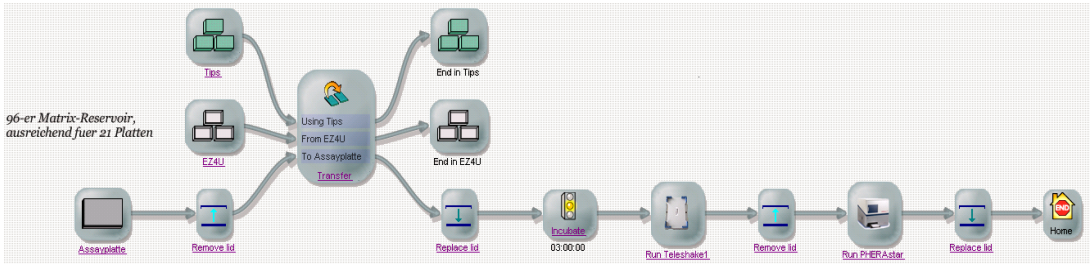


Figure 6.2: Method for one-plate assay

After scheduling and running the method, the SILAS system in the PCS generates the workflow data for the LSA experiment. Since the data are complicated and lenghy, Table 6.1 lists parts of important and representative processes in the data of this method. The processes will be demonstrated in the 4D virtualization results in the following context.

Table 6.1 Parts of the workflow data of the method “One Plate_FX”

Time	Activities	No.
.....	.....	
8:54:07 - 8:54:07	Place labware on Cytomat6001	(1)
.....	.....	
8:54:12 - 8:54:13	Place labware on BiomekFX	(2)
.....	.....	
8:54:31 - 8:54:45	Motoman move from Cytomat6001.transfer station to Regrip.BCR using LidNarrow	(3)
.....	.....	
8:54:45 - 8:54:57	Motoman move from Regrip.BCR to SMCShuttle.M1 using WideLow	(4)
.....	.....	
8:54:57 - 8:55:01	SMCShuttle move from SMCShuttle.M1 to BiomekFX.Shuttle using SelfGrip	(5)

.....	.....	
8:55:06 - 8:55:09	BiomekFX:LeftPod put Assayplatte_1 to BiomekFX.P15 using Attila_DefaultGrip	(6)
.....	.....	
8:55:17 - 8:55:31	Load tips for Transfer/9	(7)
8:55:31 - 8:55:36	Transfer for Transfer/9	(8)
8:55:36 - 8:55:43	Transfer for Transfer/9	(9)
8:55:43 - 8:55:50	Tip handling for Transfer/9	(10)
.....	.....	
8:56:03 - 8:56:07	BiomekFX:LeftPod put Assayplatte_1 to BiomekFX.Shuttle using Attila_DefaultGrip	(11)
.....	.....	
8:56:07 - 8:56:11	SMCShuttle move from BiomekFX.Shuttle to SMCShuttle.M1 using SelfGrip	(12)
.....	.....	
8:56:11 - 8:56:23	Motoman move from SMCShuttle.M1 to Regrip.BCR using WideLow	(13)
.....	.....	
8:56:23 - 8:56:33	Motoman move from Regrip.BCR to Cytomat6001.transfer station using LidNarrow	(14)
.....	.....	
8:57:18 - 11:57:18	Incubate Assayplatte_1 at Cytomat6001.2-1 for 3:00:00	(a)
.....	.....	
11:57:30 - 11:57:43	Motoman move from Cytomat6001.transfer station to ConvNX.outer using LidNarrow	(15)
.....	.....	
11:57:43 - 11:57:46	ConvNX move from ConvNX.outer to BiomekNX-Span.C1 using SelfGrip	(16)
.....	.....	
11:57:53 - 11:57:55	BiomekNX-Span:LeftPod put Assayplatte_1 to BiomekNX-Span.S1 using Chimera_DefaultGrip	(17)
.....	.....	
11:57:55 -	Issue command to Teleshake1 (1:00)	(b)

11:58:53		
.....	.....	
11:58:58 - 11:59:01	BiomekNX-Span:LeftPod put Assayplatte_1 to BiomekNX-Span.P3 using Chimera_DefaultGrip	(18)
.....	.....	
11:59:08 - 11:59:12	Open PHERAstar.R for put	(19)
.....	.....	
11:59:11 - 11:59:14	BiomekNX-Span:LeftPod put Assayplatte_1 to BiomekNX-Span.C1 using Chimera_DefaultGrip	(20)
.....	.....	
11:59:14 - 11:59:17	ConvNX move from BiomekNX-Span.C1 to ConvNX.outer using SelfGrip	(21)
.....	.....	
11:59:17 - 11:59:28	Motoman move from ConvNX.outer to PHERAstar.R using WideReverse	(22)
.....	.....	
11:59:28 - 11:59:32	Close PHERAstar.R from put	(23)
11:59:32 - 12:00:42	Issue command to PHERAstar (1:47)	(c)
12:00:42 - 12:00:47	Open PHERAstar.R for get	(24)
12:00:47 - 12:00:59	Motoman move from PHERAstar.R to ConvNX.outer using WideReverse	(25)
.....	.....	
12:00:59 - 12:01:02	ConvNX move from ConvNX.outer to BiomekNX-Span.C1 using SelfGrip	(26)
12:00:59 - 12:01:03	Close PHERAstar.R from get	(27)
.....	.....	
12:01:06 - 12:01:08	BiomekNX-Span:LeftPod put Assayplatte_1 to BiomekNX-Span.P3 using Chimera_DefaultGrip	(28)
.....	.....	
12:01:19 - 12:01:21	BiomekNX-Span:LeftPod put Assayplatte_1 to BiomekNX-Span.C1 using Chimera_DefaultGrip	(29)
12:01:21 - 12:01:24	ConvNX move from BiomekNX-Span.C1 to ConvNX.outer using SelfGrip	(30)

.....	.....	
12:01:24 - 12:01:36	Motoman move from ConvNX.outer to Cytomat6001.transfer station using LidNarrow	(31)
.....	.....	
12:01:46 - 12:01:51	Cytomat6001 put Assayplatte_1 to Cytomat6001.1-1 using SelfGrip	(32)

6.3 Data Transmission in the Data Transfer System

Figure 6.3 shows the two parts of the data transmission in the DTS. The part ① demonstrates the workflow data from the PCS is received in real time successfully in the DTS, and it is extracted for some factors including start time, duration time, activity and status, but not all factors of the data. On the other hand, the part ② demonstrates the successful connection of the DTS by the CS socket, and the real-time data has been sent to the CS and received by the CS one by one, which could be certificated in the interface by the feedback information about the data reception conditions from the CS.

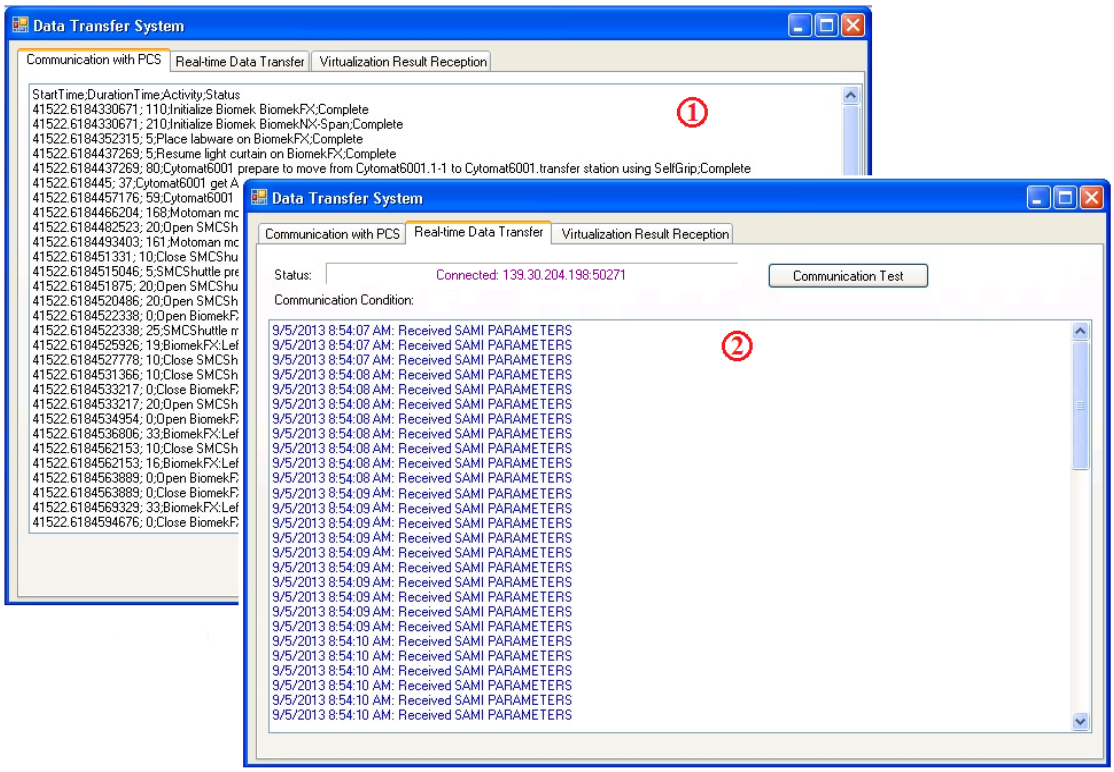


Figure 6.3: Data transmission in the DTS

6.4 Online Virtualization in the Control System

While the CS receives the real-time data one by one, it converts the data to 3D trajectories and actions in 3DCreate at soon. Figure 6.4 shows the movements the CS generated in 3DCreate. First of all, the tool base is located to define the working TCP of robot Motoman. Every movement is called from sequences of its corresponding component, and integrated to form a whole experiment workflow in 4D under the main sequence of the robot Motoman. Among the statements of the main sequence of the Motoman HP3JC, the three delay statements respond respectively the activities No. (a), No. (b) and No. (c), which refer to the action “incubate” or “issue” inside some components or inside the labware (microplate).

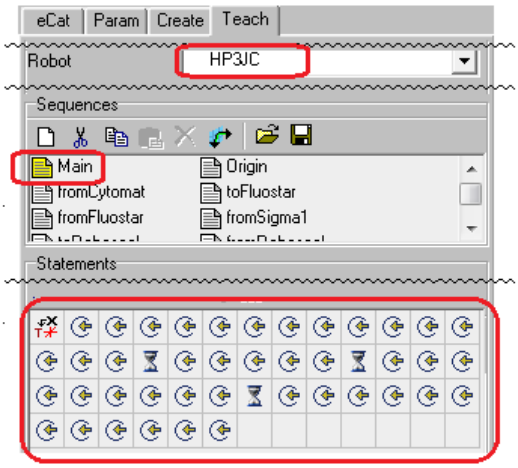
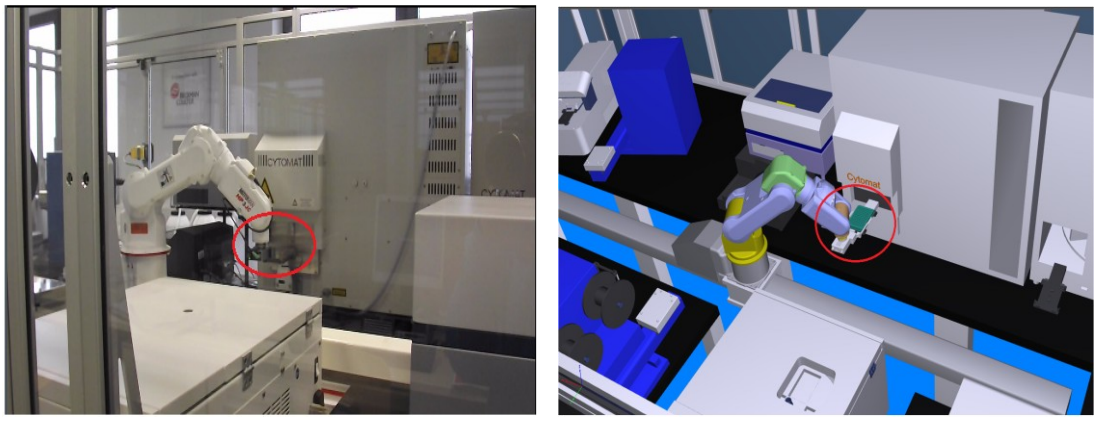


Figure 6.4: Signs of movements generated by the VS

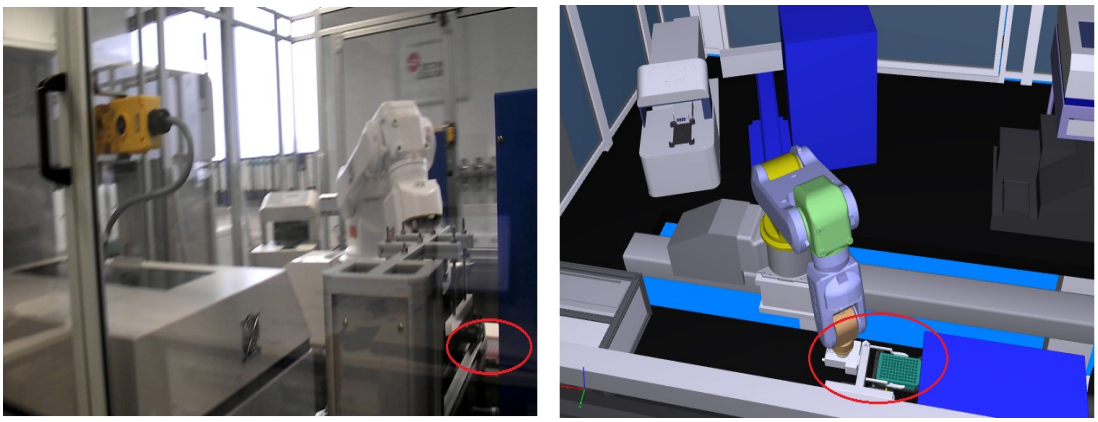
Figure 6.5(1) - Figure 6.5(16) compare the virtualization results with the movements on the realistic workstation at different moments. Every figure shows a condition of the workstation in a workflow step, which is corresponding to the data listed in table 6.1. In every figure, there are two pictures: the left one is a part of the realistic workstation photo at the running time, the right one is a screenshot from the virtualization result, which is shown as an animation. The figures show all actions and movements the virtualization system creates are the same as the realistic ones. They verify the movements in the virtualization are the same as the real workflow in the workstation, and the virtualization is an on-line simulation for the running workflow in the realistic workstation. From the figures, many components work together sometimes. Taking Figure 6.5(3) for example, while the robot adjusts its gesture for picking the plate up from the regrip position, its gripper SG0150 changes its condition from “Lidnarrow” to “WideLow”. Then the Motoman could move the plate from

Regrip.BCR position to SCShuttle.M1 position using WideLow gesture of the gripper. Figure 6.5 (7) shows the tips in the tip-box (ALP: TL1) is loaded to the BiomekFX head; Figure 6.5 (8) and Figure 6.5 (9) separately demonstrate the aspirate and dispense action for chemical liquid; and Figure 6.5 (10) is the condition that the head releases the tips back to the tip-box. Figure 6.5(14) shows while the plate is transferred on ConvNX, PHERAstar opens its ALP PHERAstar.R for putting the plate; Figure 6.5(16) shows after Motoman puts the plate on PHERAstar.R position, PHERAstar closes its ALP, and at the same time, Motoman raises its joints to avoid collision with the door of PHERAstar when it is closed. At the opposite, as the table and figures show, after Motoman gets the plate from PHERAstar.R to ConvNX.outer, PHERAstar.R closes its ALP from the "get" condition, and ConvNX moves the plate from ConvNX.outer to BiomekNX-Span.C1 synchronously. It could be clear to demonstrate from their time factor in table 6.1.



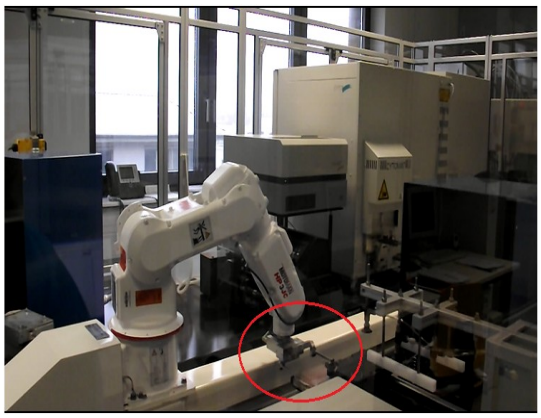
(a) Actions in realistic workstation (b) Virtualization result

(1) Motoman move from Cytomat6001.transfer station (Time: 8:54:31)

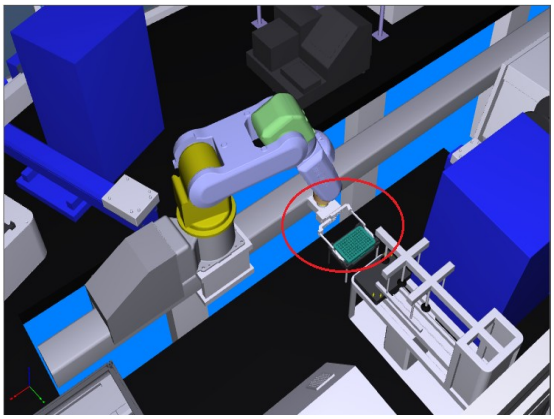


(a) Actions in realistic workstation (b) Virtualization result

(2) Motoman move to Regrip.BCR using Lidnarrow (Time: 8:54:45)



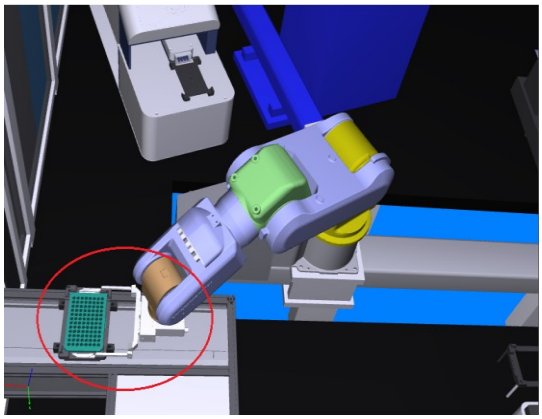
(a) Actions in realistic workstation



(b) Virtualization result

(3) Motoman move from Regrip.BCR using WideLow (Time: 8:54:45)

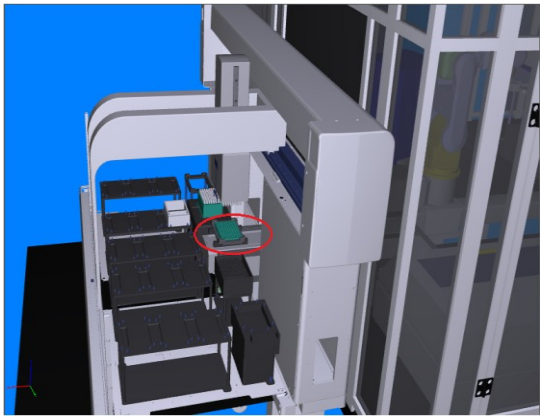
(a) Actions in realistic workstation



(b) Virtualization result

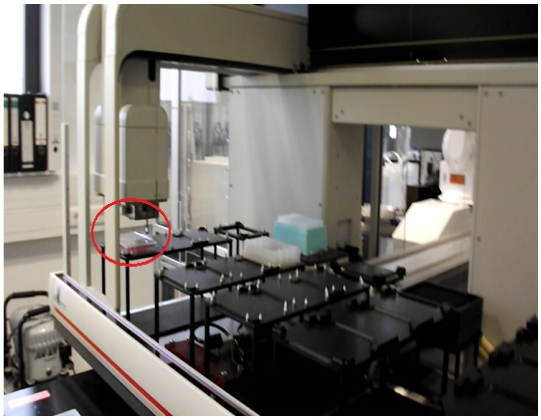
(4) Motoman move to SMCShuttle.M1 using WideLow (Time: 8:54:57)

(a) Actions in realistic workstation

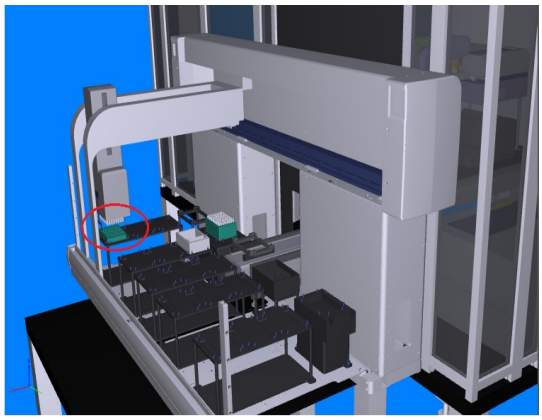


(b) Virtualization result

(5) SMCShuttle move to BiomekFX.Shuttle using SelfGrip (Time: 8:55:01)

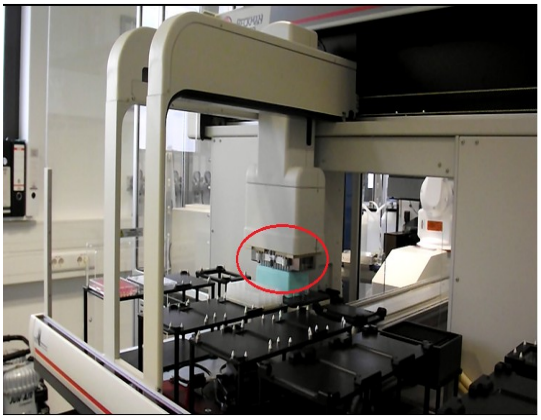


(a) Actions in realistic workstation

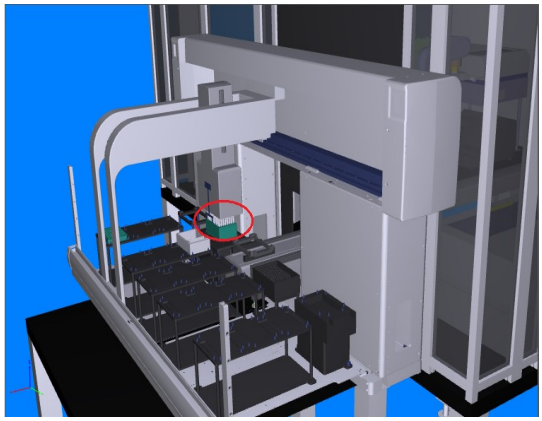


(b) Virtualization result

(6) BiomekFX:LeftPod put to BiomekFX.P15 (Time: 8:55:09)

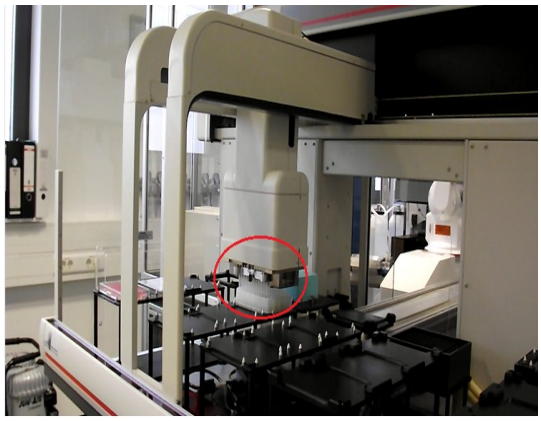


(a) Actions in realistic workstation

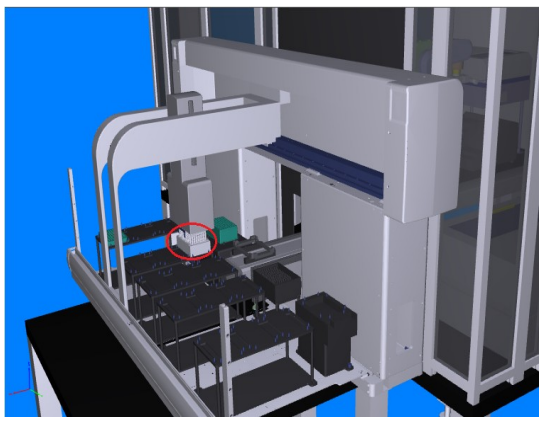


(b) Virtualization result

(7) Load tips for Transfer/9 (Time: 8:55:31)

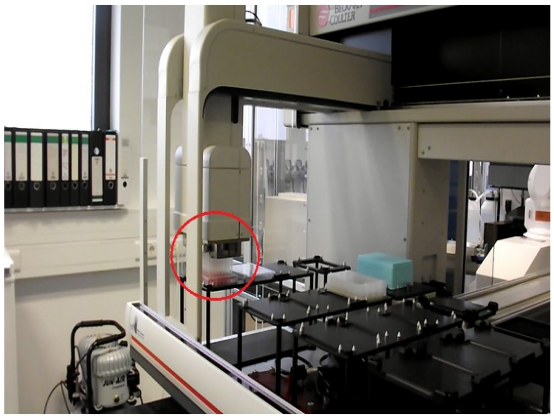


(a) Actions in realistic workstation

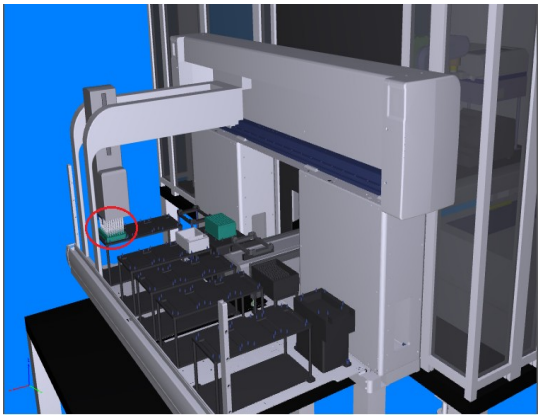


(b) Virtualization result

(8) Transfer for Transfer/9 (Time: 8:55:36)

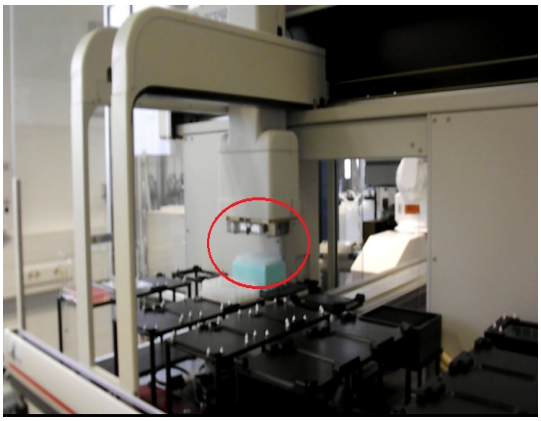


(a) Actions in realistic workstation

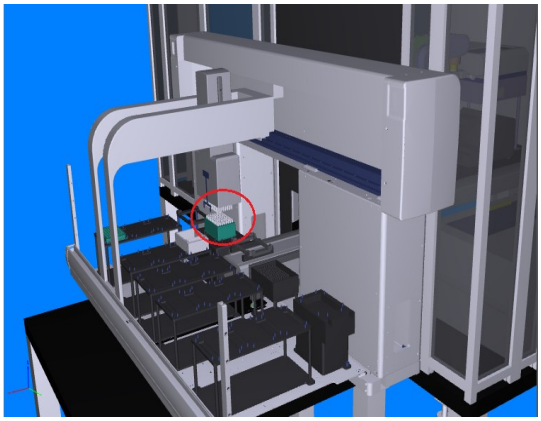


(b) Virtualization result

(9) Transfer for Transfer/9 (Time: 8:55:43)

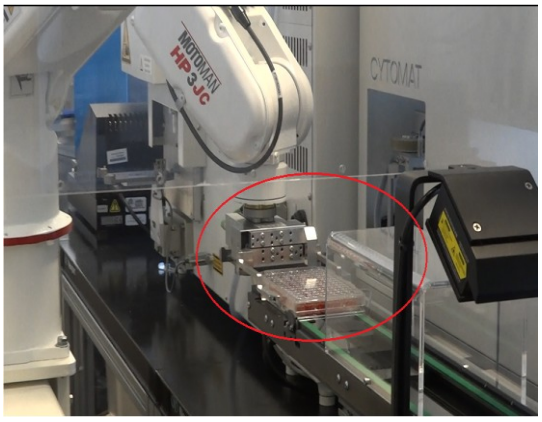


(a) Actions in realistic workstation

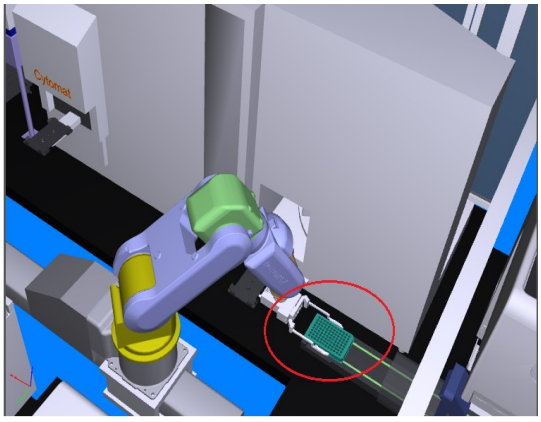


(b) Virtualization result

(10) Tip handling for Transfer/9 (Time: 8:55:50)



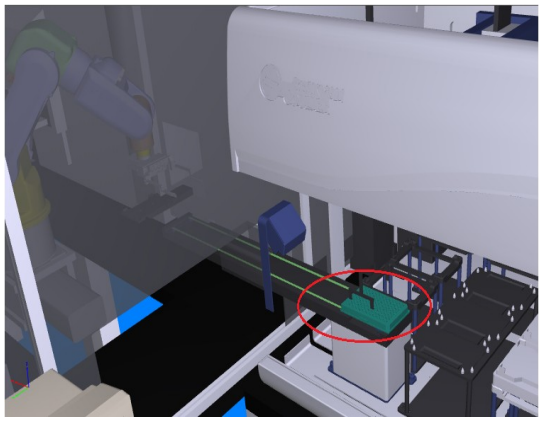
(a) Actions in realistic workstation



(b) Virtualization result

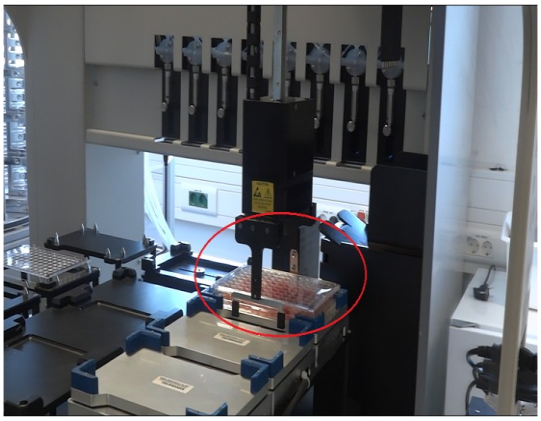
(11) Motoman move to ConvNX.outer using LidNarrow (Time: 11:57:43)

(a) Actions in realistic workstation

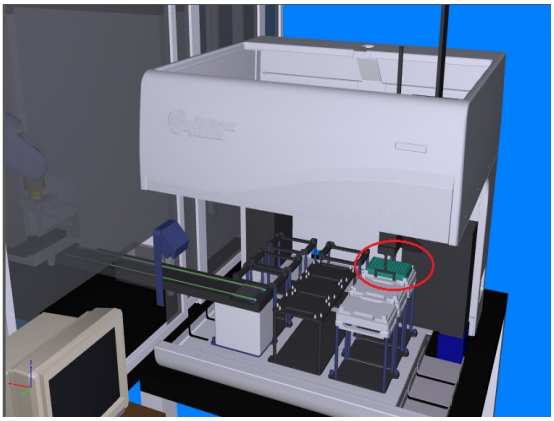


(b) Virtualization result

(12) ConvNX move to BiomekNX-Span.C1 using SelfGrip (Time: 11:57:46)

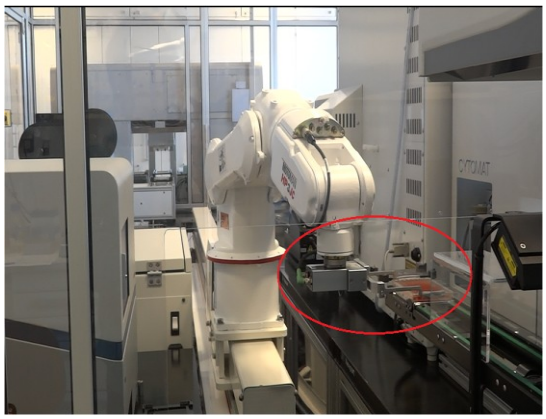


(a) Actions in realistic workstation

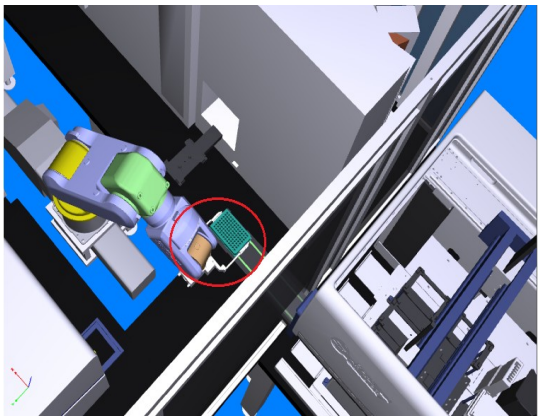


(b) Virtualization result

(13) BiomekNX-Span:LeftPod put to BiomekNX-Span.S1 (Time: 11:57:55)

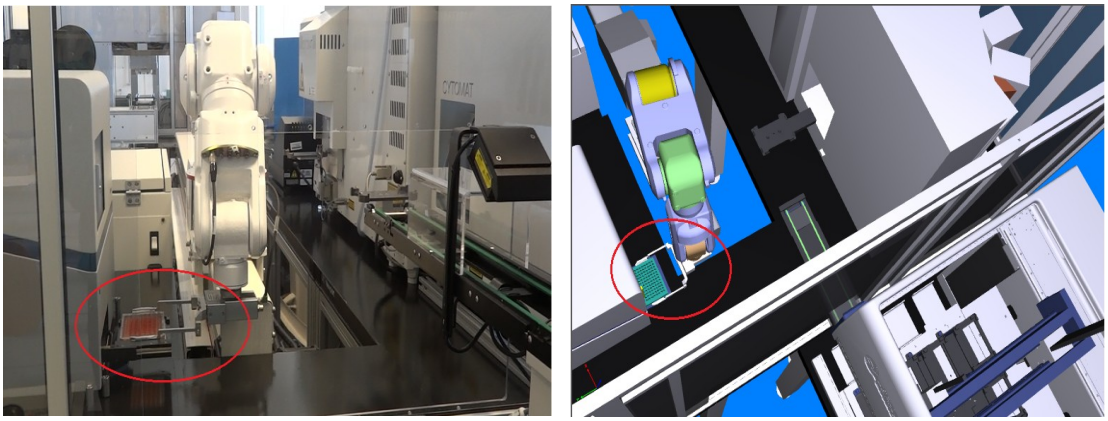


(a) Actions in realistic workstation



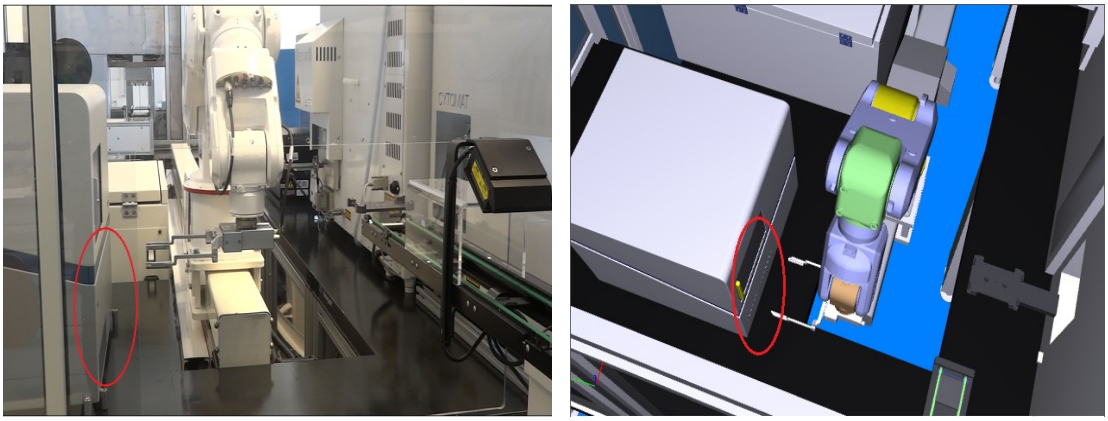
(b) Virtualization result

(14) Motoman move from ConvNX.outer using WideReverse (Time: 11:59:17)



(a) Actions in realistic workstation (b) Virtualization result

(15) Motoman move to PHERAstar.R using WideReverse (Time: 11:59:28)



(a) Actions in realistic workstation (b) Virtualization result

(16) Close PHERAstar.R from put (Time: 11:59:32)

Figure 6.5: Comparation of realistic workstation workflow and virtualization results

6.5 Virtualization Result Transmission

After the virtualization result – a .pdf animation file is generated, the CS triggers its TCP/IP socket to connect with the client DTS and send the file to the DTS in data stream format.

As Figure 6.6 shows, the connection is created, and the file has been sent to the DTS successfully. Figure 6.7 shows the file has been received fully in the DTS side, and been opened in the DTS interface.

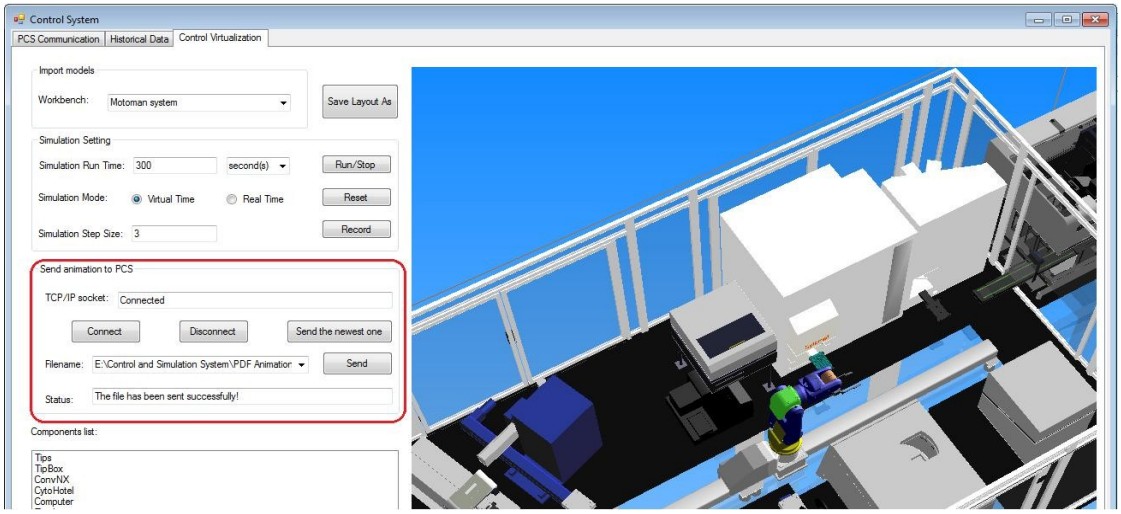


Figure 6.6: Virtualization result transmission in the CS

Figure 6.7: Online feedback information from the CS

Chapter 7 Conclusion and Outlook

7.1 Conclusion

In this dissertation, a virtualization system named VS is developed for experiment workflows of LSA. It integrates LSA hardware, PCS and simulation software to a whole, and realizes the virtualization for a PCS in life science applications. The system could simulate a scheduling workflow for both experiment shows and tests. It saves time and cost for users, and makes the demonstration on LSA experiment workflow flexibly. The achievements and functions of the VS are partly shown in publications [35]-[37], and [92]. They are summarized as followings:

(1) The system integrates the functions of realistic LSA workstations, PCS, and 4D-virtualization by many interfaces. It refers to technologies of SILAS OCX, TCP/IP socket, Visual C#, COM API, Python API and Python script, as well as 3D simulation.

(2) The PCS module drives the LSA workstations working as its design method. In addition, it supplies workflow data for the other modules of VS via SILAS OCX.

(3) The DTS module applies some technologies and realizes corresponding functions as:

   i. applies SILAS OCX technology to get the real-time workflow data from PCS;

   ii. sends the received workflow data also in real time to CS via TCP/IP socket as a server. In this part, the server socket binds its port to wait for and accept the requirement of communication, and encodes the real-time received data one by one in Unicode form for the data transmission towards CS;

   iii. tests communication through sending test data and getting feedback information from CS;

   iv. receives the virtualization result in the form of data stream from CS also via TCP/IP socket as a server. The socket receives the data stream, decodes it back to the .pdf animation format, and automatically opens the animation in its interface.

(4) The CS module realizes following functions by TCP/IP socket, programming and API technologies:

i. connects with the DTS, gets the real-time workflow data from the DTS, and feeds data reception conditions back to the DTS via its TCP/IP socket, which works as a client.

ii. processes the received data and extracts key information for the virtualization;

iii. connects with the simulation software 3DCreate, and drives it working through COM API and Visual C# programming technologies;

iv. extracts the components' information from 3DCreate via COM API;

v. assigns the workflow data to related components and layout, sets parameters and creates behaviors for them through Visual C# programming, COM API and Python API technologies;

vi. creates movements for the workflow data, forms animation via COM API;

vii. connects with DTS, and sends it the virtualization result in data stream form via TCP/IP socket technology.

(5) The VM module applies 3DCreate simulation technology to:

i. creates components which have characters of nodes, behaviors and motions besides of other 3D properties;

ii. teaches components motions and actions by jogging joints or calling Python API;

iii. simulates movements of components synchronously;

iv. be driven by other system via its strong API.

The scientific meanings in the dissertation are demonstrated as followings: (1) it presents an idea to make experiment workflow in LSA laboratory virtualization in real-time, and drive the realistic workstations running as the scheduled process based on the virtualization result; (2) it simulates the real experiment workflow, rather than virtual design; (3) most importantly, among the system modules, the Control System (CS) converts workflow data in text to dynamic parameters, kinematic parameters and actions, which the Simulation Module (SM) can recognize; (4) the system embed Python scripts into Visual C# (VC#), and applies VC# to call Python language to create and set dynamic and kinematic parameters for SM; (5) CS extracts model

parameters from PCS data and assigns them to original 3D models to convert them as components in SM; (6) CS controls SM to generate kinematic trajectories and actions, and then integrates all kinematics in specific sequences to a whole virtualization result; (7) CS controls SM to generate virtualization result as .pdf form, which could demonstrate the virtualization result for customers in dynamitic form; (8) the system developed could display the real-time virtualization result in any occasion—laboratory, meeting room, or conference hall in business trip, etc. It just asks for customers having network; (9) because the technology problems have been solved, the system could applied in any workstation with different PCS, and in any LSA laboratory.

7.2 Outlook

The Virtualization System could already simulate experiment workflows on workstations of LSA, and makes the workstation hardware, PCS and virtualization simulation a whole. However, its functions and virtualization objects are still limited. In the future, the system could be modified and improved to be stronger as followings:

(1) ***Simulate experiment workflow for different PCS:*** At present, the VS aims at the workflow virtualization for PCS SAMI EX. There are many other different PCS in LSA, which also need to realize virtualization for them. The virtualization is the need and trend in LSA development. So it will be more flexible and stronger if the VS could be used in any PCS of LSA.

(2) ***Simulate workflows of workstations synchronously:*** VS could simulate one LSA workstation at one time currently. When there are other workstations in the laboratory, it will be more vivid to show the LSA laboratory to visitors or business partners if the system could simulate all workstations together.

(3) ***Simulate workflows for the whole laboratory system:*** In the future, there are more laboratories integrated as a whole laboratory system, which could be connected by moving robots or some other devices. So the system is hoped to realize visual virtualization on not only one laboratory, but also for a whole laboratory system, such as a laboratory building with many floors.

(4) ***Validation of the developed virtulization system:*** In the Section 6.4 of this dissertation, the validation of the behaviors in the VS has been considered and provided. However, the validation of the time performance has not been included, which will be one of our future tasks.

References

[1] R. C. Anderson, X. Su, G. J. Bogdan, and J. Fenton, “A miniature integrated device for automated multistep genetic assays”, *Nucleic Acids Res.*, vol. 28, no. 12, e 60, 2000.

[2] I. Meyvantsson, J. W. Warrick, S. Hayes, A. Skoien, and D. J. Beebe, “Automated cell culture in high density tubeless microfluidic device arrays”, *Lab. Chip*, vol. 8, no. 5, pp. 717–724, 2008.

[3] A. McCabe, M. Dolled-Filhart, R. L. Camp, and D. L. Rimm, “Automated quantitative analysis (AQUA) of in situ protein expression, antibody concentration, and prognosis”, *J. Natl. Cancer Inst.*, vol. 97, no. 24, pp. 1808–1815, 2005.

[4] F. Kong, L. Yuan, Y. F. Zheng, and W. Chen, “Automatic Liquid Handling for Life Science A Critical Review of the Current State of the Art”, *J. Lab. Autom.*, vol. 17, no. 3, pp. 169–185, 2012.

[5] T. Göpel, F. Härtl, A. Schneider, M. Buss, and H. Feussner, “Automation of a suturing device for minimally invasive surgery”, *Surg. Endosc.*, vol. 25, no. 7, pp. 2100–2104, 2011.

[6] M. Vaqué, A. Arola, C. Aliagas, and G. Pujadas, “BDT: an easy-to-use front-end application for automation of massive docking tasks and complex docking strategies with AutoDock”, *Bioinformatics*, vol. 22, no. 14, pp. 1803–1804, 2006.

[7] J. P. McMullen and K. F. Jensen, “Integrated microreactors for reaction automation: new approaches to reaction development”, *Annu. Rev. Anal. Chem.*, vol. 3, pp. 19–42, 2010.

[8] K. K. Unger, R. Ditz, E. Machtejevas, and R. Skudas, “Liquid chromatography—its development and key role in life science applications”, *Angew. Chem. Int. Ed.*, vol. 49, no. 13, pp. 2300–2312, 2010.

[9] Y. WU, Q. XIE, X. ZHANG, and X. LIU, “Measures to Improve Undergraduates’ Research Experimental Level”, *Res. Explor. Lab.*, vol. 8, pp. 33, 2009.

[10] H. Liu, N. Stoll, S. Junginger, and K. Thurow, “Mobile Robot for Life Science Automation”, *Int. J. Adv. Robot. Syst.*, vol. 10, no.288, pp. 1–14, 2013.

[11] M. Panchal, “The automation of nested clade phylogeographic analysis”, *Bioinformatics*, vol. 23, no. 4, pp. 509–510, 2007.

[12] M. J. He, W. J. Cai, W. Ni, and L. H. Xie, “RNGA based control system configuration for multivariable processes”, *J. Process Control*, vol. 19, no. 6, pp. 1036–1042, 2009.

[13] M. Brundle and M. Naedele, “Security for process control systems: An overview”, *Secur. Priv. IEEE*, vol. 6, no. 6, pp. 24–29, 2008.

[14] M. P. Groover, "Automation, production systems, and computer-integrated manufacturing", *Prentice Hall Press*, pp.257-273, 2007.

[15] M. Vidyasagar, "Control system synthesis: a factorization approach". *Morgan & Claypool Publishers*, pp.36-47, 2011.

[16] A. Mehta, P. Wojsznis, W. K. Wojsznis, and T. L. Blevins, “Integrated model predictive control and optimization within a process control system”, *U.S. Patent* 7,050,863. 2006-5-23.

[17] G. K. Law, D. L. Deitz, T. D. Schleiss, and J. Naidoo, “Integrated electronic signatures for

approval of process control and safety system software objects", *U.S. Patent* 7,076,312. 2006-7-11.

[18] F. Leymann and D. Roller, "Production workflow: concepts and techniques", *Upper Saddle River: Prentice Hall PTR*, 2000.

[19] M. Flattery, "Workflow systems", *Tessella Support Serv. Tech. Rep. April*, 2005.

[20] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities", *Future Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528–540, 2009.

[21] B. Ludäscher, I. Altintas, S. Bowers, .etc, "Scientific process automation and workflow management", *Sci. Data Manag. Challenges Exist. Technol. Deploy. Comput. Sci. Ser.*, pp. 476–508, 2009.

[22] H. A. Reijers and W. M. V. D. Aalst, "The effectiveness of workflow management systems: Predictions and lessons learned", *Int. J. Inf. Manag.*, vol. 25, no. 5, pp. 458–472, 2005.

[23] P. Romano, "Automation of in-silico data analysis processes through workflow management systems", *Brief. Bioinform.*, vol. 9, no. 1, pp. 57–68, 2008.

[24] C. J. Huang, A. J. Trappey, and Y. H. Yao, "Developing an agent-based workflow management system for collaborative product design", *Ind. Manag. Data Syst.*, vol. 106, no. 5, pp. 680–699, 2006.

[25] A. P. Kalogeras, J. V. Gialelis, and C. E. Alexakos, "Vertical integration of enterprise industrial systems utilizing web services", *Ind. Informatics IEEE Trans.*, vol. 2, no. 2, pp. 120–128, 2006.

[26] D. Müller, J. Herbst, and M. Hammori, "IT support for release management processes in the automotive industry", in *Business Process Management*, Springer, pp. 368–377, 2006.

[27] C. V. Trappey, A. J. Trappey, and C. J. Huang, "The design of a JADE-based autonomous workflow management system for collaborative SoC design", *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2659–2669, 2009.

[28] P. Réant, M. Dijos, and F. Arsac, "Validation of a new bedside echoscopic heart examination resulting in an improvement in echo-lab workflow", *Arch. Cardiovasc. Dis.*, vol. 104, no. 3, pp. 171–177, 2011.

[29] K. Kochut, J. Arnold, and A. Sheth, "IntelliGEN: A distributed workflow system for discovering protein-protein interactions", *Distrib. Parallel Databases*, vol. 13, no. 1, pp. 43–72, 2003.

[30] K. Thurow, B. Göde, and U. Dingerdissen, "Laboratory information management systems for life science applications", *Org. Process Res. Dev.*, vol. 8, no. 6, pp. 970–982, 2004.

[31] H. Yin, Y. Gao, H. Yan, and J. Wang, "Simulation Data and Process Management System in the Development of Virtual Prototype", in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, vol. 3, pp. 152–155, 2010.

[32] L. Chun-quan and W. Zhao-hua, "Process management system analysis and design of SMT reflow soldering process", in *Electronic Packaging Technology, 2005 6th International Conference on*, pp. 279–284, 2005.

[33] Q. Zhang, “Development and application of process management system for coal chemical industry”, in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 2, pp. 6709–6713, 2006.

[34] “SAMI workstation EX software”, *Analis*. [Online]. Available: http://www.analis.be/bin/site/render.cgi?id=0066774_item_to_sell. [Accessed: 04-Nov-2013].

[35] Y. Li, S. Junginger, N. Stoll, and K. Thurow, “4D Simulation and Control System for Life Science Automation”, in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pp. 802–807, 2012.

[36] Y. Li, S. Junginger, N. Stoll and K. Thurow, “Real-time Simulation on Workflow of Life Science Automation Laboratory”, *in Computer Science and Automation Engineering (CSAE), 2013 IEEE International Conference on*, pp. 1331-1335, 2013.

[37] Y. Li, S. Junginger, N. Stoll and K. Thurow: Visualization System Development for Life Science Automation. *in Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pp. 191-196, 2013.

[38] P. J., S. J., and W. M., “SILAS Integration Software for Laboratory Automation” .

[39] M. Jansen-Vullers and M. Netjes, “Business process simulation–a tool survey”, in *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark*, 2006.

[40] W. M. van der Aalst, “Business process simulation revisited”, in *Enterprise and Organizational Modeling and Simulation*, Springer, vol.63, pp. 1–14, 2010.

[41] W. M. van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell, “Business process simulation,” in *Handbook on Business Process Management 1*, Springer, pp. 313–338, 2010.

[42] S. L. Mansar and H. A. Reijers, “Best practices in business process redesign: validation of a redesign framework”, *Comput. Ind.*, vol. 56, no. 5, pp. 457–471, 2005.

[43] A. Barker and J. Van Hemert, “Scientific workflow: a survey and research directions”, in *Parallel Processing and Applied Mathematics*, Springer, pp. 746–753, 2008.

[44] W. Aalst, A. Ter Hofstede, and M. Weske, “Business process management: A survey”, *Springer Berlin Heidelberg*, 2003.

[45] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter, “Conventional workflow technology for scientific simulation”, in *Guide to e-Science*, Springer, pp. 323–352, 2011.

[46] W. M. van der Aalst, “Trends in business process analysis”, in *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS) 2007, Madeira, Institute for Systems and Technologies of Information, Control and Communication*, pp. 12–22, 2007.

[47] E. Verbeek, M. van Hattem and H. Reijers, “Protos 7.0: Simulation made accessible”, in *Applications and Theory of Petri Nets 2005*, Springer, pp. 465–474, 2005.

[48] M. J. Blechar and J. Sinur, “Magic quadrant for business process analysis tools”, *Gart. RAS Core Res. Note G* 148777, 2007.

[49] M. Kovács and L. Gönczy, “Simulation and Formal Analysis of Workflow Models”,

Electron. Notes Theor. Comput. Sci., vol. 211, pp. 221–230, 2008.

[50] W. M. van der Aalst, "Business alignment: using process mining as a tool for Delta analysis and conformance testing", *Requir. Eng.*, vol. 10, no. 3, pp. 198–211, 2005.

[51] A. Rozinata, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede,and C.J. Fidge, "Workflow simulation for operational decision support", *Data Knowl. Eng.*, vol. 68, no. 9, pp. 834–850, 2009.

[52] D. Eichhorn, A. Koschmider,Y. Li, P. Sturzel, A. Oberweis, and R.Trunko, "3D Support for Business Process Simulation", in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, vol. 1, pp. 73–80, 2009.

[53] M. H. Zaroukian and A. Sierra, "Benefiting from ambulatory EHR implementation: solidarity, six sigma, and willingness to strive", *J. Healthc. Inf. Manag.*, vol. 20, no. 1, p. 53, 2006.

[54] P. Yellowlees, J. N. Cook, and S. L. Marks, "Can virtual reality be used to conduct mass prophylaxis clinic training? A pilot program", *Biosecurity Bioterrorism Biodefense Strat. Pr. Sci.*, vol. 6, no. 1, pp. 36–44, 2008.

[55] J. J. Saleem, E. S. Patterson, and L. Militello, "Impact of clinical reminder redesign on learnability, efficiency, usability, and workload for ambulatory clinic nurses", *J. Am. Med. Inform. Assoc.*, vol. 14, no. 5, pp. 632–640, 2007.

[56] J. Paul, "VMWare ESX server workload analysis: how to determine good candidates for virtualization", in *Int. CMG Conference*, pp. 483-484, 2007.

[57] J. O. Cleary, M. Modat, and F. C. Norris, "Magnetic resonance virtual histology for embryos: 3D atlases for automated high-throughput phenotyping", *NeuroImage*, vol. 54, no. 2, pp. 769-778, 2011.

[58] M. Graafland, J. M. Schraagen, M. P. Schijven, "Systematic review of serious games for medical education and surgical skills training", *British Journal of Surgery*, vol. 99, no. 10, pp. 1322-1330, 2012.

[59] A. Guerra, "Virtual improvements. Idaho's St. Luke's Regional Medical Center embraced application virtualization to empower users and reduce downtime", *Healthcare informatics: the business magazine for information and communication systems*, vol. 23, no.5, pp. 34, 36, 38, 2006.

[60] Z. Niazkhani, H. Pirnejad, and M. Berg, "The impact of computerized provider order entry systems on inpatient clinical workflow: a literature review", *J. Am. Med. Inform. Assoc.*, vol. 16, no. 4, pp. 539–549, 2009.

[61] G. Bruinsma and R. de Hoog, "Exploring protocols for multidisciplinary disaster response using adaptive workflow simulation", in *International Conference on Information System for Crisis Response and Management (ISCRAM). Newark, New Jersey*, pp. 1–13, 2006.

[62] T. Glatard, C. Lartizien, B. Gibaud, .etc, "A Virtual Imaging Platform for Multi-Modality Medical Image Simulation", *IEEE Trans. Med. Imaging*, vol. 32, no. 1, pp. 110–118, 2013.

[63] S. A. Schendel and C. Lane, "3D Orthognathic Surgery Simulation Using Image Fusion", *Semin. Orthod.*, vol. 15, no. 1, pp. 48–56, 2009.

[64] A. W. Kushniruk and V. L. Patel, "Cognitive and usability engineering methods for the

evaluation of clinical information systems", *J. Biomed. Inform.*, vol. 37, no. 1, pp. 56–76, 2004.

[65] A. W. Kushniruk, M. M. Triola, and E. M. Borycki, "Technology induced error and usability: the relationship between usability problems and prescription errors when using a handheld application", *Int. J. Med. Inf.*, vol. 74, no. 7–8, pp. 519–526, 2005.

[66] A. Boejen and C. Grau, "Virtual reality in radiation therapy training", *Surg. Oncol.*, vol. 20, no. 3, pp. 185–188, 2011.

[67] X. Mao and X. Zhang, "Construction Process Reengineering by Integrating Lean Principles and Computer Simulation Techniques", *J. Constr. Eng. Manag.*, vol. 134, no. 5, pp. 371–381, 2008.

[68] K. W. Chau, M. Anson, and J. P. Zhang, "4D dynamic construction management and virtualization software: 1. Development", *Autom. Constr.*, vol. 14, no. 4, pp. 512–524, 2005.

[69] R. J. Scherer and S. E. Schapke, "A distributed multi-model-based Management Information System for simulation and decision-making on construction projects", *Adv. Eng. Informatics*, vol. 25, no. 4, pp. 582–599, 2011.

[70] M. Kugler and V. Franz, "Development of a Simulation System for the Preparation of Work in Building Construction", in *Computation in Civil Engineering–Tagungsband zur EG-ICE Conference*, pp. 186–193, 2009.

[71] V. Kamat, J. Martinez, and M. Fischer, "Research in Visualization Techniques for Field Construction", *J. Constr. Eng. Manag.*, vol. 137, no. 10, pp. 853–862, 2011.

[72] D. Gaiotto, G. Moore, and A. Neitzke, "Four-dimensional wall-crossing via three-dimensional field theory", *Communications in Mathematical Physics*, vol. 299, no.1, pp. 163-224, 2010.

[73] T. Sider, "Four-dimensionalism: An ontology of persistence and time", *Oxford*, 2003.

[74] K. W. Chau, M. Anson, and J. P. Zhang, "Implementation of visualization as planning and scheduling tool in construction", *Build. Environ.*, vol. 38, no. 5, pp. 713–719, 2003.

[75] H. J. Wang, J. P. Zhang, and K. W. Chau, "4D dynamic management for construction planning and resource utilization", *Autom. Constr.*, vol. 13, no. 5, pp. 575–589, 2004.

[76] K. W. Chau, M. Anson, and J. P. Zhang, "Four-dimensional virtualization of construction scheduling and site utilization", *J. Constr. Eng. Manag.*, vol. 130, no. 4, pp. 598–606, 2004.

[77] K. W. Chau, M. Anson, and D. D. De Saram, "4D dynamic construction management and visualization software: 2. Site trial", *Autom. Constr.*, vol. 14, no. 4, pp. 525–536, 2005.

[78] T. Anderson, L. Peterson, and S. Shenker, "Overcoming the Internet impasse through virtualization", *Computer*, vol. 38, no. 4, pp. 34-41, 2005.

[79] J. Bond, D. Frush, and E. Samei, "Simulation of anatomical texture in voxelized XCAT phantoms", in *SPIE Medical Imaging*, p. 86680N–86680N, 2013.

[80] X. Tian, X. Li, and W. P. Segars, "Patient-and cohort-specific dose and risk estimation for abdominopelvic CT: a study based on 100 patients", in *SPIE Medical Imaging*, p. 83131R–83131R, 2012.

[81] K. Taguchi, Z. Sun, and W. P. Segars, "Image-domain motion compensated time resolved 4D cardiac CT", in *Medical Imaging*, pp. 651016–651016, 2007.

[82] W. P. Segars, G. Sturgeon, and S. Mendonca, “4D XCAT phantom for multimodality imaging research”, *Med. Phys.*, vol. 37, p. 4902, 2010.

[83] W. P. Segars, S. Mori, and G. T. Y. Chen, “Modeling respiratory motion variations in the 4D NCAT phantom”, in *Nuclear Science Symposium Conference Record, 2007. NSS’07. IEEE*, vol. 4, pp. 2677–2679, 2007.

[84] W. P. Segars, M. Mahesh, and T. Beck, “Validation of the 4D NCAT simulation tools for use in high-resolution x-ray CT research”, in *Proc. of SPIE Vol*, vol. 5745, pp. 828-834, 2005.

[85] W. P. Segars, M. Mahesh, and T. J. Beck, “Realistic CT simulation using the 4D XCAT phantom”, *Med. Phys.*, vol. 35, p. 3800-3808, 2008.

[86] W. P. Segars, B. M. W. Tsui, and E. C. Frey, “Extension of the 4D NCAT phantom to dynamic X-ray CT simulation”, in *2003 IEEE Nuclear Science Symposium Conference Record*, vol. 5, pp. 3195–3199, 2003.

[87] C. Kim, H. Kim, and T. Park, “Applicability of 4D CAD in civil engineering construction: Case study of a cable-stayed bridge project”, *J. Comput. Civ. Eng.*, vol. 25, no. 1, pp. 98–107, 2010.

[88] J. Zhang, and D. Li, “Research on 4D virtual construction and dynamic management system based on BIM”, in *Proceedings of the International Conference on Computing in Civil and Building Engineering, ICCBE*, pp.78-83, 2010.

[89] L.-S. Kang, H.-S. Moon, and S.-Y. Park, “Improved link system between schedule data and 3D object in 4D CAD system by using WBS code”, *KSCE J. Civ. Eng.*, vol. 14, no. 6, pp. 803–814, 2010.

[90] W. Li, G. Joós, and J. Bélanger, “Real-time simulation of a wind turbine generator coupled with a battery supercapacitor energy storage system”, *Ind. Electron. IEEE Trans.*, vol. 57, no. 4, pp. 1137–1145, 2010.

[91] M. Karkee, B. L. Steward, and A. G. Kelkar, “Modeling and real-time simulation architectures for virtual prototyping of off-road vehicles”, *Virtual Real.*, vol. 15, no. 1, pp. 83–96, Mar. 2011.

[92] Y. Li, S. Junginger, N. Stoll, and K. Thurow, “4D simulation system for laboratory workflow of life science automation”, in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pp. 1886–1890, 2012.

[93] B. Rooks, “Robotics outside the metals industries”, *Ind. Robot Int. J.*, vol. 32, no. 3, pp. 205–208, 2005.

[94] M. N. Rooker, T. Strasser, and G. Ebenhofer, “Modeling flexible mechatronical based assembly systems through simulation support”, in *IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 452–455*, 2008*.

[95] B. Ludäscher, I. Altintas, and S. Bowers, “Scientific process automation and workflow management”, *Sci. Data Manag. Challenges Exist. Technol. Deploy. Comput. Sci. Ser.*, pp. 476–508, 2009.

[96] B. Ludäscher, I. Altintas, and C. Berkley, “Scientific workflow management and the Kepler system”, *Concurr. Comput. Pr. Exp.*, vol. 18, no. 10, pp. 1039–1065, 2006.

[97] J. G. Hollands, and C. D. Wickens, "Engineering psychology and human performance",

Prentice Hall New Jersey, 1999.

[98] R. H. Dieck, "Measurement uncertainty: methods and applications". *Research Triangle Park, NC: Instrument Society of America*, 1992.

[99] E. Deelman, D. Gannon, and M. Shields, "Workflows and e-Science: An overview of workflow system features and capabilities", *Future Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528–540, 2009.

[100] H. V. Westerhoff and B. O. Palsson, "The evolution of molecular biology into systems biology", *Nat. Biotechnol.*, vol. 22, no. 10, pp. 1249–1252, 2004.

[101] W. M. van der Aalst, "The application of Petri nets to workflow management", *J. Circuits Syst. Comput.*, vol. 8, no. 01, pp. 21–66, 1998.

[102] T. Oinn, P. Li, and D. B. Kell, "Taverna/myGrid: aligning a workflow system with the life sciences community", in *Workflows for e-Science*, Springer, pp. 300–319, 2007.

[103] J. W. Hong and S. R. Quake, "Integrated nanoliter systems", *Nat. Biotechnol.*, vol. 21, no. 10, pp. 1179–1183, 2003.

[104] S. Finger and J. R. Dixon, "A review of research in mechanical engineering design. Part II: Representations, analysis, and design for the life cycle", *Res. Eng. Des.*, vol. 1, no. 2, pp. 121–137, 1989.

[105] R. Brooks, "A robust layered control system for a mobile robot", *Robot. Autom. IEEE J.*, vol. 2, no. 1, pp. 14–23, 1986.

[106] "World News," *J. Assoc. Lab. Autom.*, vol. 12, no. 6, pp. A14–A43, 2007.

[107] J. G. Houston and M. Banks, "The chemical-biological interface: developments in automated and miniaturised screening technology", *Curr. Opin. Biotechnol.*, vol. 8, no. 6, pp. 734–740, 1997.

[108] S. D. Garrett, D. J. Appleford, and G. M. Wyatt, "Production of a recombinant anti-parathion antibody (scFv); stability in methanolic food extracts and comparison to an anti-parathion monoclonal antibody", *J. Agric. Food Chem.*, vol. 45, no. 10, pp. 4183–4189, 1997.

[109] V. M. Bohaychuk, G. E. Gensler, and R. K. King, "Evaluation of detection methods for screening meat and poultry products for the presence of foodborne pathogens", *J. Food Prot.*, vol. 68, no. 12, pp. 2637–2647, 2005.

[110] G. M. Banowetz, J. R. Hess, and J. G. Carman, "A monoclonal antibody against the plant growth regulator, abscisic acid", *Hybridoma*, vol. 13, no. 6, pp. 537–541, 1994.

[111] D. Katz, W. Shi, M. Wildes, and J. K. Hilliard, "Automation of serological diagnosis for herpes B virus infections using robot-assisted integrated workstations", *J. Assoc. Lab. Autom.*, vol. 7, no. 6, pp. 108–113, 2002.

[112] D. B. Kaber, N. Segall, and R. S. Green, "Using multiple cognitive task analysis methods for supervisory control interface design in high-throughput biological screening processes", *Cogn. Technol. Work*, vol. 8, no. 4, pp. 237–252, 2006.

[113] R. S. Green, "Cognitive Task Analyses for Life Science Automation Training Program Design", *ProQuest*, 2008.

[114] H. Dong, M. Goldberg, and D. Nguyen, "Automated high-throughput microarray system",

US20040191807 A1, 2003.

[115] "SAMI EX CUSTOMER TRAINING" [Online]. https://www.beckmancoulter.com/wsrportal/wsr/research-and-discovery/products-and-services/research-automation/sami-scheduling-software/index.htm.

[116] "SILAS Class Presentation" [Online]. https://www.beckmancoulter.com/wsrportal/bibliography?docname=BR-8270C.pdf.

[117] A. Morbidoni, C. Favi, and M. Germani, "CAD-Integrated LCA Tool: Comparison with dedicated LCA Software and Guidelines for the improvement", *Glocalized Solutions for Sustainability in Manufacturing. Springer Berlin Heidelberg,* pp. 569-574, 2011.

[118] L. Q. Dong. "Applications of Three-dimensional CAD in Mechanical Design", *Coal Technology,* vol. 9, p. 8, 2009.

[119] A. Staranowicz, and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software". *In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments,* p. 56. ACM, 2011.

[120] "Comparison of 3D computer graphics software" [Online]. http://en.wikipedia.org/wiki/Comparison_of_3D_computer_graphics_software.

[121] "Comparison of computer-aided design editors" [Online]. http://en.wikipedia.org/wiki/Comparison_of_computer-aided_design_editors.

[122] "Comparison of 3D computer graphics software" [Online]. http://en.wikipedia.org/wiki/Comparison_of_3D_computer_graphics_software.

[123] O. Roulet-Dubonnet and P. A. Nyen, "A method and application to simulate and validate manufacturing control systems based on a discrete manufacturing simulation platform", in *Industrial Applications of Holonic and Multi-Agent Systems*, Springer, pp. 152–162, 2013.

[124] J. Krüger, V. Katschinski, and D. Surdilovic, "PISA: Next Generation of Flexible Assembly Systems-From Initial Ideas to Industrial Prototypes", in *Robotics (ISR), 41st International Symposium on and 6th German Conference on Robotics (ROBOTIK)*, pp. 1–6, 2010.

[125] L. Ferrarini and C. Veber, "3D graphic simulation of flexible manufacturing systems with Day Dream Daemon and 3DCreate", in *6th IEEE International Conference on Industrial Informatics*, pp. 1401–1406*,* 2008.

[126] M. Sacco, G. Dal, and F. Milella, "Virtual factory manager", *Virtual and Mixed Reality-Systems and Applications*, Springer Berlin Heidelberg, pp. 397-406, 2011.

[127] M. Rooker, T. Strasser, and G. Ebenhofer. "Modeling flexible mechatronical based assembly systems through simulation support", *IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA), pp. 452-455, 2008.

[128] B. Rooks. "Robotics outside the metals industries", *Industrial Robot: An International Journal*, vol. 32, no. 3, pp.205-208, 2005.

[129] "Visual Components 3DCreate 2012 - User Manual and Reference Guide", *Visual Components,* 11-Sep-2011.

[130] L. Dupuy L, T. Fourcaud, and P. Lac, "A generic 3D finite element model of tree anchorage integrating soil mechanics and real root system architecture", *American Journal of Botany,* vol. 94, no. 9, pp. 1506-1514, 2007.

[131] I. Dressier, M. Haage, and K. Nilsson, "Configuration support and kinematics for a reconfigurable Gantry-Tau manipulator", *2007 IEEE International Conference on Robotics and Automation*, pp. 2957–2962*,* 2007.

[132] T. Shi, "A Robot Simulation of the Tower of Hanoi Puzzle using OpenRAVE with Sensor Feedback", Vanderbilt University, U.S., 2013.

[133] J. L. Rojas, "Autonomous cooperative assembly by force feedback using a control basis approach", Vanderbilt University, U.S., 2009.

[134] "Yaskawa Motoman Robotics, Solutions in Motions, HP3JC data sheet", *Yaskawa America, Inc.*, 2011 DS-264-D.

[135] H. Zhu, C. Li, and J. Gu. "Implementation of Paralleling Genetic Annealing Algorithm on Grid", *3rd IEEE International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 471-474, 2010.

[136] C. Wögerer, H. Bauer, and M. Rooker, "LOCOBOT-low cost toolkit for building robot co-workers in assembly lines", *Intelligent Robotics and Applications*, Springer Berlin Heidelberg, pp. 449–459, 2012.

[137] S. Zenoni, A. Ferrarini, and E. Giacomelli, "Characterization of transcriptional complexity during berry development in Vitis vinifera using RNA-Seq", *Plant Physiol.*, vol. 152, no. 4, pp. 1787–1795, 2010.

[138] R. Darken, P. McDowell, and E. Johnson, "Projects in VR: the Delta3D open source game engine", *Comput. Graph. Appl. IEEE*, vol. 25, no. 3, pp. 10–12, 2005.

[139] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone", in *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living*, Springer, pp. 796–799, 2009.

[140] J. Heilala, J. Montonen, and O. Väätäinen, "Life cycle and unit-cost analysis for modular reconfigurable flexible light assembly systems", *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 222, no. 10, pp. 1289–1299, 2008.

[141] B. Uekert and D. Dancy, *State courts and elder abuse: Ensuring justice for older Americans*. Williamsburg, VA: National Center for State Courts, 2007.

[142] M. Wang, J. Hong, and W. Wu, "An improved model of motorized spindle for transient thermo-structural analysis", in *Assembly and Manufacturing (ISAM), 2013 IEEE International Symposium on*, pp. 1–7, 2013.

[143] A. Ji, Z. Xu, and L. Lu, "Customization Research on Collaborative Simulation of the Parts of Truck-Mounted Crane", *Applied Mechanics and Materials*, , vol. 201, pp. 569-573, 2012.

[144] A. Pichler, P. Barattini, and C. Morand, "Tailor Made Robot Co Workers Based on a Plug&Produce Framework", in *Robotics in Smart Manufacturing*, Springer, pp. 113–126, 2013.

[145] L. Lindqvist, "Unified infrastructure for simulation, communication and execution of robotic systems", M. SC. thesis, *Helsinki University of Technology,* Finland, 2007.

[146] F. Wickert, "A test for personal goal-values", *J. Soc. Psychol.*, vol. 11, no. 2, pp. 259–274, 1940.

[147] K. Khairunnisa, "Kinematic Analysis Of 6-Axis Comau Robot", *Universiti Teknikal*

Malaysia Melaka, Malaysia, 2008.

[148] D. Palm, K. Johansson, and A. Ozin, "Molecular epidemiology of human pathogens: how to translate breakthroughs into public health practice, Stockholm, November 2011", *Euro Surveill*, vol. 17, no. 2, pp. 1-4, 2012.

[149] J. R. Alvarado, R. V. Osuna, and R. Tuokko, "Distributed simulation in manufacturing using high level architecture", *Micro-assembly Technologies and Applications*, Springer US, pp. 121–126, 2008.

[150] T. R. Keeney, C. Bock, and L. Gold, "Automation of the SomaLogic proteomics assay: a platform for biomarker discovery", *Journal of the Association for Laboratory Automation*, vol. 14, no. 6, pp. 360-366, 2009.

[151] M. A. Jensen, L. Jauregui, and R. W. Davis, "A Rapid, Cost-Effective Method of Assembly and Purification of Synthetic DNA Probes> 100 bp", *PloS One*, vol. 7, no. 4, e34373, pp.1-4, 2012.

[152] J. DelProposto, C. Y. Majmudar, and J. L. Smith, "Mocr: a novel fusion tag for enhancing solubility that is compatible with structural biology applications", *Protein Expr. Purif.*, vol. 63, no. 1, pp. 40–49, 2009.

[153] M. de Arruda, V. I. Lyamichev, and P. S. Eis, "Invader technology for DNA and RNA analysis: principles and applications", *Expert Rev. Mol. Diagn.*, vol. 2, no. 5, pp. 487–496, 2002.

[154] D. E. Burger, L. Wang, and L. Ban, "Novel automated blood separations validate whole cell biomarkers", *PloS One*, vol. 6, no. 7, e22430, pp.1-11, 2011.

[155] K. Allison, "The first automated high content screening system", *J. Assoc. Lab. Autom.*, vol. 8, no. 3, pp. 27–29, 2003.

[156] P. P. So, E. Zeldich, and K. I. Seyb, "Lowering of amyloid beta peptide production with a small molecule inhibitor of amyloid-β precursor protein dimerization", *Am. J. Neurodegener. Dis.*, vol. 1, no. 1, pp. 75-87, 2012.

[157] D. Meyre, K. Proulx, and H. Kawagoe-Takaki, "Prevalence of loss-of-function FTO mutations in lean and obese individuals", *Diabetes*, vol. 59, no. 1, pp. 311–318, 2010.

Appendixes

Appendix A: Biomek FX

The Biomek® FX Laboratory Automation Workstation is a multi-axis liquid-handling instrument used in the drug discovery laboratory. Its system components described below correspond to the components shown in Fig. A. 1.

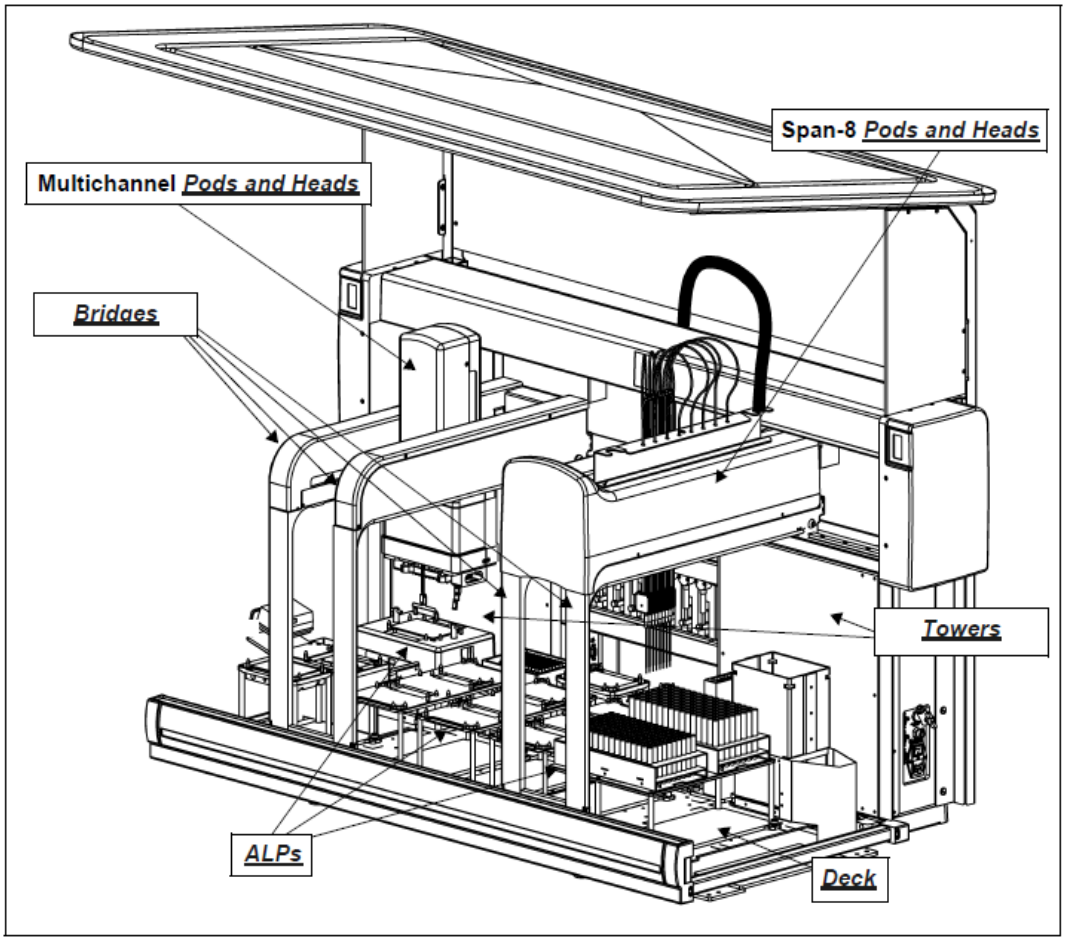


Figure A.1 Biomek FX main components

A.1 Towers

The Biomek FX towers form the rear vertical and horizontal uprights of the base unit along which the bridges travel in the X-axis (see as Fig. A.2). The links for master control of Biomek FX, plus utility hook-ups and ALP connections, are on the towers. Built into the towers are green and amber indicator lights that keep users aware of the current operational status of Biomek FX instrument.

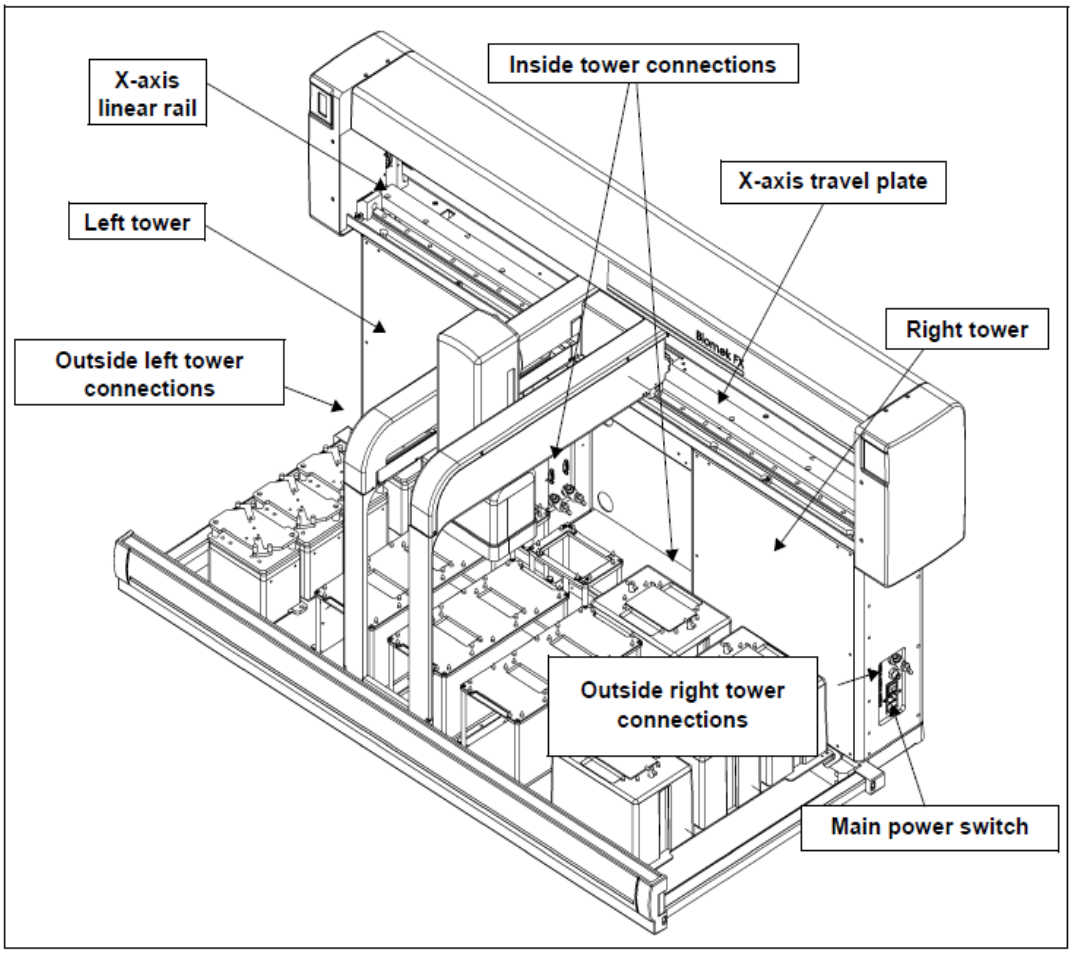


Figure A.2 Main components and connections of the Biomek FX towers

A.2 Bridges

As Fig.A.3 shows, the Biomek FX bridges are structures that move in the X-axis, and the pods are self-contained components supported and positioned by the bridges. The bridges hold the pods and move them in the Y- (front to back) and Z-axes (up and down). One or two bridges are available on the Biomek FX instrument to create a single- or dual-pod instrument. In a dual-pod system, the pods can work together to expand liquid-handling capabilities.

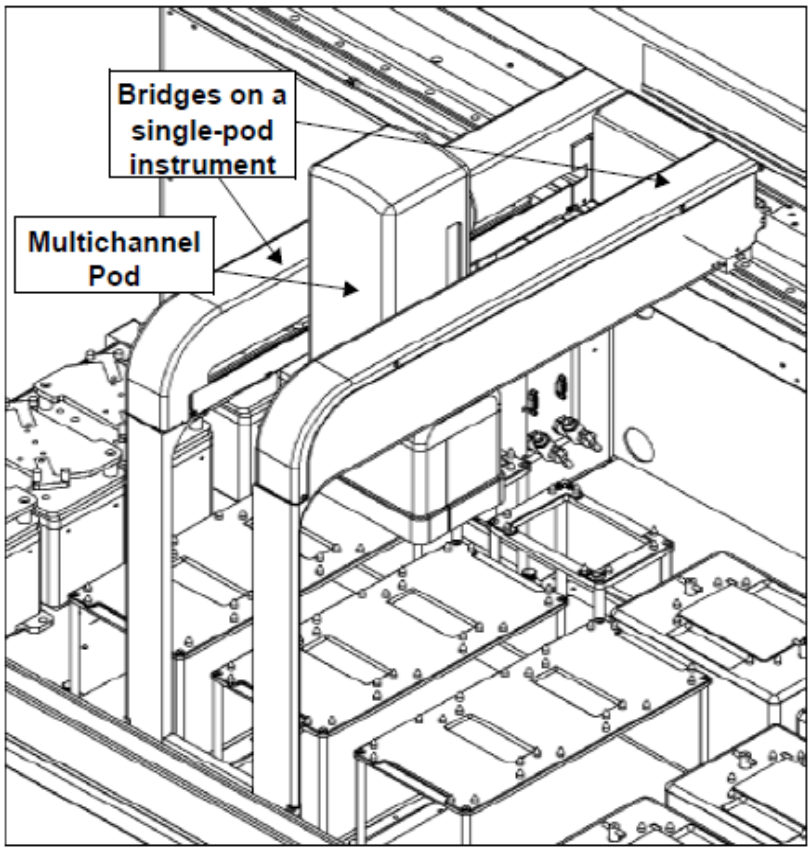


Figure A.3 Bridges move in the X-axis, hold and move pod in the Y- and Z-axes

A.3 Multichannel Pod

The Multichannel Pod is a self-contained working unit installed on the right, left, or both bridges of Biomek FX. The Multichannel Pod is a full microplate replication tool incorporating a gripper and interchangeable heads to accommodate a variety of functions. It interacts with ALPs located over the entire deck area of Biomek FX.

The main components of the Multichannel Pod are (as Fig. A.4):

• Pod — Houses operating mechanism, pneumatic air line, communication and electrical power connections to the base unit, and moves in the Y-, Z-, and D-axes for liquid-handling functions.

• Interchangeable Heads — Holds mandrels and tips for performing full-plate replication.

• Gripper — Grip labware along the long side of the labware.

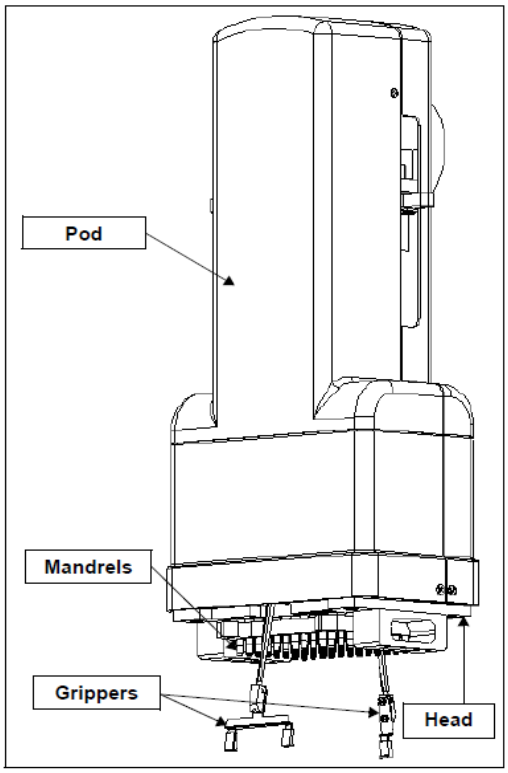


Figure A.4 Multichannel Pod — main components

The Multichannel Pod performs movements in the Y-, Z-, and D-axes (see as Table A.1).

Table A.1 Multichannel Pod Axes Movement

Axis	Movement
Y-	Entire pod moves front-to-back.
Z-	Entire pod moves up-and-down.
D-	Up-and-down aspirate/dispense, disposable tip shucking, and close/open

A.4 Automated Labware Positioners

Automated Labware Positioners (ALPs) are removable and interchangeable platform structures that are installed on the Biomek deck to allow automated assays to be performed.

ALPs are either:

• Passive ALPs — some hold labware in place on the deck while others act as receptacles for by-products from methods, such as system fluid and disposed tips, tip boxes, and labware.

OR

• Active ALPs — contain mechanisms that may hook to power and/or air sources for mechanical operation, such as tip loading, tip washing, mixing/stirring, shaking, and precisely positioning labware.

Figure A.5 shows the ALPs of Biomek FX. Among them, the positions P1-P16 are passive ALPs that hold labware on the deck during liquid-handling procedures; TL1 is an active ALP that loads disposable tips onto a 96-well head or a 384-well head mounted on a Multichannel Pod; the Shuttle is a passive position for setting one end of SMCShuttle; WS1 is an active ALP that washes fixed or disposable tips on the deck; the Top1 and Res1 are belong to one single active ALP featuring a reservoir that can be drained and refilled automatically using steps in a Biomek Software method or manually using Advanced Manual Control.

Figure A.5 ALPs of Biomek FX

Appendix B: Biomek NX Span-8

The Biomek® NX Span-8 Laboratory Automation Workstation is a multi-axis liquid-handling instrument used in the life sciences and bioresearch laboratory. The components of the workstation are described as Fig. B. 1.

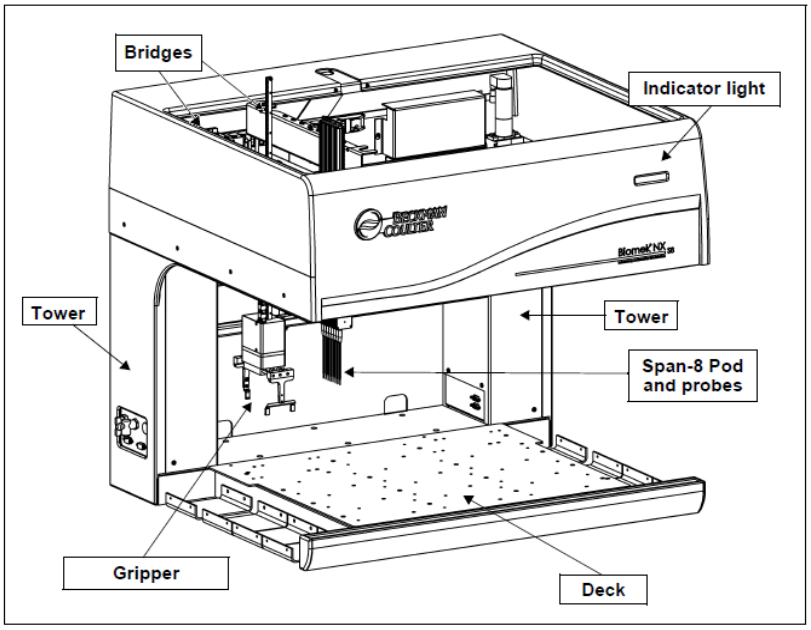


Figure B.1 Biomek NX with Span-8 Pod and optional gripper

B.1 Bridges

As Fig.B.1 shows, there are two groups of bridges that move along the X-axis in Biomek NX. They have separated components for different tasks: one for gripper movement, and the other one for the Span-8 pod and probes. The bridges hold their components and move them in the Y- (front to back) and Z-axes (up and down).

B.2 Span-8 Pod

The Span-8 Pod is a self-contained working unit installed on the Biomek NX Span-8. It is a liquid-handling tool capable of performing liquid transfers from test tubes and large pieces of labware to smaller pieces of labware, or vice versa.

The Span-8 Pod houses the operating mechanisms, communications, and electrical connections to the base unit. It performs liquid transfers using a series of eight probes that can move independently in the Z-axis, pipette independently in the D-axis with

the assistance of the pumps, and span from 9mm to 20mm between the probes in the S-axis (see Table B.1). The main components of a Span-8 Pod are shown as Fig. B. 2.

Table B.1. Span-8 Pod Axes Movement

Axis	Movement
Y-	The probes move simultaneously front to back.
Z-	The probes move up and down independently.
D-	The aspirate/dispense action is controlled independently by the pumps.
S-	The span (or distance) between the eight probes can expand and collapse.

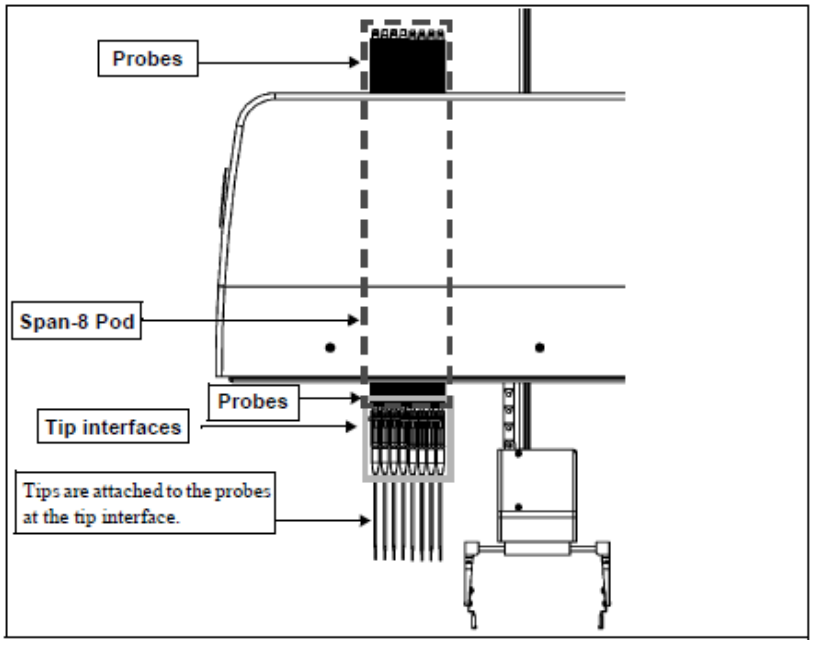


Figure B.2. Span-8 Pod with gripper (detailed view)

B.3 Gripper

The Biomek NX gripper tool grasps and moves labware from one location on the Biomek NX deck to another. The gripper moves independently of the pod and can move in the Y-axis (back to front) and Z-axis (up and down). It can also rotate to access positions that are oriented at different angles in relation to the front of the Biomek NX instrument.

As Fig.B.3 shows, the gripper tool contains two sets of finger pads:

• A double finger pad located to the front of the tool

• A single finger pad located to the back of the tool

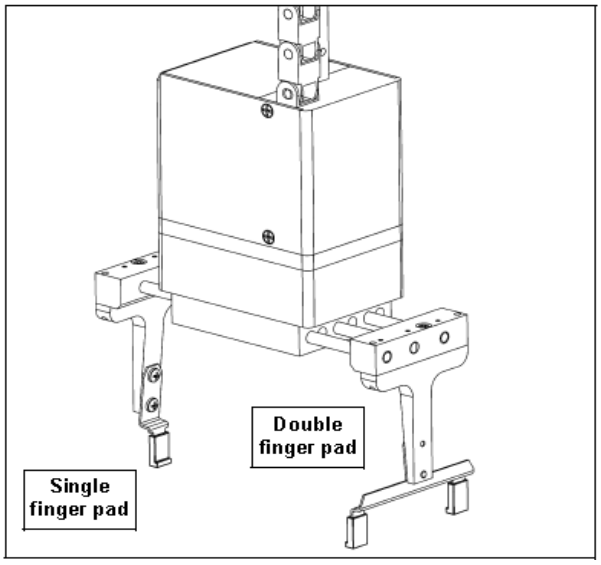


Figure B.3. Factory-installed gripper tool

B.4 Automated Labware Positioners

As the description of Biomek FX ALPs, there are also some passive and active ALPs installed on the deck of the Biomek NX instrument to allow automated assays to be performed. Figure B.4 shows the ALPs of Biomek NX Span-8: C1 is an active ALP for setting ConvNX; P1-P8 are passive ALPs for holding labware on the deck during liquid-handling procedures; S1-S3 are the ALP for the device Teleshake to shake a microplate in the chosen direction (as Fig. B.5); W1 is a passive ALP used to wash fixed tips on the probes of a Span-8 Pod while the reservoir side of the Span-8 Tip Wash ALP is used to dispose of system fluid used; TR1 is a Half-Position Disposal ALP with slide, which provides a means to dispose of tips, tip boxes, and labware.

Figure B.4 ALPs of Biomek NX Span-8

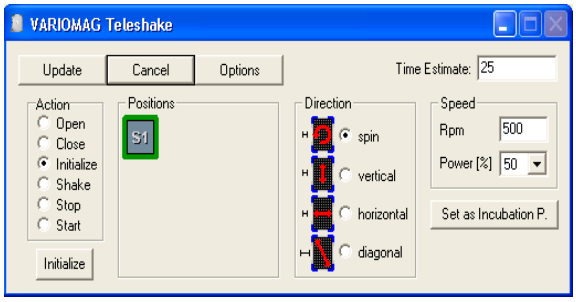


Figure B.5 Action commands of the Teleshake

Declaration

This dissertation 'Virtualization System for Life Science Automation Laboratory' is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work of this dissertation has been done by myself under the guidance of Prof. Dr.-Ing. habil. Kerstin Thurow and Prof. Dr.-Ing. Norbert Stoll at the University of Rostock, Germany. Also the dissertation has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Rostock, 29 January, 2014

Yanfei Li

Theses

1. Virtualization System (VS) for Life Science Automation (LSA) integrates many technologies which include PCS, SILAS OCX, TCP/IP socket, COM API, Python API, Visual C# programming, Python scripts, and 3D Simulation.

2. VS for LSA has four modules: Process Control System (PCS), Data Transfer System (DTS), Control System (CS), and Virtualization Module (VM).

3. VS for LSA could not only simulate real-time experiment workflow, but also simulate historical workflow data for demonstrations in conference or business occasions.

4. PCS generates real-time workflow data for VS via SILAS integration system, and transmits the data to DTS via SILAS OCX.

5. DTS receives workflow data in real time from PCS by calling SILAS OCX, and sends the data to CS via TCP/IP socket synchronously. The TCP/IP socket in DTS woks as a server for sending workflow data.

6. DTS receives simulation results in the format of .pdf animation from CS via TCP/IP socket, which also works as a server. The animation results are received as data stream, and transferred back to .pdf file. They are opened automatically in the DTS interface after being transferred wholly.

7. CS receives the real-time workflow data from DTS via TCP/IP socket, which works as a client. At the same time, CS extracts key information from the complex workflow data to do preparation for simulation.

8. CS links and drives the simulation software 3DCreate to do simulation by calling its COM API and Python API.

9. COM API of 3DCreate is called to create behaviors, trajectories, and actions for components, set properties and parameters of components, and invoke commands of the software for file operations, simulation setting, etc. The COM API is called and programmed by Visual C# in CS.

10. Python API of 3DCreate is called and programmed in Python language. It is edited in Python scripts, which is a behavior of a component. It’s called to define

properties of a component, create basements for robots and servos, and perform different actions with robots and their tools.

11. CS assigns the workflow data to related components in the workstation layout, teaches robots and servos motions and actions, and creates specified sequences for motions and actions of components to form experiment workflow as the data depicts.

12. CS forms the orderly movements of components to a whole experiment workflow of LSA once the real-time data is generated fully in PCS. It saves the workflow with 3D trajectories to .pdf animation, and backs up the simulation result as 3DCreate layout for modification.

13. PDF animation could be showed in Adobe Reader. It could be demonstrated in various views and sizes. When it is opened in the reader, the sizes of components and distances in the layout could be measured. Customers could get all information about the layout just from the .pdf animation.

14. CS embeds 3DCreate OCX into its interface, which could intuitively show the workstation layout and simulation results to users in real time with the workflow data received.

15. Once the simulation result (.pdf animation file) is generated, CS extracts the result and sends it to DTS by TCP/IP socket. That is the second socket in CS, which is also works as a client. The socket works to connect the server socket, and send the .pdf animation file in the form of data stream when the connection requirement is accepted.

16. Beside processing and simulating real-time workflow data, CS could also extract and process historical workflow data, and then simulate it by driving 3DCreate working. The system generates the simulation result and shows it in its interface to the users, or generates .pdf animation file for demonstrations of LSA laboratories to business partners.

Abstract

In Life Science Automation (LSA), Process Control System (PCS) could realize automatic control for experiment workflows in workstations of laboratories. However, the virtualization for workflows is lacked and becomes more and more necessary. This dissertation developed a Virtualization System (VS) to simulate LSA experiment workflows virtually in a flexible and fast way, which solves the virtualization problem for LSA experiments.

Virtualization System integrates technologies of PCS, TCP/IP socket, database, Visual C#, Python Script, Visual Component 3DCreate and 3D modeling. It mainly has four modules to realize their separate functions in the system: (1) ***PCS*** module for generating workflow data, and sending the data through its SILAS OCX, as well as driving realistic workstations running sync with the virtualization process; (2) Data Transfer System (***DTS***) module for receiving the workflow data from PCS by calling its SILAS OCX, transferring the workflow data to Control System (CS) module and getting the virtualization result from CS both by TCP/IP socket technology, as well as showing the virtualization result automatically at the side of PCS by calling Adobe PDF Reader DLL; (3) ***CS*** module for getting workflow data via TCP/IP socket, and simulating not only real-time but also historical workflow data in 4D animation form through controlling the Virtualization Module (VM) by calling and expanding its COM API and Python API, as well as sending the virtualization result to DTS via its TCP/IP socket; (4) ***VM*** for generating 3D models of devices in Solidworks, and transferring the 3D models to components and layouts in 3DCreate, teaching robots and servos with motions and actions, as well as being controlled to generate the virtualization result via its COM API and Python API.

In the VS, DTS and CS are developed by Visual C# to connect workstation hardware, PCS and 3D simulation tool as a whole. The system realizes virtualization on LSA experiment workflows not only in real time but also in historical. It supplies a vivid and flexible 4D virtualization on LSA experiment workflows for customers, and makes demonstrations for LSA laboratories more convenient.

Zusammenfassung

Im Bereich der Life Science Automatisierung (LSA) kann die automatische Steuerung von Arbeitsabläufen bei Experimenten an Laborarbeitsplätzen durch ein Prozesssteuerungssystem (PSS) realisiert werden. Es fehlte jedoch bisher die immer wichtiger werdende Möglichkeit, die entsprechenden Arbeitsabläufe zu visualisieren. Im Rahmen dieser Dissertation wurde ein Virtualisierung Steuersystem (VS) entwickelt, das die Arbeitsabläufe bei Experimenten im Bereich der LSA schnell und flexibel simuliert und so das Problem fehlender Virtualisierung löst.

Das VS integriert die folgenden Technologien: PSS, TCP/IP Sockets, Datenbanken, Visual C#, Python Script, Visual Components 3DCreate und 3D Modellierung. Es besteht im Wesentlichen aus vier Modulen, die im System verschiedene Aufgaben übernehmen: (1) Das PSS-Modul generiert Daten über die Arbeitsabläufe, die es anschließend über sein SILAS OCX sendet und es steuert die realen Arbeitsplätze synchron zum virtualisierten Prozess. (2) Das Datentransfersystem (DTS) empfängt die Arbeitsablauf-Daten vom PSS-Modul indem es dessen SILAS OCX aufruft und überträgt diese Daten an das Steuerungssystem (SS). Das DTS nutzt die TCP/IP Socket-Technologie um die Virtualisierung Ergebnisse vom SS zu erhalten und die Adobe PDF Reader DLL um die Virtualisierung Ergebnisse im PSS anzuzeigen. (3) Das SS-Modul erhält Arbeitsablauf-Daten über TCP/IP-Sockets und verarbeitet Echtzeitdaten ebenso wie gespeicherte Arbeitsabläufe zu einer 4D-Simulation durch Steuerung des Virtualisierungsmoduls (VM) mittels Aufruf und Expansion von dessen COM API und Python API. Anschließend werden die Virtualisierung Ergebnisse über TCP/IP-Sockets an das DTS gesendet. (4) Die Funktion des VM besteht darin, 3D-Modelle von Geräten in Solidworks zu generieren und diese in 3DCreate in Komponenten und Layouts umzuwandeln. Hier werden die Bewegungen und Aktionen von Robotern und Servos definiert. Die Berechnung der Virtualisierung Ergebnisse wird gesteuert über eine COM API und eine Python API.

In dem VS wurden das DTS und SS in Visual C# implementiert um die Arbeitsplatz-Hardware mit dem PSS und dem 3D-Simulationswerkzeug zu verbinden. Das System realisiert die Virtualisierung von Arbeitsabläufen bei LSA-Experimenten sowohl für Echtzeitdaten als auch Archivdaten. Es liefert eine anschauliche und flexible 4D-Virtualisierung der Arbeitsabläufe in LSA-Experimenten für Kunden und vereinfacht Demonstrationen für LSA-Labore.