Heterogeneous Modeling & Design of a Robot Arm Control System

Antonio Yordán-Nones
University of Puerto Rico, Mayagüez
djstone@caribe.net

ABSTRACT

Robots and embedded control systems are traditionally programmed to perform automated tasks, yet accomplishing complicated and dynamically changing tasks will typically require the use of specialized computational resources and software systems. In order to design these sophisticated control systems the use of computational modeling and simulation techniques is necessary to ensure that final implementations of the system perform optimally. Since even the simplest robotic control systems are intrinsically a mixture of heterogeneous sub-systems, each focusing on a particular task, we need a means of modeling the different aspects of the system accurately and intuitively. Using the Ptolemy II software environment, one can visually design and model elaborate control systems by incorporating heterogeneous models of computation that govern the interaction of multiple components within the system. In this paper we design a robot arm controller component for Ptolemy II that enables various other Ptolemy components to interface and control the robot arm in novel ways, hereby expanding the scope of innovative uses and future applications. We then illustrate an implementation of a Ptolemy model that uses an X-10 remote control, communicating via an X-10 network, to control the movements of a Lynx-5 Robot Arm.

1. INTRODUCTION

As pervasive computing begins to establish its presence in future society, the demand for more elaborate embedded systems becomes evident. With each new generation of embedded applications pushing the boundaries of its predecessors in both complexity and performance, there is an imminent need for computational tools and software environments that will unleash a new wave of flexibility and reliability. Systems designers and engineers typically rely on software modeling and simulation techniques to attain proper functionality from their systems prior to production, and with current, more demanding applications calling for enhanced performance capabilities, modeling software must now be as precise as ever while maintaining a straightforward and intuitive approach to design.

Embedded software designers will generally select an underlying *model of computation* to accurately and efficiently represent the interaction of components in a system under inspection. For example, an electro-mechanical system could be represented under continuous dynamics, while a digital communications system may be more appropriately modeled under the discrete-event domain. Sometimes the inherent complexity in designing an embedded system may lead to the mix of computational models and domains. Systems that employ multiple modes of operation and numerous computational domains can best be described as *hybrid models* [8].

The focus of study in the Ptolemy project [1,7] is on modeling, simulating, and designing embedded systems that require heterogeneous and hybrid modeling, concurrent interaction of components, and real-time responsiveness. The components in Ptolemy are referred to as *actors* and the models of computation that govern the interaction between actors are called *directors*[1]. Typically a designer will decide on a model of computation that best represents a sub-system, look through the library of pre-existing actors in the Ptolemy II architecture, and visually construct a model of the system based on the relations between components, after which, the system can be simulated and checked for adequate behavior. If additional complexity is needed, then a hierarchal approach to designing the system is possible by combining different domains and composite actors.

The importance of component-based design, or actor-oriented design (under Ptolemy), is crucial in extending the capabilities of modern, software systems design environments. Since a clear priority when designing actors for use in Ptolemy is maintaining domain and data polymorphism, actors become powerful, reusable components, and although the behavior of actors and their interactions in multiple domains may vary, designing complicated embedded systems with simple building-blocks is an incredibly practical approach. Ptolemy extends component-based design principles by accurately managing the time continuum and maintaining concurrency among actors, features that are especially useful when designing control systems software.

One of the main focuses of this paper is designing and modeling a control system for a robot arm in the Ptolemy

[1] Actors and directors in Ptolemy are mostly coded in Java.

II software environment. Some examples of tasks the robot arm would be capable of performing under Ptolemy control are, moving to user specified 3D coordinates, exercising a sequence of pre-programmed movements, executing automated tasks under finite state machine control, and following generated path trajectories (possibly generated from a computer vision model in Ptolemy). We also propose experimental uses of the robot to help test new features in Ptolemy and leverage performance of embedded software control over hardware components. The intended goal is to create a general-purpose actor that can be used for any practical application that falls within the scope of the robot's functionality, interface parameters, and inherent control limitations. In the implementation discussed in this paper, we include the newly designed actor in a Ptolemy model where the movements of the robot arm are controlled by an X-10 remote control communicating through the X-10 network. Since what must be specified in the Ptolemy model to control the robot arm is the desired position and orientation of the end-effector, the X-10 remote control will have designated keys placed in a cardinal-direction orientation and the movements of the robot will be interpreted accordingly.

2. CONTROL SYSTEMS AND ROBOTICS

Embedded control systems are now most implemented in software, mainly due to the fact that performing complicated, real-time tasks requires speed, accuracy, dedicated computational power, and the added benefit of being easily tweaked by systems designers to find more efficient future implementations and alternate design configurations. Robotic control systems operate in a similar manner, for example, software control of a robot manipulator will usually require modeling continuous time, differential equations to define the governing *dynamics* of the system as well as deriving the *kinematic*s, or laws of motion, and *workspace* that determine the movements of the robot manipulator. The calculated dynamics and kinematics will then need to be interfaced with the actual robot, thus allowing communication with the robot's physical control mechanisms and actuators, all this without taking into consideration the use of sensors on the robot. Depending on the mathematical complexity of the robots defining model, physical geometry, dynamics, and kinematics, any one particular robotic configuration will greedily make use of its computational resources in order to assure precise robot response and control [4].

2.1 Geometry of the Lynx-5 Robot Arm

To apply modeling techniques in Ptolemy and demonstrate embedded software control of a robotic manipulator we chose the Lynx-5 Robot Arm (Figure 1). The Lynx-5 [2] is an articulated manipulator (RRR), with 2 planar *links* in an elbow-like configuration, 4 *degrees-of-freedom*, and 4 rotational *joints*, each controlled by dedicated servo motors. We will refer to each joint as the *base*, *shoulder*, *elbow*, and *wrist*, respectively. The Lynx-5 also has an *end-effector* which is a parallel jaw gripper attached to the wrist joint, and although the gripper has a dedicated servo motor to control grasping objects, the servo is not used for the purposes of this paper.

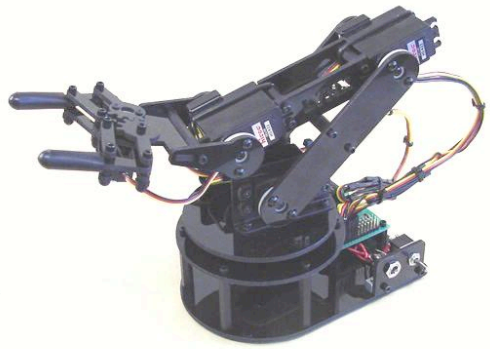


Figure 1: Lynx-5 Robot Arm

2.2 Inverse Kinematics

The general objective for controlling the movements of the robot arm in Ptolemy is to specify a Cartesian coordinate in 3D space (in centimeters) and then have the end-effector reach this position with a given orientation. The origin of the global coordinate system is chosen to be directly under the base rotational joint on the ground plane, so in order to find the robots final configuration, the inverse kinematics of the Lynx-5 must be calculated. If we were to use a kinematic approach to moving the robot arm we would specify the angle and coordinate system of each joint as a *transformation matrix* and use matrix multiplication to arrive at a final coordinate position. Although this is easily implemented, possibly as a separate actor in Ptolemy, the inverse kinematic approach was chosen to allow more flexibility and more advanced uses of the robot arm [9].

When calculating the inverse kinematics of any robot configuration one may easily run into certain limitations and ambiguities that will lead to undesired functionality of the robot arm. One of the principle issues is determining the exact orientation of the joints, when many possible configurations can render the same functionality[2]. Since the Lynx-5 robot arm has a limited workspace, most ambiguous configurations were not taken into account because the restrictions on the rotation of the joints would not permit reaching these configurations. In addition, a restriction was imposed on

[2] For a 2 link, elbow-like robotic configuration this is known as the elbow-up and elbow-down configuration problem.

the inverse kinematic equations that specifies the orientation of the end-effector must be parallel to the ground, thus specifying the geometric equations. The other predominant issue that must be worked around is distinguishing between reachable points in space and non-reachable points that lie outside of the dexterous workspace of the robot arm. When this is the case the robot joints will rotate to their maximum range in the correct direction. Derivation of the inverse kinematic equations for this specific type of operation can be seen in Appendix A.

2.3 Discretization of Joint Rotations

The four joints on the Lynx-5 Robot Arm are each controlled by dedicated servo motors connected to a serial servo controller hence the interface used to control the Lynx-5 from a computer is through the serial port. A sequence of 3 consecutive unsigned bytes is sent to the serial servo controller specifying a default sync byte first, the joint servo identifier second, and the desired position of the servo last. Since the rotational position specified when moving each servo is a discrete value with absolute positioning, each angle must first be transformed into a discrete value representing the final servo position. Each servo on the Lynx-Five Robot Arm is constrained to a range of motion of 90 degrees, or of 180, degrees when a jumper is appropriately placed on the external serial servo controller. For the implementation described at the end of this paper we used a 90 degree range of motion to eliminate having to choose from ambiguous robot configurations, as aforementioned. Finally when the final position of all joints is known, we step through the movements of the joints in a smooth and sequential way.

3. MODELING CONTROL SYSTEMS IN PTOLEMY II

Modeling embedded control systems for robotics in Ptolemy is very flexible since there are a variety of models of computation to choose from - some more suitable than others - depending on the task at hand. If the entire implementation of a complex control system were to be modeled using strictly Ptolemy II actors and directors, an initial approach would be to create a *modal model* (closely resembling a finite state machine), that describes the different modes of operation, and then add *refinements*, or sub-models particular to a specific state of operation. Each refinement can then introduce a model of computation and sub-level of hierarchy that will impose a prescribed mode of communication between the components of the sub-system. For example, if one were to model the dynamics that control the actuators in a robot, the best alternative for solving the differential equations regarding velocity and acceleration would be to operate under the continuous time (CT) domain, using the extensive library of actors that compute mathematical functions, expressions, derivations, integrations, and differential equations to build the continuous time model.

Often times, as is the case of the Lynx-5 Robot Arm, one must deal with discrete information and digital communication between hardware and software control. This functionality is best implemented in the discrete-event (DE) domain, an appropriate model of computation for control systems since it supports a strict, time-oriented approach. In DE, all events and communication signals have a globally recognized time-stamp, therefore, the interaction of actors under the DE director places all events in a global queue and then processes them following a time-stamp order. It is worth noting that communication between actors in all available domains in Ptolemy is carried out by sending and receiving data in the form of *tokens*.

3.1 Design of the Robot Arm Actor

The approach taken in this paper to achieve more detailed software control of the Lynx-5 Robot Arm, while simultaneously adding portability and convenience, was to create a Ptolemy actor [3,6] that would take a coordinate position in 3D space as an input token, analyze the motion, and then output the stream of bytes representing the sequence of incremental movement commands to the servos controlling the robot joints (Figure 2). The stream of bytes is sent to the appropriate serial port actor which outputs to the external servo controller. The main reason for outputting a stream of bytes, versus sending the output directly to the serial port from within the actor's code-base, is to enable precise control over the motion of the rotational joints from other parts of the model and not the actor. This can usually be achieved by inserting a time activated queue (in the case of DE) to process the information and retransmit periodically, as specified by the system designer.

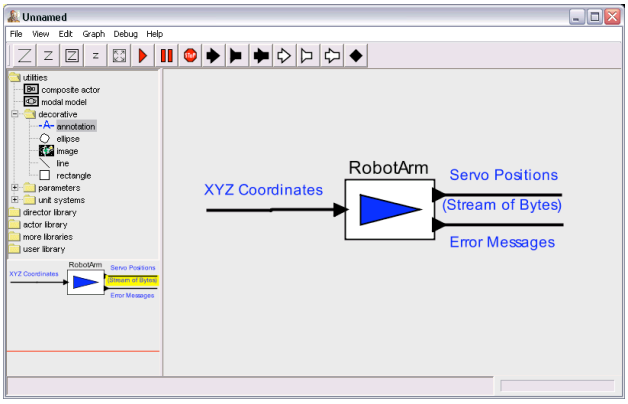


Figure 2: Robot Arm actor with I/O ports

There are many design [5] advantages for creating a robot arm actor to control the Lynx-5 rather then building a dedicated Ptolemy model for this particular control system. One of the advantages is achieving portability

and reusability of the robot arm. Since we wish to experiment with atypical uses and applications for the robot arm, designing an actor abstracts the mathematical complexity of the actual kinematics and discretization calculations necessary for moving the arm appropriately. Instead, finding a general interface format that establishes control of the Lynx-5 via input coordinates and ultimately connects the stream of output servo steps to the serial controller gives sufficient flexibility for the actor to be useful under numerous applications. An additional and useful feature included in the robot arm actor for simulation in a Ptolemy model, is that the actor can produce appropriate error messages when the input coordinate is outside of the robot's reachable workspace.

3.2 Control of the Robot Arm – DE

Another issue that is of importance when controlling the robot arm in a Ptolemy model is correctly synchronizing the movements of the robot arm to physical time. The best representative model of computation for manipulating and using the robot arm actor is using the discrete-event (DE) domain since the protocol for interaction of components and communication in DE works by scheduling events according to their position in a global event queue. In this manner, we can best model the application for the robot arm by sequentially organizing the components (the robot actor being central) to communicate in real-time. Also worth noting is that when simultaneous events occur (i.e. matching time-stamps) in concurrent components of an embedded system, they are handled deterministically.

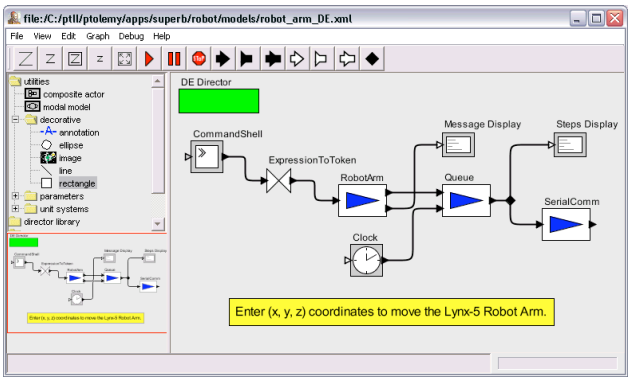
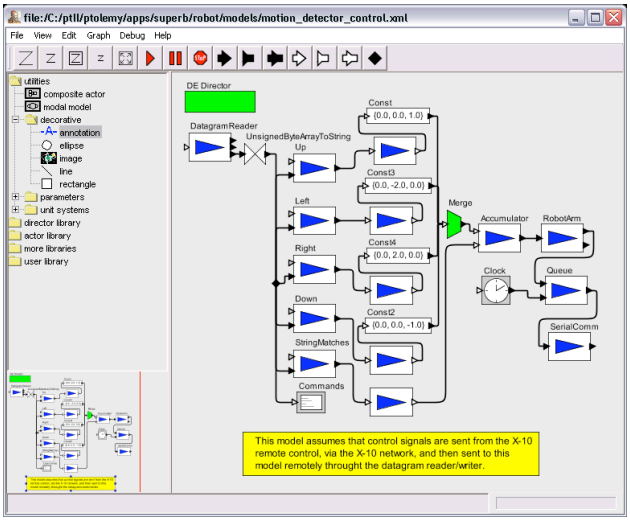


Figure 3: Implementation of a robot arm model

In the design of the robot arm control system the dynamics were not taken into consideration due to the detached and unreliable nature of the servo motors, and the limited capability of the serial interface to the robot arm. Instead of directly establishing the velocity and(or) acceleration for the rotation of each joint, the final position of each servo motor corresponding to the degrees of rotation for the calculated angle is established and then a stream of positions with fixed steps will alternately increment the servo motor of each joint in the correct direction. This allows the designer the flexibility needed for comprehensive control of the applied torque and in effect, stepping of the servo motors. As mentioned before, this type of functionality can be achieved by including a queue actor for precisely timed control of sending commands to the servo motors.

4. CASE STUDY

A unique application that integrates the concepts of heterogeneous modeling and interaction of concurrent components in an unusual way is illustrated by constructing an embedded system that controls the Lynx-5 Robot Arm from a wireless X-10 remote control. The buttons on the remote control are customized to output cardinal-directions that move the robot arm on a user selected course - mainly up, down, left, and right movements. This is a novel application that combines communication between two Ptolemy models running on two separate machines and communicating by means of a datagram writer and reader. On the first machine, the X-10 network listener reads X-10 commands and sends the data to the second machine, which first translates the instruction and then steps the robot arm in the user-specified direction by a constant step value. One of the drawbacks of this control system is the generated time lag when communicating over the X-10 network, which makes issuing commands from the remote control neither time accurate nor robust.



5. CONCLUSION

We presented the different models of computation useful for embedded control systems design, modeling, and simulation, with emphasis on robotic systems. We found that achieving a high level of complexity in the design of such a system is straightforward and highly intuitive using the Ptolemy II software environment. Following the component-oriented design principles

described in this paper, we designed and tested, under different models and configurations, a newly introduced robot arm actor that abstracts the kinematics and tedious calculations from the actual model and enables the systems designer to experiment with unusual, innovative applications. Although briefly described in this paper, one aspect that would further extend the interface to the robot arm would be to input a set of rotations for each joint, instead of the desired position in space, which has been left for future work.

Acknowledgements

I would like to thank the members of the Center for Hybrid and Embedded Software Systems (CHESS) at the University of California, Berkley, for their continuous support of my research. Special thanks go to my faculty mentor, Professor Edward A. Lee, who gave me the opportunity to participate in Berkeley's SUPERB-IT 2003 Program and conduct research with his group, and my graduate mentor, Xiaojun Liu, who helped throughout the stages of design and review.

6. REFERENCES

[1] Shuvra S. Bhattacharyya, Elaine Cheong, John Davis II, Mudit Goel, Christopher Hylands, Bart Kienhuis, Edward A. Lee, Jie Liu, Xiaojun Liu, Lukito Muliadi, Steve Neuendorffer, John Reekie, Neil Smyth, Jeff Tsay, Brian Vogel, Winthrop Williams, Yuhong Xiong, Haiyang Zheng, " Heterogeneous Concurrent Modeling and Design in Java ,"Memorandum UCB/ERL M02/23, University of California, Berkeley, CA USA 94720, August 5, 2002.

[2] Jeppe Mikkelsen, "A Machine Vision System Controlling a Lynxarm Robot along a Path ," University of Cape Town, South Africa, October 28, 1998.

[3] Jie Liu, Johan Eker, Xiaojun Liu, John Reekie, and Edward A. Lee, "Actor-Oriented Control System Design ," submitted to IEEE Transactions on Control System Technology, special issue on "Computer Automated Multi-Paradigm Modeling," March, 2002 (revised version, 2003.)

[4] Xiaojun Liu, Jie Liu, Johan Eker, and Edward A. Lee, " Heterogeneous Modeling and Design of Control Systems ," in Software-Enabled Control: Information Technology for Dynamical Systems , Tariq Samad and Gary Balas (eds.), Wiley-IEEE Press, April 2003.

[5] Jie Liu, Johan Eker, Jörn W. Janneck, Xiaojun Liu, and Edward A. Lee, " Actor-Oriented Control System Design: A Responsible Framework Perspective ," to appear in IEEE Transactions on Control System Technology , Draft version, March, 2003 (revised from previous version .)

[6] Jörn W. Janneck, " Actors and their composition ," Memorandum UCB/ERL M02/37, University of California at Berkeley, December 18, 2002.

[7] Edward A. Lee, " Overview of the Ptolemy Project ," Technical Memorandum UCB/ERL M01/11 , University of California, Berkeley, March 6, 2001.

[8] Rajeev Alur, Franjo Ivancic, Jesung Kim, Insup Lee, and Oleg Sokolsky, "Generating Embedded Software from Hierarchal hybrid Models ," LCTES'03, June 11-13, 2003, San Diego, California, USA.

[9] Mark W. Spong and M. Vidyasagar, Robot Dynamics and Control , John Wiley & Sons, Inc, 1989, New York, New York, USA.

APPENDIX A:
Geometric Equations and Conversion of Rotations to Servo Positions

Input Coordinate

(x, y, z)

Base:

$$\theta = \arctan\left(\frac{y}{x}\right)$$

$$for\left[-\frac{\pi}{4} \le \theta \le \frac{\pi}{4}\right]$$

Shoulder:

$$x' = \sqrt{x^2 + y^2}$$

$$z' = z - BASE\_HEIGHT - SHOULDER\_HEIGHT$$

$$x' = x' - ARM\_LENGTH$$

$$h = \sqrt{(z')^2 + (x')^2}$$

$$\phi = \arctan\left(\frac{z'}{x'}\right) - \arccos\left(\frac{LOWER\_LINK^2 + UPPER\_LINK^2 - h^2}{2 * LOWER\_LINK * h}\right)$$

$$for\left[\frac{\pi}{6} \le \phi \le \frac{\pi}{2}\right]$$

Elbow:

$$\psi = \arccos\left(\frac{LOWER\_LINK^2 + UPPER\_LINK^2 - h^2}{2 * LOWER\_LINK * h}\right)$$

$$for\left[-\frac{5\pi}{6} \le \psi \le -\frac{\pi}{6}\right]$$

Wrist:

$$\varpi = -\phi - \psi$$

$$for\left[-\frac{\pi}{4} \le \varpi \le \frac{\pi}{3}\right]$$

NOTE: The ranges of rotation for each joint is rough estimate.

Conversion of Rotations to Servo Positions:

$$Base\ Position = -\left(\frac{BASE\_HOME}{\pi/4}\right)(\theta) + BASE\_HOME$$

$$Shoulder\ Position = \left(\frac{SHOULDER\_HOME}{\pi/3}\right)\left(\phi - \frac{\pi}{2}\right) + SHOULDER\_HOME$$

$$Elbow\ Position = -\left(\frac{ELBOW\_HOME}{\pi/3}\right)\left(\frac{\pi}{2} + \psi\right) + ELBOW\_HOME$$

$$Wrist\ Position = \left(\frac{WRIST\_HOME}{\pi/4}\right)(\varpi) + WRIST\_HOME$$