A Software Architecture for Internet-Aided Design of Linkages

C. L. Collins
Department of Mechanical Engineering
University of California
Riverside, CA 92521

J. M. McCarthy, A. Perez, and H. Su
Department of Mechanical and Aerospace Engineering
University of California
Irvine, CA 92697

February 7, 2002

Abstract

This paper describes a Java-based architecture for a computer-aided-design system for spatial linkages. These linkages are formed by a workpiece supported by one or more serial chains. The task of the linkage is defined by specifying rigid positions for the workpiece that approximate a desired workspace. Because there are many serial chains available for this design process, the architecture is organized to allow incorporation of individually defined serial chain classes and synthesis routines. Here we present the basic components of the design system as well as the structure of the mechanism analysis and synthesis classes. Their integration is demonstrated using a set of classes developed for three position synthesis of spatial RR chains and TS chains. Serial chains can be combined to form more complex parallel linkage architectures, each of which must be analyzed for display and animation. The system provides for incorporation of specially tailored kinematic analysis routines for arbitrary spatial linkage prototypes. The result is an extensible system that supports the design of planar, spherical, and spatial serial and parallel linkage topologies.
1 Introduction

The focus of commercial computer-aided design (CAD) software has been on tools for shape representation and manipulation, assembly and detail drawing creation, and integration with computer-aided manufacturing (CAM) and analysis packages. This focus is clearly represented in high end systems such as Dassault's CATIA, and PTC's Pro/ENGINEER. Mid-range modeling applications such as SolidWorks, AutoCAD, IronCad, and Vellum Solids have export capabilities or additional plug-in interfaces to CAM packages such as SurfCam or MasterCam, FEA packages such as Algor or Nastran, and dynamic simulation packages such as WorkingModel or ADAMS.

While this software is extremely powerful and versatile, it does not help automate the step in the design process described as the function-to-form transformation. This is the creative phase in the product development cycle where the designer synthesizes the geometric model of individual parts and assemblies to realize the particular function and satisfy the particular requirements.

One specific area of design research that has well developed tools for this function-to-form transformation is the theory of mechanism synthesis. Mechanism synthesis theory can be divided into methodologies for type and number synthesis, and dimensional synthesis [12]. Type synthesis allows the designer to match the input requirements to a space of possible mechanism types, such as gear systems, cam systems, or linkage systems. Once a mechanism type is identified, number synthesis can be used to generate the set of potential topological configurations available for further consideration. For example if an input specification is compatible with a planar linkage, number synthesis would generate a variety of linkage topologies with single and multiple loops. Once a mechanism topology is identified, the designer can use the tools of dimensional synthesis to identify the values of geometric parameters that satisfy the motion requirements.

1.1 Linkage Design Software

For linkages, the synthesis problem is particularly well defined. The input specification, or task, describes the desired workspace or motion capabilities, and is typically represented by a set of discrete positions defined by some convenient representation of rigid body displacements, such as a set of $4 \times 4$ homogeneous transformation matrices. From this description, the geometric constraints associated with the linkage architecture are resolved to determine its physical dimensions. This requires the solution of a complex set of nonlinear equations. These problems range in size and complexity depending on the number of specifications, and the type of linkage being synthesized. Dimensional synthesis research is well established and solution algorithms have been presented for a variety of architectures and input conditions (see for example, Suh and Radcliffe [14] and McCarthy [8]).

While there are numerous special purpose algorithms for synthesizing a variety of linkage architectures, no general purpose design tool exists that integrates them. Each synthesis problem typically has different input specifications and different output requirements. This characteristic has lead to the development of special-purpose software systems for linkage design. Research software has been developed to aide the design of planar (Erdman and Gustafson [2], Ruth and McCarthy [11]), spherical (Ruth and McCarthy [11], Furlong,
Vance and Larochelle [3]), and spatial (Larochelle [6], Kihonge, Vance and Larochelle [5]) linkages. These packages have a number of features in common. They all provide an interface for specifying discrete workspace positions, they solve for the dimensions of a single architecture, and they provide some kind of graphical visualization of the resulting linkage designs. In general, they are developed for specific computing platforms and have no export capabilities.

Two commercial packages for planar linkage design are also available. SyMech software [15] allows the designer to synthesize planar mechanisms from within Pro/E. Wireframe kinematic diagrams are generated that can be used to create parts. WATT software [19] is a stand-alone package that can synthesize 4, 6 and 8 bar planar linkages and display them as stick models; it has very limited import and export capabilities.

1.2 Objectives for Linkage Design Software

We desire to close the gap between commercial CAD software that aides the development of part and assembly models, and special purpose linkage design software that is limited to a single mechanism architecture. The goal is to develop a software system that embodies the early phases of the design process by providing a natural environment for synthesizing and evaluating linkage designs based on functional specifications and constraints. The system should be flexible, extensible, and provide output that is easily integrated with current CAD systems. The following is a general list of design requirements consistent with the basic objectives. Thus we seek to develop linkage design software that will: 1) provide an interface for motion specifications, 2) provide an interface for general constraints, 3) accommodate different linkage architectures, 4) accommodate different synthesis algorithms, 5) provide an interface to display multiple solutions, 6) provide an interface for evaluation of solutions, 7) provide an interactive graphical display of design solutions, 8) provide an interface for extended analysis and optimization tools, 9) provide an interface for CAD model input and output, and 10) provide multiple platform accessibility.

1.3 Paper Overview

The goal of this paper is the definition of a software architecture that can incorporate a variety of linkage synthesis routines into an interactive web-based design system. This architecture is modeled on the engineering design process, and provides a graphical user interface to integrate features common to linkage design systems. We begin with a discussion of the dimensional synthesis problem for linkages and highlight how it can serves as a template for our interactive design system. We then provide an overview of the functionality and implementation of Synthetica, an extensible Java applet for linkage synthesis. The flexibility and interactive capabilities of the applet arise from three key components: the Java programming language and platform specification, the OpenGL libraries and GL4Java bindings, and the architecture and interface specifications that we have developed. We then address how a user can utilize Synthetica as a design environment, or extend Synthetica to include new linkage architectures and synthesis routines. We conclude with an evaluation of the current prototype software.
2 The Synthetica Design Methodology

We have adopted a focused design methodology, that while not the most general approach, applies to a large variety of linkage architectures. It is basically a procedure for dimensional synthesis with a finite number of rigid body positions. We consider a set of positions as a discrete representation of the desired workspace of a linkage. When formulating the design problem, we typically specify a set of positions and identify a particular type of linkage we wish to design. We then compare the number of positions to the number of free parameters in the design to determine what additional constraints are needed to fully specify a solution to the problem. In general, we synthesize serial chains, which we call linkage primitives, and combine them to form parallel mechanisms. The primitives can have the same topology, or different. If they are designed to reach the same set of positions, the combined closed chain linkage is guaranteed to be assembled at those positions.

Typically, serial chain synthesis routines lead to a finite set of solutions. We form a discrete set of closed chains using every combination of open chains from our solution sets. In order to determine what resulting linkage satisfies our quantitative and qualitative specifications we need then analyze them for their properties. For the 4R planar or spherical linkage, for example, we can calculate each mechanism type and display it graphically. Only certain mechanism types lead to “good” solutions. This post-synthesis evaluation formalism allows the designer to generate a large number of linkages that cannot otherwise be optimized at this time. In particular, it supports platform linkage topologies where a set of serial chains act together on a common link or workpiece.

Figure 1 shows an RRTS linkage designed to move through three positions. First, a pair of RR chains was synthesized. Next, a set of TS chains was synthesized. Then, the solutions from each set were combined and evaluated to determine what RRTS linkage could move through the three positions smoothly.
3 Software Overview

The functionality of the Synthetica Applet is based on our model of the linkage design process. We confine the initial scope of the software to the specification and synthesis of serial chain architectures (planar, spherical and spatial) and architectures with a platform topology (that is linkages that have a single ground and single moving link connected by serial subchains). The key feature of the software architecture is that it can support the integration of multiple linkage topologies and multiple synthesis routines.

The key components and flow of information for Synthetica are shown in Figure 2. The first version of the software is in the form of a Java applet which is designed to run under a web browser or applet viewer on Windows, and Mac OS operating systems. The details of the implementation are discussed in the next section.

Once the applet loads, the user is presented with the first key GUI element called the Design Matrix. This is a tabular representation of the mechanism topologies and synthesis routines available for specifying and creating new mechanism designs. This table is dynamically constructed from information about the available classes. At this point the user can also load customized classes and include them in the selection matrix. The purpose of this
module is to allow the user to direct the design from either a task oriented perspective, or a linkage topology perspective. From this point, the designer can either manually input the dimensions of the linkage under consideration, or specify the parameters of the task required by the selected synthesis method.

Once a linkage topology and synthesis method is selected, a Task Specifier module is dynamically generated to allow the user to input the required position information, and any additional constraints required by the synthesis method. Once this information is provided, the synthesis method is called, and a list of solutions is generated and made available in the Solution Browser. At this point the user can change the selected linkage/synthesis routine, modify the task and constraints, or directly modify the linkage dimensions. In addition, if an evaluation routine is available for the linkage, the designer can select to use it to help rank and classify the solutions.

Once a particular solution is selected, a 3D model of the linkage can be generated, displayed, and manipulated interactively. In the first version of the software, the designer can change the material, color, and texture of the linkage components. From here, the user can select a different linkage to view, or continue to refine the task. Data from the project, can be saved for further analysis, or for reinstancing the state of the project.

The GUI elements, 3D graphics capabilities, dynamic class loading, and data file management are all implemented using standard Java packages, third-party packages, and custom designed packages.

4 Synthetica Packages

The objectives of our linkage design software are well suited for an implementation using Java. The Java Virtual Machine is available on all common computing platforms. The Java language and API specifications have become fairly stable, and it also naturally provides a flexible and powerful set of GUI elements to enhance user interaction (see the Java Tutorial [1, 18] for an introduction). In addition to the natural networking capabilities built in to the Java API, it also has a facility for utilizing dynamically linked libraries on the client machine. This not only provides a means to more efficiently implement particularly complex computations, but it also allows Java programs to use existing core libraries such as OpenGL. The object-oriented nature of Java also allows us to design a set of class specifications that enables collaborative development of the special purpose synthesis algorithms and linkage analysis routines. The platform independent nature of Java allows these classes to be developed independently by researchers with their own software development resources and dynamically integrated with the applet/application at run-time.

Synthetica has four major components organized into Java packages as shown in Figure 3. The main program integrates and coordinates the flow of information using the four underlying packages. It can run as a Java applet with a web browser or as standalone application. The packages are briefly described as follows:

**GUIModules** The **GUIModules** package provides classes that generate the graphical user interfaces for designer input and interaction. The Java 2 Swing library is used extensively in this package.
glprimitives The glprimitives package provides the 3D graphics engine for the main program. It uses GL4Java, a JNI encapsulation of the OpenGL graphics library, to generate and display mechanism models in real-time. This allows the applet to access the OpenGL DLL’s on the client machine. GL4Java is available for both Macintosh and Windows operating systems and can be downloaded for free.

kinemath The kinemath package provides basic mathematical resources used by other parts of the program. It allows programmers to conveniently access mathematics operations such as vector and matrix manipulation, and kinematics operations such as motion interpolation. Thejavax.vecmath package is used extensively; this package is distributed with Java3D, but is independent of the Java3D specification and implementation.

mechanism The mechanism package provides the base classes and interface specifications for defining linkages. It is designed to allow a programmer to easily implement new specialized linkage analysis and synthesis routines for integration with the Synthetica design environment. See Figure 4 for the components of the mechanism package.

The GUIModules and mechanism packages encapsulate the primary functionality of Synthetica by defining the user interface modules and the linkage routines respectively. We now provide a more detailed discussion of these two packages.

4.1 GUIModules package
The GUIModules package creates the windows and panels that allow users to modify/view the design data. The three key components are the Design Matrix, the Task Specifier, and the Solution Browser. The functionality of these modules is dependent on the data structures and class definitions found in the mechanism package, and on the standard Java 2 packages.

To accommodate the incorporation of multiple linkage and synthesis routines, we have taken advantage of Java’s built-in capabilities to dynamically inspect class contents at
runtime. This feature is represented in Figure 2 by the Class Loader. When the Synthetica applet loads, the available mechanism and synthesis classes are inspected to determine the mechanism topology, and task characteristics. This information is saved in a map and presented to the user in the Design Matrix. The ability to load and view the contents of an arbitrary class allows us include classes generated by other programmers. In particular, the java.net.URLClassLoader class allows us to load any user defined class given its url. The class specifications defined in the mechanism package ensures that the classes contain all the information we need to integrate them into our list of mechanisms and synthesis routines.

A task is defined by its number of positions and additional constraints. This information is used to dynamically generate the Task Specifier window which directs the user to enter all the information required by the particular synthesis routine. Position and Constraint defaults are provided along with a listing of constraint names.

A mechanism is defined by a set of serial chains composed of links and joints. This information is entirely dependent on the linkage topology. The Solution Browser window is dynamically generated to present the key information about each serial chain that makes up the mechanism. Numerical information is presented using standard editable text fields. An information panel is generated for each serial chain, and presented together in the Solution Browser. This allows us to display all the chains associated with a parallel linkage topology in a single window.

4.2 mechanism Package

In order to support multiple mechanism topologies and synthesis routines, a detailed package has been developed for programmers who wish to extend Synthetica. The mechanism package provides the classes which define the data structure of spatial linkages as well as a number of key interfaces for implementing special purpose kinematics and synthesis routines. The package is designed using the subclassing architecture of the Java language specification.

The mechanism package hierarchy shown in Figure 4. This subclassing hierarchy follows the “is-a” rule of object-oriented programming (OOP). For example, JointR which defines a revolute joint is a case of a general Joint. It’s natural to extend JointR from Joint. Also since SerialMechanism is a case of Mechanism, we extend SerialMechanism from Mechanism. Note that the lines with arrows represent the inheritance relationship and the data in the parentheses represent the key fields in the class. For example, a SerialChain object is a primitive that has an array of joints and an array of links. This relationship is called “has-a” relationship in OOP. The DesignTask class is a data class that contains the positions and constraints required by a synthesis routine.

We have also defined four Java interfaces. The ForwardKinematics and InverseKinematics interfaces specify the methods required for linkage position analysis. The Synthesizable interface specifies a set of methods for defining default tasks, constraint names, and for performing finite position synthesis. The Drawable interface provides the programmer with the ability to create customized geometry for the mechanism joints and links.
5 Implementing Mechanism and Synthesis Classes

Basically, Synthetica serves two roles. It can run as a standalone application/applet which causal users can use it to synthesize and analyze implemented mechanisms. It is also a programming package. Interested users can define their own mechanisms and implement their own synthesis or kinematics algorithms. At runtime, the program examines the contents of these classes to determine the mechanism topology and DesignTask information needed to construct the Design Matrix.

All mechanism classes and synthesis routines are implemented in the same way using the following basic procedure:

1. Create a new class by extending a base mechanism class, for example, SerialMechanism.

2. Implement any interface methods you require, such as Synthesizable and InverseKinematics.

3. Compile the source together with the Synthetica API, to obtain a java .class file.

4. Run Synthetica, and load the new class file using the Class Loader.

5. The information found in the new class will be mapped into the Design Matrix and made available for use.

For the example design discussed in the next section, we implemented five classes. The first two classes define the RR and TS linkage primitives. The next three classes define specific synthesis routines we were interested in implementing. In general, we have found
it convenient to implement one class per mechanism topology, and one class per synthesis
to implement one class per mechanism topology, and one class per synthesis
routine. In this case the five classes are defined as follows:

```java
class RRmechanism extends SerialMechanism implements InverseKinematics {
    /* define mechanism topology and associated data */
    /* override default inverse kinematics function */
    ...
}

class TSmechanism extends SerialMechanism implements InverseKinematics {
    /* define mechanism topology and associated data */
    /* override default inverse kinematics function */
    ...
}

RRsynthesis1 extends RRmechanism implements Synthesizable {
    /* define default task, default constraints, an constraint names */
    /* implement 3 position synthesis method */
    ...
}

TSsynthesis1 extends TSmechanism implements Synthesizable {
    /* define default task, default constraints, an constraint names */
    /* implement 3 position synthesis method */
    ...
}

TSsynthesis2 extends TSmechanism implements Synthesizable {
    /* define default task, default constraints, an constraint names */
    /* implement 3 position synthesis method with different constraints */
    ...
}
```

When the Class Loader encounters these classes it can inspect them to find that the
first class defines and RR mechanism, the second class defines a TS mechanism, the third
class contains a synthesis method for 3 position RR synthesis, the fourth class contains a
synthesis method for 3 position TS synthesis, and the fifth class contains a method for 3
position synthesis, with a different constraint list than the fourth.
6 Example Design Session: RRTS Linkage Synthesis

In this section we provide an example design session for specifying and synthesizing an RRTS linkage. Since we have already defined our mechanism and synthesis classes, we can run the Synthetica applet and have it automatically inspect the classes. The resulting Design Matrix is shown in Figure 5. The top portion of the Design Matrix window is a table of buttons. The columns are labeled with the available linkage primitives, and the rows are labeled with the number of design positions. Within the table, there are buttons used to select a particular synthesis method. The number that appears on the button indicates how many synthesis methods are available for that particular combinations of positions and linkage primitive. In this case, we have one 3 position synthesis method for the RR chain, and two for the TS chain. When we press the button, information associated with the available synthesis routines is displayed below the table. This allows the designer to choose between different synthesis methods. We have also provided an additional button for manual specification of the linkage dimensions.

In this example, we wish to proceed to synthesize an RR chain. Pressing the Synthesize button leads to the generation of the Task Specifier, shown in Figure 6. The Task Specifier window allows us to enter the design positions along with any additional constraints required by the synthesis method. It is customized for the particular linkage/synthesis routine. For the RR chain, it will only allow the user to specify three positions. When we design a TS chain, the names of the additional required constraints are provided along with the position information.

Once the design task is completely defined, the program executes the synthesis method and displays the solutions in the Solution Browser, shown in Figure 7. At this point we have a number of options. We can change the entire design selection, we can revise the task, we can edit the linkage parameters, we can add an additional chain to the design, or we can generate a 3D model of the linkage solution.

In this case we want to add a second chain to the design. Pressing the Add button brings us back to the Design Matrix from which we select a TS chain synthesis method. We proceed through the Task Specifier and generate a set of TS linkage solutions. Now, the Solution Browser contains a panel for each set of chains. On the left we have the solutions for the RR linkage, and on the right we find the solutions for the TS linkage. We can now view each solution individually, or combine the serial chains into a parallel linkage. In either case, the program takes the linkage data associated with the selected solution and displays an interactive 3D model of the design as shown in Figure 8.

Depending on what kinematics routines are available, the designer can then interactively move the linkage through the various positions, or change the joint variables to animate the mechanism. It is important to note that any serial chains designed for the same positions can be assembled at those positions. This does not guarantee that the mechanism can move in a desirable way, thus motivating the need for post-synthesis evaluation.
Figure 5: Sample Design Matrix

Figure 6: Sample Task Specifier
Figure 7: Sample Solution Browser

Figure 8: Sample Mechanism Viewer
7 Discussion

The current version of Synthetica is an applet that runs in a web browser or an applet viewer. The applet has been tested under Windows 95/98/NT, Mac OS 9, and Mac OS X. True cross platform compatibility is complicated by the fact that different web browsers accommodate different Java Virtual Machines, and the OpenGL links to Java are platform dependent. However, with once the proper libraries are installed, the same java code can be compiled to run on any of the supported operating systems.

Once the java applet loads, all computations are performed on the client machine. The applet is less than 100K in size, and loads and initializes in a few seconds. The majority of the applet functionality comes from the Java 2 packages and OpenGL libraries which reside on the client machine. The synthesis routines for the RR and TS mechanisms are not particularly complex, and execute instantly. The 3D graphics are initialized and generated directly within the applet itself and passed to the resident OpenGL implementation which is typically hardware accelerated. The RRTS linkage can be colored, textured, manipulated, and viewed in real-time.

The extensible architecture of Synthetica serves three purposes. First, it gives the designer access to multiple implemented linkage topologies and multiple synthesis methods. Second, it allows interested users to implement a documented synthesis or analysis routine, thus making it available to the design community. Third, it aides researchers, who are developing and testing new synthesis and analysis routines for new linkage architectures, by providing them access to the common GUI and graphics functionality required for linkage design.

The Synthetica applet prototype and package architecture is a solid first step towards satisfying the objectives set out in section 1.2. In particular, the software satisfies 7 out of 10 objectives. Future work will focus on addressing objectives 6, 8 and 9. In particular, we plan to develop multiple evaluation interfaces for classifying and ranking solutions, and interfaces for useful tools such as force, tolerance, and interference analysis. CAD integration is especially important and efforts are underway to develop a simple model interface that can be used to develop translators to common CAD file formats.

8 Conclusions

This paper presents a software architecture for the functional design of linkages consisting of four primary modules: a Design Matrix, a Task specifier, a Solution browser, and a Mechanism viewer. These modules are supported by four packages. Mechanism definitions and synthesis routines are implemented using the specifications found in the mechanism package. After these routines are written and compiled, their properties are automatically identified and integrated into the Design Matrix. The system supports open chain primitives and platform topologies and allows the designer to synthesize any number of different open chains for the same fundamental positions. This approach allows the user to investigate a large number of spatial mechanism topologies and extend the system with new mechanism definitions and synthesis routines. The first prototype system defines spatial RR and TS open chains and closed chain topologies that can be constructed from them. We show that OpenGL, GL4Java, and Java 2 combine to provided a convenient cross-platform
development environment. Future research will seek to involve multiple design laboratories in a collaborative development effort for the computer-aided design of spatial linkages and robotic systems.

**Acknowledgement**

The author gratefully acknowledge the support of the National Science Foundation and the University of California, Riverside.

**References**


Software


