3

The Systems Integration Architecture: an Agile Information Infrastructure

by John J. Mills, Ramez Elmasri, Kashif Khan, Srinivas Miriyala, and Kalpana Subramanian
The Agile Aerospace Manufacturing Research Center,
The Automation & Robotics Research Institute,
The University of Texas at Arlington,
7300 Jack Newell Blvd. S,
Fort Worth TX, 76118, U.S.A.
Principle Contact: Dr. John J. Mills, 010-817-794-5903; 010-817-794-5952 [fax]; e-mail: jmills@arri.uta.edu

Abstract

The Systems Integration Architecture project is taking a fresh approach to integration with special emphasis on supporting Agile, Virtual enterprises and their unique need for reconfigurable, de-centralized information systems. SIA is based on a different conceptual model of integration - called the TAR model - one which allows higher level integration than the traditional common, neutral format, shared database. The basic concept is that all information processing consists of a series of transformations of data sets called "Aspects" into other "Aspects". The transformations are effected by what are called Functional Transformation Agents which provide a single function and have a well defined interface structure which allows them to be integrated in a variety of ways. The Systems Integration Architecture provides three high level services which allow FTA's to be defined, combined into diverse networks which provide transformations of aspects in a manner transparent to the user and executed under a variety of control algorithms in a heterogeneous, de-centralized environment. These are: an Executive--to provide business control services; a Librarian--to act as somewhat of a meta data dictionary, keeping track of FTA's and their input and output Aspects; and a Network--which is based on the Common Object Request Broker Architecture to facilitate communication between objects on a heterogeneous network. An extended entity relationship diagram is provided and each of the objects in the system is described. A simple example of its use is provided.

A. Sölvberg et al. (eds.), *Information Systems Development for Decentralized Organizations*

INTRODUCTION AND BACKGROUND

Agile Manufacturing is a new paradigm in which manufacturers must be able to rapidly respond to unanticipated market demands, producing new products uwith resources outside of their immediate company in what has become known as "Virtual Corporations". In these, complementary core competencies from diverse companies including suppliers and customers, are combined temporarily into a new, decentralized company existing only to fulfill that market demand and dissolving when the demand is satisfied.

To support such rapid formation and reconfiguration of business processes, a unique information infrastructure will be required. The requirements of such an infrastructure are not clear at present, but some things are fairly clear. A business consists of a series of processes (order entry, generate purchase orders and invoices, design part or product, produce artifact, etc.). An individual business process consists of a series of functions or activities in IDEF0 formalism [1] which takes some input and "transforms" or converts it into an output. Both the processes and the lower level functions perform some action. The processes perform the action to achieve a business goal or objective, subject to a set of business rules.

Even if a company is not ready to become "Agile", to survive in today's competitive world, it must be continually seeking to improve itself. Such continuous improvement requires change; changes within individual processes and changes within the organization of those processes and activities.

There is a strong move in industry to treat these business processes like any other industrial process and to apply engineering principles to their design and implementation. This move is called Business Process Re-Engineering. If we are to be able to reconfigure the business functions or processes at any level, the information system supporting these must be modular in nature with clearly defined inputs and outputs. These modules must be supported by an information infrastructure into which the modules "plug" in such a manner that they provide integration at multiple levels. At this juncture, it is not necessary to define the granularity of the modules.

STEP is an international effort to standardize the exchange of data within an enterprise. While this effort will go a long way to addressing the issue of interfaces, we believe that it is based on a conceptual model which is no longer adequate for integration within an agile or continuously improving environment.

A conceptual or basis model is a simple model which is used to communicate the ideal which one is attempting to achieve when building some system. Another way of viewing it is that it is an abstraction of the purpose of the system.[2] Most implementations of integrated systems are based on a model consisting of a common, neutral format, logically unified, but physically distributed, shared database. This model has its problems as we have discussed elsewhere[3]. It requires mechanisms for "Unification" of the various views extracted from it [4]. It does not support time dependency of the data (being a static model), it is data centric, it ignores the processes which create the data in the first place and it does not provide an easy migration path to its eventual creation. The CIM-OSA project has defined at least three levels of integration as illustrated in Figure 1[5]. The common database model only supports one aspect of this concept.

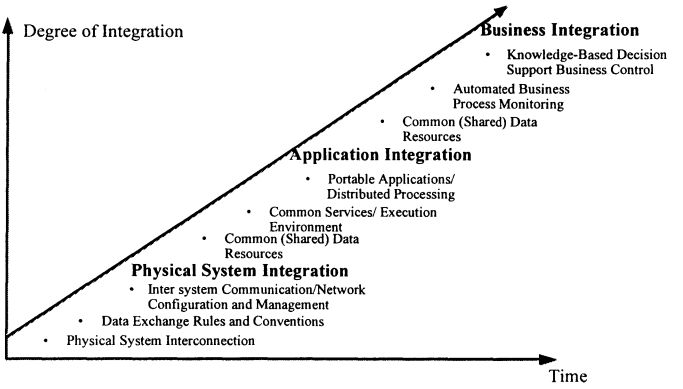

Figure 1. CIM-OSA Levels of Integration

The common shared database model is basically sound and should be incorporated in any model that replaces it. Elsewhere we are proposing a basis model derived from one emerging from the design theory world. We call it the Transformation of Aspects and Representations or TAR model and briefly describe it below. It supports all of the concepts in the CIM-OSA model.

The Automation & Robotics Institute (ARRI) is creating the *Systems Integration Architecture* that facilitates seamless integration of tools not necessarily designed to be used together. The *Systems Integration Architecture (SIA)*, based on the TAR model, was designed to integrate heterogeneous functional modules and systems with a minimum of effort, thus allowing the definition and execution of processes that use multiple application modules in an agile fashion. SIA also facilitates agility and flexibility in the integration of heterogeneous tools using an object oriented approach. The goal of SIA is to support the rapid reconfiguration of software modules into an information system supporting the enterprise as it is reconfigured. It can support integration, from the lowest level processes to the highest level processes in an enterprise, with any degree of business control desired.

Below we briefly describe the TAR model followed by an overview of SIA, with some further detail. We end with a short example of its use.

TAR MODEL

The derivation of this model can be traced back at least two decades in the design theory arena [6,7,8,9,10]. In the latest instantiation, Fenves and Talukdar express it as the TAO model. We have expanded it out of the design theory world to encompass the whole product realization process from needs definition to product delivery and support -- and from order entry to shipping.

The model is quite simple in essence. The product realization process (PRP) is one or more transformations of information and material. The artist, or craftsman, takes a set of needs or

requirements from inside themselves or from a customer and transforms these into the end product by a creative process with very few intermediate steps. In the engineering world, the realization of products is a process with many steps, each of which transforms a set of data or information into another data or information set, with the end result being a transformation of an information set and raw material into the finished product and supporting information.

For convenience we define two kinds of information sets which are transformed. The first, called "Representations" consist of all the information required at the completion of some phase or stage of the product realization process. The TAR model leaves the precise definition of these phases and their Representations up to the user. The PRP then consists of the transformations of the various Representations, starting with the needs and ending with the product. However, in reality, a "Representation" is not transformed in a single event, but in multiple parts and multiple steps, some of which are in series and some in parallel. We define a Representation, therefore, to consist of a set of "Aspects" which are subsets of Representations required by some function during the PRP and it is these "Aspects" which are transformed. Details of the model can be found in Reference 3

Transformations are effected by software applications which we call Functional Transformation Agents or FTA's. Fully manual transformation agents are humans performing some function for which an algorithm has not yet been defined. Computer aided transformations agents are applications which help a user either document the result of their creative activity or analyze, under human control, some data set or aspect. Fully automated transformation agents are applications which perform the transformations on their own. An example of a fully automated transformation agent is a rule based design system which transforms a set of specifications into a full design according to the rules embedded in it. A key feature of our model is the nature and structure of the interfaces between the transformation agents. Figure 2 illustrates the concept of the transformation agent and its interfaces.

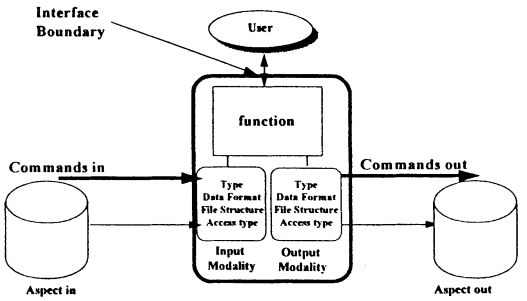

Figure 2. The Concept of a Transformation Agent and Its Interfaces

Systems engineering precepts imply a single function for each transformation agent. This has advantages, but most legacy software is not set up in that manner and they can provide many functions, each of which is an FTA. Transformations can also be combined into processes in any kind of network; each process carrying out a higher level function. Functional Transformation Agents transform an input Aspect into an output Aspect. To combine them, however, the interfaces between the agents becomes a critical part of the process design. If

the output Aspect(s) of one transformation agent does not match the input Aspect(s) of the next agent, the process cannot be executed. However, just specifying the type of data (i.e. the Aspect) does not guarantee that the functions can be integrated. The Aspects can be in many different file structures and data formats. They can also be accessed in many different ways. Accordingly, we have defined each Aspect to have, what we have termed, a "Modality' with three attributes: the file structure, the data format and the access type. These are described in more detail later.

Commands are also part of the interface definition, since, for true integration, it may be necessary for one FTA to activate another.

The TAR model is suitable for an Agile Enterprise because each transformation agent forms a module which can be configured into a system with other modules as long as the appropriate Aspects and Modalities can be matched.

THE SYSTEM INTEGRATION ARCHITECTURE

Overview

SIA achieves the required integration as specified by the CIM-OSA model, by providing certain key services in data management, control and communications. In SIA three modules provide these three categories of services; the Executive, the Librarian, and the Communications Kernel. We discuss these concepts in more detail later in the paper.

The approach used in the development of SIA is different from what is used in some of the existing integration frameworks. *SIFRAME,* developed by SIEMENS/NIXDORF, for instance, is a domain-less framework that uses a work space concept to support teamwork.[11] However, concurrent development is hampered because the user keeps the rights to a particular data set or Aspect until they free it. In our view, an integration system needs to provide for a variety of data sharing modes, not just lockout. The framework also assumes a logically unified database which has inherent problems as discussed earlier.

Khoros and *PowerFrame* are two other applications that facilitate integration at workflow level, but these do not integrate the services that are required with the data. Product data management systems, such as SDRC's DCMS or the new Metaphase product data manager being created by a joint venture between SDRC and CDC, focuses on facilitating product data management. It facilitates the integration of the data and provides certain other services which activate the appropriate application associated with a set of data, but does not provide for work flow management.

The Systems Integration Architecture on the other hand allows true agile information processing by allowing integration at all three CIM-OSA levels: data, service, and business. In SIA, functions are integrated along with the data needed by the functions in a manner that is transparent to the user. The user can call up any function supporting the activity or process they need and the system provides the data set in correct file structure, format, and access type without the user having to worry about the location or format of the tool.

Like SIFRAME, PowerFrame and Khoros, SIA uses the concept of processes to allow the user to define a workflow by checking for the appropriate format for tool integration through a user interface called *Executive*. However, SIA also keeps track of the information interfaces through a data management system called *Librarian* via a *Communication* kernel which is based on the Common Object Request Broker Architecture. All these modules are described in the next section. Figure 3 illustrates the conceptual System Integration Architecture.

Executive Module

The Executive module operates at the front end of SIA. It interacts only with the user and the Communications kernel module of SIA which in turn interacts with the Librarian module to provide the Executive with the required information. The primary objective of the Executive is to provide the user interface, which controls the user access to system data and resources, and to interpret user requests. The core idea of a minimal executive is, by invoking a specific control algorithm, it allows any degree of business control from a tight workflow management approach like SIFRAME, defining who does what and when, all the way to a fully autonomous system with autonomous agents reacting to a change in their environment. Hence, the executive acts as a client in SIA, with the Librarian and tools acting in the role of servers that provide specific services. The user interaction with SIA depends on the category of the user (i.e. project member or the project manager). It allows the team members to login and logout, browse through information about SIA objects, select a context, and launch enterprise processes. It helps the project manager to manage all the team member services along with storing and editing information about new or existing SIA objects, creating and editing enterprise process templates, and adding and deleting SIA objects. This feature is elaborated in the section below on Modes of Use.

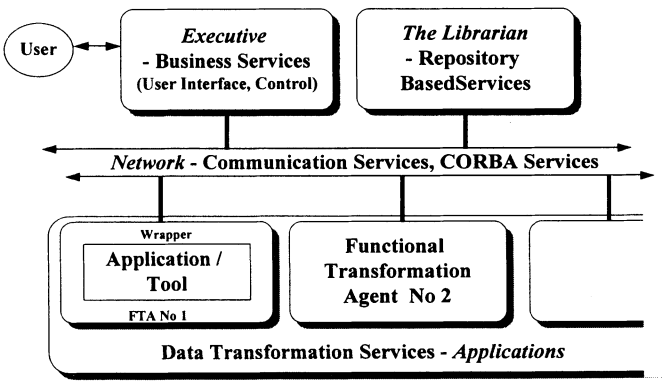

Figure 3. The Conceptual Model of the Systems Integration Architecture

The core Executive also allows the project manager to define transformation agents and their interfaces. Several other transformation modules are under development to provide other

required services to the users, including a menu driven interface to facilitate setting a context for passing of these commands to specific transformation agents and a project management function to define the flow of transformations and information management.

The Executive can start, stop, and pause/resume processes as well as request status information. It may activate certain specific processes which provide more functionality to the user, such as a system configuration function to allow the project's leader to configure the system with the degree of control he desires. It may activate a process to provide very specific sequencing of functional transformations for all users, or may allow the users to select the functions they want in whatever order they prefer, subject to the availability of data.

The Executive also has naming algorithms which the Librarian uses to name the data files that are generated by the transformation agents. The names must be compatible with conventions used by tools for naming their files.

The Librarian

The Librarian is the central object in SIA and its basic function is to manage the relationships among processes, tools, and data sets, and to maintain information about the existence and format of all data in the system. It acts as an information base in the architecture. The Librarian does not actually store items, it stores information about where and how the applications and data items are stored (i.e. meta data). It integrates both the data sets and the functionality.

The Librarian invokes modules for dealing with concurrency and sharing of data. It also responds to requests for access to data items as they are created. The Librarian maintains a list of all current objects (tools, transformation agents etc.) available in the system and enables the user to build new objects as needed. In addition, the Librarian maintains a history of all transformations performed on data items, as well as individual logs and versions for existing data files. The Librarian maintains knowledge about the capabilities and functions of tools, such as the transformations that the tools perform. It will allow the executive to return the user to the same place in his project on login. It also maintains records of all projects that are active, including the history of how, who, and when the data was created and where it is stored.

The Librarian also helps compute efficient transformation processes for data to be transformed from one format to another through the use of other modules. The Librarian stores these information paths for future reference. The Librarian manages all the configuration tables (that store information about the users, their access rights, and the status of each project) and addresses issues such as, how much information can one project access about any other project.

Communication Module

The main function of this module is to provide communications services among distributed heterogeneous modules. It is, essentially, the lower six layers of the ISO/OSI communication protocol architecture plus a message passing protocol. While we initially created our own network kernel, we are currently re-implementing it using the Common Object Request Broker Architecture (CORBA)[12].

CORBA allows us to define SIA services provided by the tools, functional transformation agents, and Librarian. These services are implemented as SIA services. The SIA client, the Executive can now communicate with and invoke these services through CORBA anywhere on the network. This will allow us the advantages of an object-oriented framework with the obvious benefits of inheritance, encapsulation of services, etc., thereby allowing heterogeneous platforms to communicate.

COMPONENT OBJECTS OF SIA

The Extended Entity Relationship[13] diagram, Figure 4, depicts all the component objects of SIA. A description of each follows.

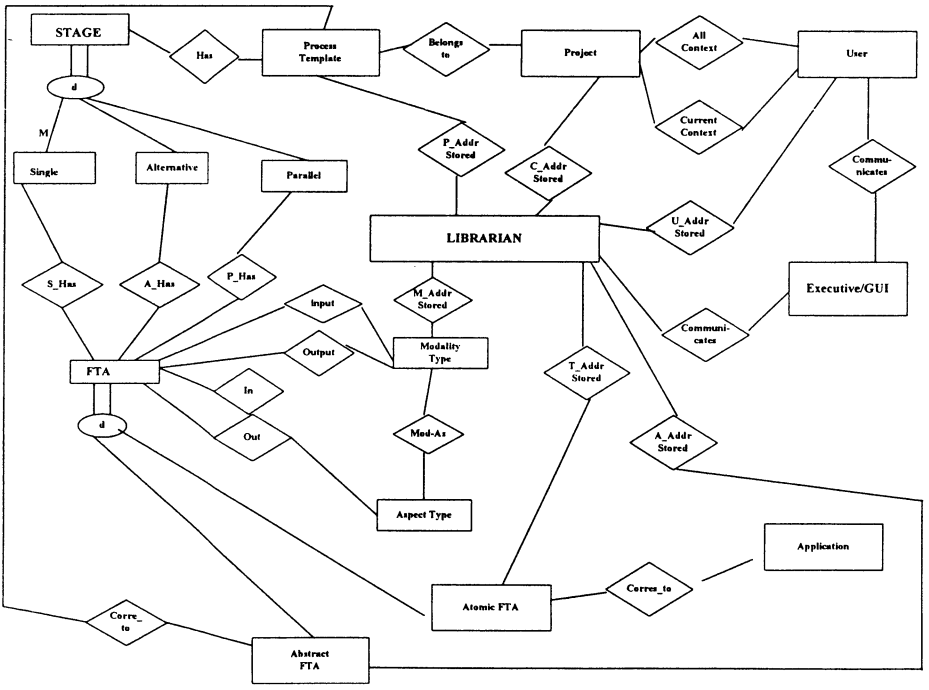

Figure 4. Extended Entity Relationship Diagram for the Systems Integration Architecture

Users, Contexts and Projects

Users in the architecture represent the current users of the system. They are divided into two subgroups- ordinary users and project leaders. Project objects are the primary interfaces for users. Projects maintain a user list with information about users passwords and data access rights. One user in the list is designated as the Project Leader. Each Project maintains a list of Process objects and their possible interdependencies.

The User is capable of launching and shutting down processes. Also, each Project object maintains a project log, detailing users' connect times and the processes invoked during user sessions. Project objects, or their users, may have priority levels assigned to them, in which case the Project with the higher Priority would be granted access to the FTA. The notion of context is related to the project and the skills of the user.

Aspects, Modality, and Functional Transformation Agents

Aspects in the Librarian module are represented as a combination of identifier and modality pair.

A ***data identifier*** is simply a unique name of an "Aspect" which is simply a specific set of information, such as a product part or a document. We term these "Aspects" since they can be considered as an "Aspect" or "View" of the product data. Each data identifier may have several modalities associated with it, representing different versions or formats of the information. In such a case, each modality represents either the same data in a different format, or another version of the information. The data item description contains a unique identifier associated with the data item and a time stamp used for version management. The combination of data items and description identifies a unique collection of data that can be stored as a file, or in a database, or in any other suitable data format.

Modality is a triplet that is a specification of physical format of data (including storage and data format).

Modality = { data_format, file_structure, access_type}

where *data_format = {DXF, IGES, EXCEL, WORD, STEP, AUTOCAD native format }*

file_structure = { ASCII, flat, binary, MacBinary, database}

access_type = { read/write, dbQuery, pipes, SDAI, RPC, API}

The ***Functional Transformation Agent*** (FTA) is the function that provides mapping from an input Aspect/Modality combination to an output Aspect/Modality.

FTA: input Aspect/Modality -------> output Aspect/Modality

SIA has a graphical user interface that allows new FTA's to be defined as needed. The modalities of the FTA's can be used by the process object to determine the consistency of the input and output specifications on which the FTA's should operate. Once launched, an FTA object should be able to start, stop, and abort operation and report its status.

FTA is an abstract class which has two subclasses. ***Abstract_FTA*** is a process template, in its own right, with its Aspects and their Modalities. It represents a higher level abstract function that is defined as a process of lower level FTA's and allows a hierarchical design of a process template. An Abstract FTA can include other abstract FTA's, which are more detailed process templates, and provide a high level view of a complex manufacturing process included as a component of another process. This allows abstract FTA's to be recursively composed.

The Atomic_FTA is the combination of Function and its interfaces. If the function is a computer based tool or application, then the ***Atomic_FTA*** also includes the enclosing wrapper of that function of the application. The wrapper in SIA encapsulates a function because each function requires a different aspect and, usually, a different modality. The ***Atomic_FTA*** is an FTA that can be directly executed using one or more available tools.

As each Atomic FTA is launched, the process will query the Librarian for the input and output modalities. These will be combined with the data identifiers listed in the transformation path and sent to a tool that performs the desired FTA function. If at some point a requested modality of an Aspect intended for input does not exist, the Process will request the Librarian to generate an extension to the transformation path. The Librarian will look up the existing modalities for the specified Aspect, and consult its list of FTA's, trying to locate one or more that can accomplish transforming one of the existing modalities into the requested one. If the Librarian succeeds in generating a sub-path that will produce the requested modality, the path is returned to the process, which incorporates it into its existing transformation path for future use.

A Process is the most autonomous action that a user can request. It represents a production, design or business process transformation path with a sequence of stages to be invoked. Processes are thus composed of FTA objects arranged in a transformation path, which represents a series of FTA's and the data items that are input and output from them.

A Process Template is a description of a process. It describes a process transformation path for the sequence of Functional Transformation Agents (FTA) to be invoked. Each process template can be used by several users to launch the corresponding process and each launched process is associated with only one process template. The process template consists of several stages, and each stage is either single, parallel, or alternative. A ***Single Stage*** consists of only one FTA. A ***Parallel Stage*** consists of several transformation agents, all of which must be invoked in parallel. An ***Alternative Stage*** consists of several FTAs, only one of which must be invoked. Figure 5 illustrates this concept.

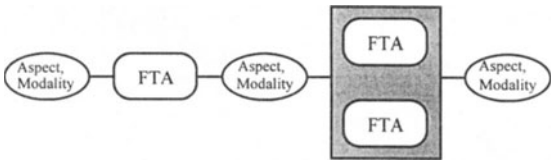

Figure 5. Concept of Stages in a Process Graph

MODES OF USE OF THE SYSTEMS INTEGRATION ARCHITECTURE

Modes of Systems Integration Architecture

To obtain its goals, the Architecture operates in three modes. *Application Integration mode* is used for defining a new tool or application and its location and launching mechanism. It assists in creating wrappers when necessary. It also allows the systems engineer to define the functionalities for a tool. *Process Definition Mode* allows SIA to check the compatibility of FTA's input/output while defining a new process. Transformation of FTA's can be used to make the input and output modalities compatible. *Process Launching Mode* helps keep track of current progress in a launched process, location of I/O files, etc.

Application integration mode

Application Integration deals with the principles and mechanisms that help us make the functions provided by the software tools of the environment cooperate efficiently towards the smooth flow of the transformation of information. SIA supports the integration of in-house and commercial tools. When a tool/platform is incompatible with SIA, a wrapper is developed to provide the capability for connecting the tool to SIA. This allows SIA modules to communicate with the new tool in the SIA protocols. The wrappers will allow seamless data communications between tools or applications that are normally not compatible. The architecture allows many diverse computer based tools to be integrated and interact in such a manner that is transparent to the user. The architecture allows any software application to be pluggable, which means that any application can be removed from a system and a new application, providing the same functionality, can be inserted without having to do a major rewrite of the integration software.

Process definition mode

Process in SIA can be compared to a workflow - a long duration, multi-step activity with some data and control flow between the constituent steps of the workflow. The basic reason for workflow is to automate routine work. It provides a structure to accomplish a series of tasks in an automated fashion,[14] the same objective being desired in SIA through the use of processes.

In process definition mode, SIA allows the project managers to design processes that require routine automation. This is achieved through the use of defining a process through different stages by the project manager based on FTA's needed to perform that particular task. If the output Aspect and its Modality of the first tool matches with the input Aspect/Modality of the other, the next stage can be defined; otherwise, SIA will search for a transformation tool that converts the output of the stage to the modality compatible with the input of the next stage.

A mark of a good workflow system is flexibility - both in design and operation.[15] SIA provides this flexibility by allowing the user to pick any tool out of the list of tools that can

perform the same function. If the tool is not available, another tool, able perform the same function, can be specified.

Process launching mode

The process launching mode deals with the execution of an already defined process.

Each FTA in the path is executed in order. In the case of multiple tools/applications that can perform the same function, the user may select which function they prefer. As each FTA is launched, the Process will query the FTA for its input and output modalities. These will be combined with the data identifiers listed in the transformation path and sent to the Librarian. The Librarian will then return file names for each data item.

The Process will start the FTA, passing it the filenames to be used for input and output. Each reference to data may be flagged as persistent or transient data. In the case of transient data no query to the Librarian is made. Instead, the two FTA's on either side of the data are launched with an interposed communication channel, established between them. This facilitates real-time function interaction in the system. Also each reference to an FTA object may or may not be flagged 'must run'. If the FTA is not flagged "must run", the output file is compared to the input file's data to determine if the output data is out of date. FTA is launched only if it is out of date.

Finally, if at some point a requested piece of data intended for input does not exist, the Process will request the Librarian to generate an extension to the transformation path. The Librarian will look up the existing modalities for the specified data, and consult its list of FTA's trying to locate one or more that can accomplish transforming one of the existing modalities into the requested one. If the Librarian succeeds in generating a sub path that will produce the requested modality, that path is returned to the Process, which incorporates it into its existing transformation path for use.

IMPLEMENTATION EXAMPLE

Suppose a user wants to define a process for designing an assembly of a new product using a Create Geometry function; passing this design to an assembly planning function; converting the assembly plan to a robot assembly program; and finally executing the robot assembly program.

The user first logs into SIA through the Executive module. The user may browse through a list of FTA's providing Create Geometry services available through SIA and select the desired FTA for the first step. The FTA selected is placed as the first stage in a Process Graph, which represents the process template being created. There could be several FTA's based on different CAD tools placed in parallel to allow the eventual user a selection based on his expertise. Each commercial application providing this functionality is encapsulated in a Wrapper to allow the association of the Aspect/Modality Definition with the appropriate functionality. The user then selects the planning FTA required for the assembly planning step, and attempts to place it as the second stage in the Process Graph, Figure 6.

SIA checks whether the output Aspect and Modality of Stage 1 (Create Geometry FTA) is compatible with the input Aspect and Modality of Stage 2 (planning FTA); if they are not compatible, SIA may search for a FTA that converts the output of Stage 1 to an Aspect/Modality combination compatible with the input of Stage 2. If such a FTA is found , it is inserted as Stage 1a in the process - otherwise, Stage 1a may be designated as a manual stage in the process. User may then select a FTA to convert the output of the planning tool into a robot assembly program (Stage 3); alternatively, Stage 3 could be designated as a manual stage.

Stage 4 in the Process Template could be manual stage to check that the robot used for assembly is properly set up. Stage 5 is then specified as invoking the Assembly Robot FTA with the file that is output from Stage 3. No output is shown from Stage 5 since the major output is the assembled artifact. There could be a status report as an output if that was desired, however.

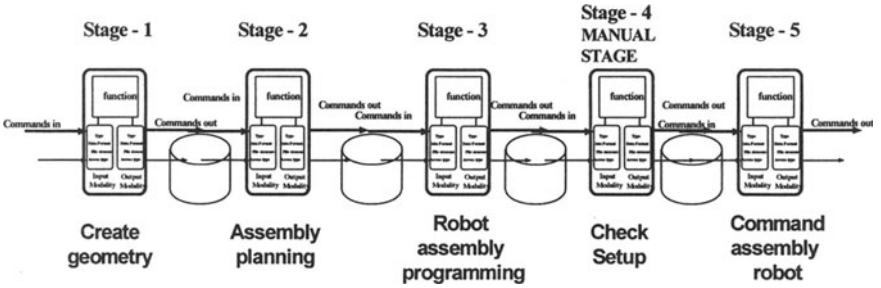

Figure 6. A Specific Process Graph for Integrating CAD to Robot Assembly

In the process definition scenario presented, we assume that the user wants to launch that process. A user logs in through the Executive module and selects the previously defined process template from the Librarian and launches the process. SIA identifies the first stage in the Process Template and starts the designated CAD tool, via its wrapper, to provide the required functionality of that tool. SIA keeps track of the stage of the user in the process. Once Stage 1 is completed, the output file from the CAD tool is registered with the Librarian.

SIA can now automatically invoke Stage 1a in the process (by starting the transformation tool with the appropriate input file). The output file, from the transformation FTA, is then provided as the input to Stage 2 in the process (invoking the assembly planning FTA).

The file created by Stage 2 is then input to Stage 3. If Stage 3 is a manual stage, SIA notifies the user that the function of Stage 3 must be performed manually and the process waits till the user provides the input file to Stage 4. SIA now informs the user to perform Stage 4 by checking that the robot is set up properly. After the user indicates that Stage 4 is completed, SIA can invoke the assembly robot (Stage 5 in the process) by providing the output file from Stage 3 (the robot assembly program).

CONCLUSIONS

The Systems Integration Architecture is based on a different model of integration than the traditional model of a shared, common database. The new model, called the TAR model, views all data processing as one or more transformations of data sets called Aspects. SIA integrates these data sets and the Functional Transformation Agents, which perform these transformations, by keeping track of the input and output Aspects and their structure called Modalities. SIA is currently being implemented in C++ on a variety of machines, including a Silicon Graphics Indigo, a Sun Sparcstation and a DEC ALPHA, all running with a UNIX operating system on a local area net within the Institute. When completed, SIA will, not only support business process re-engineering by facilitating the reconfiguration of the information systems to support the newly engineered business processes, but the concept of virtual companies as well. Specifically, the heterogeneous design of SIA, the use of standards and the definition of the standard module interface structure allows companies to manage the integration of their information systems with those of other companies to form the information system supporting the virtual company.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support of the National Science Foundation under Cooperative Agreement No DDM-9320949, April 15, 1994, the Agile Aerospace Manufacturing Research Center. The assistance of Pulin Chhatbar and Anup Bhatt in implementing the Executive and the Librarian and Neil d'Cunha in implementing the preliminary network kernel is also gratefully recognized.

AUTHOR'S BIOGRAPHIES

John J. Mills is the Director of The Agile Aerospace Manufacturing Research Center(AAMRC) and The Automation & Robotics Research Institute(ARRI). He also holds positions as a Professor, Mechanical Engineering at The University of Texas at Arlington and is the Fort Worth Chamber Foundation Chair in Automation and Robotics. He holds a Ph. D. in Applied Physics from The University of Durham, England and a B.Sc. Physics from The University of Glasgow, Scotland.

As the Director of the Automation & Robotics Research Institute, Dr. Mills manages a multi-million dollar institute dedicated to helping U.S. industry become globally competitive. In addition to his duties as director, Dr. Mills leads the Information Technologies Research Group, studying computer architectures and methods to support concurrent engineering approaches to the design, development and commissioning of complex manufacturing systems. He is also a Principle Investigator on the project to develop the Dynamically Reconfigurable Assembly System under Texas State funding. As Director of the Agile Aerospace Manufacturing Center, Dr. Mills heads one of three Agile Manufacturing Institutes in the

nation. Funded by an ARPA/NSF Agility Initiative, the Center is helping to define agility in aerospace manufacturing in the United States.

Ramez Elmasri is an Associate Professor in the Department of Computer Science and Engineering at The University of Texas at Arlington. He has worked in database research for over 15 years and holds M.S. and Ph.D. degrees in computer science from Stanford University. Prior to joining The University of Texas at Arlington in 1990, he was an Associate Professor at the University of Houston and a Principal Research Scientist at Honeywell. Elmasri is co-author, with S. Navathe, of the textbook "Fundamentals of Database Systems", Benjamin/Cummings, 1989 (Second Edition, 1994).

His work on a Tool Integration Architecture (TIA) for manufacturing systems is at the Automation & Robotics Research Institute (ARRI) at UT-Arlington, and focuses on developing an underlying architecture that supports agile manufacturing by quickly integrating new tools and processes. His work on software maintenance environments focuses on developing software maintenance process models and utilizing advanced database technologies in a maintenance environment.

Kashif Khan has an M.S in Computer Science and Engineering (1995) and MBA (1992) from The University of Texas at Arlington, and a B.S. in Electrical and Communications Engineering from the University of Engineering and Technology, Pakistan. He is currently working as a Research Assistant at the Automation & Robotics Research Institute, Fort Worth, Texas.

Srinivas Miryala has an M.S. (1995) in Mechanical Engineering from The University of Texas at Arlington. He is currently working for Erricson in Dallas, Texas.

Kalpana Subramanian is currently student of graduate program in Computer Science and Engineering at The University of Texas at Arlington. She has a B.S. in Electrical Engineering and is currently working as a Graduate Research Assistant at the Automation & Robotics Research Institute Fort Worth, Texas.

REFERENCES

[1] Mayer, R.J., "IDEF0 Function Modeling," Pub. by Knowledge Based Systems, College Station, TX, 1990.

[2] Mettala, E. Private Communication, September, 1994,

[3] Mills, J.J., "A Basis Model for Agile Computer Integrated Manufacturing," to be presented at ICCIM '95, Singapore, September, 1995.

[4] Fulton, J.A., PDES/STEP Model Unification: SUMM," Product Data International, May, pp. 6-7, 1992.

[5] Anon, "CIM-OSA Reference Architecture Specification,," ESPRIT Project Nr 688, pub. by CIM-OSA/AMICE, Brussels, Belgium, 1990.

[6] Balzer,, R., Goldman, N., and Wile, D "On the Transformational Implementation Approach to Programming," Proc. 2nd International Conference Software Engineering, pp. 337-343, 1976.

[7] Mostow, J., "Program Transformations for VLSI,' IJCAI-83, 1983.

[8] Ullrich, K., and Seering, W, Conceptual Design: Synthesis of Systems of Components," in Intelligent and Integrated Manufacturing Analysis and Synthesis, PED Vol. 25, ASME. NY, 1987, PP 57-66.

[9] Shah, J.J., and Wilson, P.R. "Analysis of Knowledge Abstraction, Representation and Interactions Requirements for Computer Aided Engineering," Computers in Engineering, Vol. 1, 1988, Pub. by ASME, New York, pp. 17-24.

[10] Talukdar, S.N., and Fenves, S.J., "Towards a Framework for Concurrent Design, in D. Sriram et. al, Computer-Aided Cooperative Product Development, MIT-JSME Workshop, November 1989, Lecture Notes in Computer Science No. 492, Springer-Verlag, 140-151.

[11] "SIFRAME V6.0 System Overview," Siemens Nixdorf, New York, 1992

[12] Betz, M., "Interoperable Objects," Dr Dobbs Journal, pp 18-39, October, 1994.

[13] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, Second Edition, Benjamin/Cummings, 1994

[14] Davis D.B. "Software that makes your workflow," Datamation, April 15, 1991.

[15] Dyson E. "Workflow," Forbes November 23, 1992.