Outline

- Manufacturing User Requirements
- Control Software Requirements
- Technical Elements
- Research Agenda
- Summary
“Lean” Manufacturing Objectives

- Reduce Costs
- Increase Responsiveness
- Improve Quality

Lean = Responsiveness + Cost + Quality

Responsiveness:
- System Flexibility
- Adoption of new Technologies

Cost:

Investment:
- Equipment Procurement
- Stability of Technology

Operating:
- Reliability & Maintainability
- Capability
- Usability

Responsiveness:
- Design & Engineering
- Manufacturing & Production
Cost Areas

- Focus on costs that can be impacted with engineering.
- Need a more efficient engineering & maintenance process.

- Commodity HW and shop labor
- Engineering-impacted costs
Engineering-impacted lifecycle cost profile

- Preparation
- Hardware design
- Software design
- Test, debug, integrate, startup training
- Operation & Maintenance
Engineering & Maintenance Needs

- Efficient Engineering
  - New applications
  - Retrofits
- Efficient Start-up
  - Easy, safe learning curve at start-up
  - Efficient, safe debugging
- Efficient, Safe Operation and Maintenance
Assuring Correctness of Programs

- Support to capture process specs (sequence of operations) correctly
- Validation of process specs, e.g.,
  - Check for inconsistencies
  - Support for simulation with manufacturing process in the loop
- Efficient transformation from process specs to executable programs
- Ability to debug on shop floor at high level program/spec
Reducing Design Time/Cost

- Support for efficient reuse of library components
- Support for efficient maintenance of reusable libraries
- Platform independence
- Export/import in vendor-neutral, tool-neutral form
Making Programs Easier to Understand

• Proper modularization, e.g., IEC 61499 function block types
• Reuse of software
  – Standard library elements
• Standards for module interconnection & interaction
• Consistent control structure that is not error-prone
• Eliminate need for changes in low level code
  – Change at high level only
Supporting Operation & Maintenance (1/2)

• Minimize time cost to locate cause of interruption in production
  – Eliminate need to browse “raw” control logic
  – Support operator in correcting manufacturing problems
    ▪ Reduce avoidable calls to skilled trades
  – Isolate condition not fulfilled
  – Indicate specific fault point
  – Provide online help & documentation
    ▪ Support adding helpful hints from experience
• Minimize engineering effort in providing troubleshooting help
  – Integrate diagnostics with control logic
  – Reuse & integrate data from multiple sources, e.g., IO devices database
• Integrate subsystem status & diagnostics info, e.g., servos
• Safe recovery procedures when out of sequence
**Existing Deficiencies**

- The lack of Open Modular Architecture Controls has been a significant impediment during the debug and ramp-up phase of a manufacturing system
  - Can not support proper reconfiguration of manufacturing systems
- The lack of portability and reusability application software imposes significant avoidable costs and delays in creating new manufacturing system configurations
- There is a lack of organized, systematized knowledge to specify, apply, integrate, use, test and evaluate an OMAC-based system correctly and efficiently
General Control Software Requirements

- Control logic software shall be modular, portable, and composed of reusable library components
- The definition of all language elements shall be publicly published, supplier-neutral, and independent of any graphical presentation form or language
- Application control logic shall be accessible in a view-independent form for portability
- Software organization shall be consistent with the model described in IEC 61499
Reuse - Standard Library Elements

**Automation domain Function Block types**
- Examples:
  - AxisSupervisor, Servo-axis of motion
  - 2PositionMotionDevice, IOdevice

**General Function Block (FB) types**
- Examples:
  - counter/timer
  - character-string

**Common elements**
- (ref IEC 61499)
  - Identifiers, literals, data types, variables
  - Function block type structure

Integral diagnostics
• In cases where no IEC standard is sufficiently defined
  – specified through or mappable into XML Schema, utilizing the most expressive elements
Reusable generic interfaces

FSM

GenericTask

GenericResource

Application-specific Task

IODevice

2PositionMotionDevice

AxisSupervisor

ServoAxis
Process Plan-to-Task Transformation

(Job scheduler)

ApplicationSpecific
ProcessPlan

ResourceBinder

Reusable resources & configuration

ApplicationSpecific
ResourceSpecific
Program

ProgramTranslator

ApplicationSpecific
Task (FSM)
with bindings to reusable resources
From Application Specific Task to Reusable Resources

Application Specific Control Logic Task

Locator
Clamp
X-Axis
Y-Axis

Reusable Resources

2PositionMotionDevice
AxisSupervisor
Control Logic Design Structure

Modularization of programming software and resources

Multiple notations ("languages") in use to specify, present or view a program

Design tools

Data access, Browsing, Navigation, Presentation, & Conversion Tools

Application domain oriented library building blocks, e.g., Axis of motion
Generic library building blocks, e.g., counter, character-string

General, application-independent system services (ref IEC 61499-1 CD, Sec 3-4, Annexes B.3, F-H)
  e.g., task configuration, inter-task communication, resource allocation, scheduling, timing

Common elements (ref IEC 61499-1 CD, Section 1.4,2, & Annexes), e.g., Identifiers, literals, data types, variables (Annex B.4, E), abstract data (function block) type structure (Annex B.2).

Basic operating system services & hardware abstractions

View-independent database of control logic programs and components

HMI signals, Peer Interlocks, IO

Denotes view-specific data

Multiple notations ("languages")

Textual - Graphic

IL | ST | XML | UML | SFC | LD | FBD | Flowcharts

\[\begin{align*}
\text{Denotes view-specific data}& \\
\text{IL, ST, XML, UML, SFC, LD, FBD, Flowcharts}&
\end{align*}\]
• Compare various architectural alternatives
  – Reduction in learning time
  – Reduction in debug time
  – Reduction in integration time
  – Reduction in machine control application development time
• Develop the metrics to compare various architectural alternatives
Software Engineering Research (2/2)

- Test methods & procedures to assure “plug and play” integration of software components for the machine control application domain
  - including conformance to timing requirements
Which FSM Model? 
- Controversy in FSM model - action on transition or in state?
Event management research issue (1/2)

- **Abnormality events**
  - An abnormality event can occur in any state, e.g., some emergency.
  - Should its transition be explicitly specified in every state?
  - Or, can there be a shortcut without affecting verifiability?
Event management research issue (2/2)

External event flow in hierarchical FSM: inside->out, outside->in, ...?

Control system
Task Coordinator

Application Specific Task (Control Program)

2PositionMotionDevice

I0device

Fault event
Summary

• OMAC User Requirements given in terms of impact on the overall engineering & maintenance process lifecycle.
• No new inventions / paradigms needed.
• R&D needed in software engineering to compare alternatives with respect to
  – Ease of learning
  – Reduction of debug, integration & startup time
  – Ease of maintenance
  – Reuse