Design Principles for Agile Production

Rick Dove, Sr. Fellow, Agility Forum, dove@well.com, Paradigm Shift International, 505-586-1536

This series of articles has been exploring the nature of Agility in production systems and occasionally the enterprise systems that encompass them; making the argument more than once that Agility is a characteristic which emerges from design. Behind each of these systems are "business engineers" responsible for the system's design - consciously or unconsciously as the case may be.

Good engineering is applied science. Some would argue about management as science, and others believe a manufacturing science remains elusive. Nevertheless, the design of manufacturing enterprise systems, from production process to business procedure, can result in a more or less adaptable system to the extent that certain design principles are employed. The expression of Agility design principles explored in three production systems (Sep, Oct, Nov '95) is assembled on the next page in tabular form showing various applications.

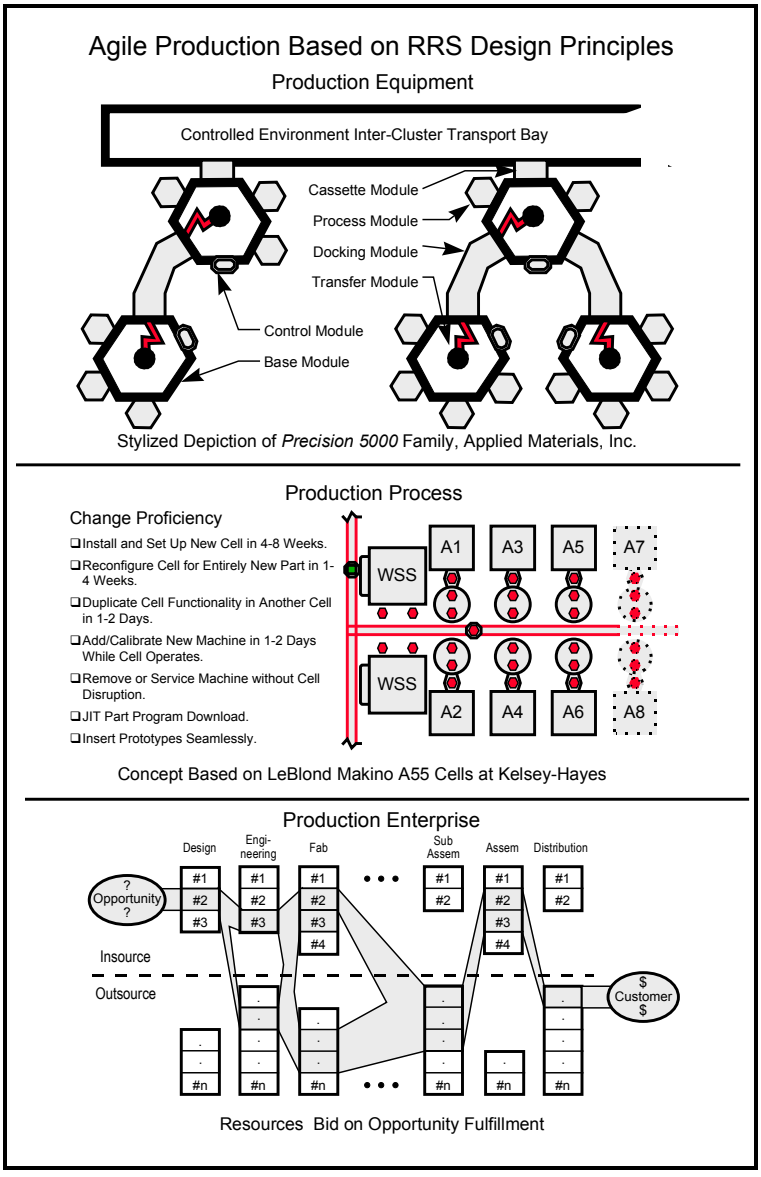
Instead of simply lurching to the next competitive operating state, Agile design principles facilitate continuous evolution.

Science is born from gathering data, analyzing this data for patterns, making hypothesis on principles, and iterating toward validation. We are not employing the rigors of scientific investigation yet; but we are finding repeatable patterns that appear to govern adaptability.

Few would disagree that information automation systems are critical enablers for modern production; but what will the information automation system do to support an Agile operating environment? Perhaps more importantly, what will make the system itself Agile so that it can continue to support an Agile operating environment rather than guarantee its obsolescence? Are there fundamental characteristics that provide Agileness that we can look for in selecting information automation systems?

Adaptability (Agility) actually became a reasoned focus with the advent of object-oriented software interests in the early '80s. The progress of software technology and deployment of large integrated software systems has provided an interesting laboratory for the study of complex interacting systems in all parts of enterprise. The integrated software system, whether it's in the accounting area, providing management decision support, or spread over countless factory computers and programmable logic controllers, is understood to be the creation of a team of programmers and system integrators. We recognize that these people also have the responsibility for ongoing maintenance and upgrade during the life of the system. In short, the integrated software system is the product of intentional design, constant improvement, and eventual replacement with the cycle repeating.

As engineering efforts, the design and implementation of these integrated software systems proceeds according to an "architecture", whether planned or defacto. Over the years the size and complexity of these systems grows to a point where traditional techniques are recognized as ineffective. This awareness comes from experience: from waiting in line for years to get necessary changes to the corporate accounting system; from living with the bugs in the production control system rather



than risk the uncertainty of a software change; and from watching budgets, schedules, and design specifications have little or no impact on the actual system integration effort.

The problem stems from dynamics. Traditional techniques approach software design and implementation as if a system will remain static and have a long and stable life. New techniques, based on "object oriented" architectures, recognize that systems must constantly change, that improvements and repairs must be made without risk, that portions of the system must take advantage of new sub-systems when their advantages become compelling, and that interactions among subsystems must be partitioned to eliminate side-effects.

These new approaches have been matured over a decade now and are emerging most visibly into everyday employment under the name client-server architecture. Though there are significant differences between systems concepts called client-server and those called object-oriented, “encapsulated” modularity and independent functionality are important and shared key concepts. More to the point, information automation practitioners are now focusing a good deal of thought on the architectures of systems that accommodate change; providing a rich laboratory and experience base from which fundamental Agility principles are beginning to emerge.

The ten “RRS” (Reusable, Reconfigurable, Scalable) design principles introduced earlier (Aug ‘95) and tabulated below are based on object-oriented concepts augmented with understandings from production and enterprise systems exhibiting high degrees of adaptability.

Readers far removed from systems engineering or computer technology may find the words used to describe these principles too abstract at first. A human resources director, for instance, might feel more comfortable with "empowered work team" then with "encapsulated modules", though the two are similar architectural concepts. But then, few people in business are taking a business engineering approach as yet.

The Agile RRS design principles identified here are presented as a useful working set that will undergo evolution and refinement with application. Their value is in their universal applicability across any system that would be adaptable. Instead of simply lurching to the next competitive state, Agile design principles facilitate continuous evolution.

Each of these principles will be subsequently examined in much greater depth, as will methods for establishing the objectives of an Agile system and the metrics of success. Though we have focused on the “solution” approach here, it is critical to establish an objective understanding of the opportunity and “problem” before embarking upon the solution.

Agile Reusable-Reconfigurable-Scalable (RRS) Design Principles

RRS	Design Principles	Production Equipment (Cluster Machines - Sep ‘95)	Production Process (Agile Machining Cell - Oct ‘95)	Production Enterprise (Enterprise Job Shop - Nov ‘95)
Reusable	Self Contained Units: System of separable, self-sufficient units not intimately integrated. Internal workings unimportant externally.	Wafer transfer module, various process modules, docking module, cassette transfer module, utility-base module.	Machines, work-setting stations, pallet changers, fixtures, rail-guided vehicles.	Design, engineering, fabrication, sub-assembly, assembly, and distribution resource modules.
Reusable	Plug Compatibility: System units share common interaction and interface standards and are easily inserted or removed.	Common human, mechanical, electrical, vacuum, and control system interfaces.	Common human, mechanical, electrical, and coolant system interfaces. Common inter-module mechanical interfaces.	Common info system and procedures among captured corporate resources, common interface in outsourcing contracts.
Reusable	Facilitated Re-Use: Unit inventory management, modification tools, and designated maintenance responsibilities.	Machine manufacturer extends/replicates module family for new capabilities. Fast module-swap maintenance is facilitated.	Machines do not require pits or special foundations, and are relatively light and easy to move.	Corporate outsourcing department maintains pre-qualified pool of potential outsources.
Reconfigurable	Non-Hierarchical Interaction: Non-hierarchical direct negotiation, communication, and interaction among system units.	Processing modules decide how to meet part production objectives with closed-loop controls.	Complete autonomous part machining, direct machine-repository download negotiation.	Business unit resources free to bid on internal jobsand external jobs.
Reconfigurable	Deferred Commitment: Relationships are transient when possible; fixed binding is postponed until immediately necessary.	Machine custom configured with processing modules at customer installation time.	Machines and material scheduled in real-time, downloaded part programs serve individual work requirements.	Individual business unit assigned to opportunity fulfillment at last possible moment.
Reconfigurable	Distributed Control & Information: Units respond to objectives; decisions made at point of knowledge; data retained locally but accessible globally.	Intelligent process modules keep personal usage histories and evolving process characterization curves.	Part programs downloaded to machines, machine history kept in machine controller, machines ask for work when ready.	Enterprise integration information system queries data bases local to the busines unit.
Reconfigurable	Self Organizing Relationships: Dynamic unit alliances and scheduling; open bidding; and other self-adapting behaviors.	Real-time control system makes use of processing units available at any given time, scheduling and re-routing as needed.	Cell control software dynamically changes work routing for status changes and new or removed machines on the fly.	Bid-based production-flow alliances.
Scalable	Flexible Capacity: Unrestricted unit populations that allow large increases and decreases in total unit population.	Machines can be interconnected into larger constant-vacuum macro-clusters.	Cell can accommodate any number of machines and up to four work-setting stations.	Outsourced resources can be easily added or deleted to increase the population of production modules with no size restrictions.
Scalable	Unit Redundancy: Duplicate unit types or capabilities to provide capacity fluctuation options and fault tolerance.	Machine utility bases are all identical, duplicate processing chambers can be mounted on same base or different bases.	Cells have multiples of each module, all cells made from same types of modules, machines have full work functionality.	Multiple duplicate production resources and second-outsources.
Scalable	Evolving Standards: Evolving, open system framework capable of accommodating legacy, common, and completely new units.	Base framework becoming standard across vendors, and has accommodated processing technology across generations.	Utility services and vehicle tracks can be extended without restrictions imposed by a cell or its modules.	Enterprise integration Information system is open architecture, client-server based.