SSC05-II-4

The Standardization Process: What Works, What Doesn't

Doug Caldwell
CTO & V.P., Engineering

Ecliptic Enterprises Corporation
Pasadena, CA

9 Aug 2005

Contents

- The Standardization Process

- Example #1: FutureBus+ / Profile S
- Rule #1: Solve the Right Problem

- Example #2: Ada vs. C
- Rule #2a: You Can't Legislate Behavior
- Rule #2b: Technology is a Moving Target

- Example #3: PC Platforms
- Rule #3: Proprietary Standards are Problematic

- A Standardization Checklist

The Standardization Process

- **Evolutionary**
  - Individual users adopt an existing solution.
  - A user community emerges.
  - The community codifies its practice.

- **Revolutionary**
  - A group identifies an unmet need.
  - Group creates a solution (design by committee).
  - The group seeks adoption of its solution.

- **Revolutions are not committee activities!**

Futurebus+ / Profile S (Space)

- **~1990**
  - 16-bit parallel VMEbus is adopted by space community.
  - Adoption reduces costs, speeds system developments.
  - But it's "only" 16-bit, "only" single string, etc.
- **1992**
  - IEEE Computer Society establishes working group 896.10.
- **896.10 Purpose:**
  - "Current backplane protocol standards do not address the unique requirements associated with spaceborne processing systems.
  - "This standard will establish and specify the requirements (bus, mechanical and electrical) to support implementation of spaceborne Futurebus+ based processing systems."

Futurebus+ / Profile S Attributes

- **Flexible high performance architecture**
  - Support for 32-, 64-, 128-, and 256-bit wide data paths (in the era that VMEbus supports 8-, 16-, and 32-bit data)

- **High-reliability & high-availability configurations**
  - Dual-bus option for board-level block-redundant cold-spares
  - Triple modular redundant option for real-time operate-through
  - Secondary serial bus on backplane for reconfiguration
  - Conduction-cooled form factor (IEEE 1101.4)

Futurebus+ / Profile S Timeline

- **1992: Committee established**
  - Members represent major primes & government users, L/V & S/C builders, component vendors.
- **1997: Standard adopted by IEEE**
  - IEEE 896.10-1997
- **2003: Standard withdrawn**
  - No users; no systems deployed.

  …not even among the organizations represented on the committee.
- **What went wrong?**

The Death of Futurebus+/S

- **Design by Committee**
  - S/C guys wanted high-rel (block redundancy).
  - L/V guys wanted high-availability (TMR).
  - Futurists wanted high performance.
  - Everyone was representing his own needs.

- **No user was willing to saddle the burdens.**
  - Mass & power penalties for unused features.
  - Cost to develop completely new capabilities.

- **VME was "good enough" for real users…**

  … and CompactPCI was on the way.

Rule #1:
Solve the Right Problem

- **The perfect standardization candidate:**
  - A technical solution exists.
  - At least a few users have adopted that solution.
  - Others would like to leverage predecessors' learning curve.
  - Standardization codifies what's already being done.

- **If no technical solution exists:**
  - Identify the real needs (using good engineering practice).
  - Address the need generally.
  - Don't pile on nice-to-have additional features.
  - Get others involved early.

- **These are same steps as building any product!**

Example #2: Ada versus C

- **Problem (1975):**
  - Reduce the spiraling cost of DoD project software.
  - Reduce the number of programming languages used on DoD projects (from ~400!).

- **Solution**
  - Create a new, object oriented language.
  - Incorporate language features that:
    - Reduce the intrinsic defect rate;
    - Facilitate the synthesis of large projects (e.g., reuse);
    - Reduce the cost of maintenance.
  - Mandate its use on all DoD projects.

Ada Timeline

- **1975: *Higher Order Language Working Group* created**
- **1976: Language requirements established**
  - no existing language is deemed acceptable

- **1977: RFP released**
- **1979: Winner selected; language is named ‘Ada’**

- **1983: Ada is standardized: MIL-STD-1815A-1983**
- **1983: DoD mandates it for all mission-critical S/W**

- **1997: Dod embraces commercial standards**

Ada is “Better”, but C Won

- **Ada objectives were met:**
  - Total life-cycle costs are much lower than C/C++.
  - Defect rates are lower.
  - Product reliability is higher.
  - (And fewer languages are in use today than in 1975.)

- **But today, for embedded real-time software development projects (Ada’s primary target), C/C++ personnel are demanded ~10x as Ada.**

- **There remains perception that Ada isn’t the right tool – for the very applications for which it was developed.**

Rule:
Don't Try to Legislate Behavior

- **Ada mandate allowed waivers.**
  - Waivers needed to avoid short-term costs.
  - Loopholes allowed non-believers to avoid Ada.
- **Creative people don't like dictates.**
- **If you need a mandate,**
  - An autocratic central authority better exist;
  - Your solution isn't good enough to stand on its own;
  - You can't convince a jury of your peers.

Rule:
Expect Technical Change

- **Technology happens.**
  - The world won't stand still for you.

- **C evolved in parallel with Ada (~'73 – '83),**
  - …but in a different environment – academia.
  - Universities created C programmers (adherents).
  - C++ evolved from this already-willing base.

- **Keep development cycles short.**
  - Involve users from the beginning.

Example: PC/AT, PC/104, Macintosh

- **PC/AT (ISA) Bus was used as a *de facto* standard long before it was formalized.**
  - IBM did not attempt to protect their creation.
- **PC/104 was created to support expansion.**
  - Ampro encouraged others to adopt its invention.
- **Macintosh architecture is closed.**
  - Apple protects the Mac from unworthy intrusions.
- **Most PCs are IBM Clones, not Macs.**
- **Mac architecture is not used in embedded applications.**

Rule: Proprietary Standards are Problematic

- **Standards can be maintained by:**
  - Standards bodies (e.g., IEEE, ANSI, ISO)
  - Industry groups (e.g., PCIMG, PCMCIA, USB)
  - Governments (e.g., MIL-STD, NFPA / legislation)
  - Companies & smaller institutions

- **Few proprietary standards have worked.**
  - I^2C licensed through chip vendors.

A Standardization Checklist: The Problem

- **Problem Statement**
  - What problem am I solving?
  - Do I understand how others have addressed this problem?

- **Problem Type**
  - Is this a technical, process, or a personnel problem?
  - Can it be solved with a widget or might it be better solved with a standard procedure?
  - Can the solution be packaged in a way that is easily conveyed to its intended audience? Can it be “sold”?

A Standardization Checklist: The Users

- **Market / User Community**
  - Who cares?
  - Whose problem is this?
  - Does anyone else actually have the same problem?
  - Can it be stated to describe a target market or community?
  - Are my assumptions about the intended audience correct?
  - Have I really researched the target market or community?

- **Market / User Value**
  - What is the likely perceived value of the solution?
  - Is that target market likely to accept the costs associated with my solution, whether direct financial costs or indirect costs (e.g., mass, power, complexity, etc.)?

A Standardization Checklist: The Sponsor

- **Host Organization Value**
  - What will my organization get out of this?
  - Will we give it away to enhance our other business?
  - Will we try to protect it and license it?
  - What is the risk that others might object to such apparently self-serving behavior and then develop their own solutions?

A Standardization Checklist: Implementation

- **Implementation Approach**
  - How will I get a prototype built?
  - Who will be my beta-testers?
  - Should I do this internally or seek outside partners?
  - Would potential customers or potential competitors be the best partners? (Having a potential competitor validates the existence of a market, creates a unified front, and might make raising capital easier.)
  - How quickly can I get the ideas developed?
  - Will the need still exist when I'm done?

- **End Game**
  - How will I know that I'm done?
  - What are my success criteria?

Free-Market Standardization

- The “Checklist” mirrors any product development for a competitive marketplace.
- The Small Sat community is a dynamic marketplace of entrepreneurial players.
- Develop solutions to your technical problems, with an eye for letting others use what you come up with.

Create standards as you would create products.
Let natural selection decide what’s good, what’s not.