SIOUX EMBEDDED SYSTEMS

Architecting & Agile: *friends or enemies?*

Lessons learned from a project,
concluding with do's and don'ts

Ger Schoeber
Executive Consultant

May 19th, 2010

SATURN 2010

SIOUX EMBEDDED SYSTEMS

Question:

***Architecting* & *Agile*,**

friends

or

enemies *?*

SATURN 2010

Overview

- Quick introduction Sioux, Ger Schoeber
- Architect, Architecture, Architecting
- Agile

} Complementary?

- Incremental multi-disciplinary approach
- The Maestro project
  - The environment and the main drivers
  - The approach
    - Development Project
    - Architecting Process
  - The challenges and the lessons learned
  - Information Radiators and NFRs
- Summary, do's and don'ts

SATURN 2010

SIOUX / Ger Schoeber

SIOUX

- Domain: Embedded Systems & Technical Automation
- Machine Control, Medical Systems, Automotive, Consumer Electronics
- Software & electronics development
- Development Centre Projects: fixed price …. risk reward
- Partnering in product development
- Platform product: Machine2World® remote monitoring & control
- BinC – Be in Control: integrated security solutions for businesses and consumers (end2end solutions)
- Consulting and assessments: software process, system architecture (both world wide)

Ger Schoeber

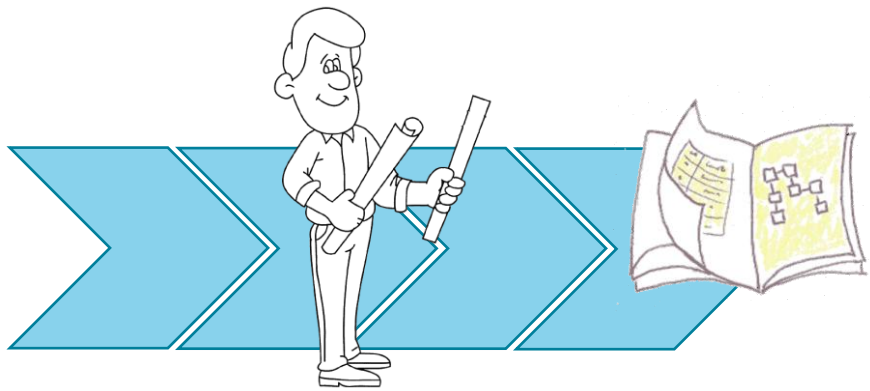
- 48, married, 2 children and a dog
- Executive Consultant,
- System Architect, Agile Project Management
- Trainer: SW Engineering/System Architecting
- Coach for System Architects
- 26 years experience in embedded and real-time systems:
  - Distributed real-time OS development
  - High way management systems
  - Digital Printers
  - Patient Monitoring systems
  - Fail safe systems
  - Baggage Handling systems
  - Interactive Digital TV development
  - Touch Screen Remote Controls
  - Express Parcel systems
  - Electronic Gates Remote Monitoring & Control
- *1984 Philips*
  1989 High Tech Automation
  2001 Task Switch
  2008 Sioux

SATURN 2010

SIOUX EMBEDDED SYSTEMS

System Architect | -ure | -ing

WHO | WHAT | HOW



SATURN 2010

SIOUX EMBEDDED SYSTEMS

"WHO"
System Architect: responsibilities

(wikipedia)

- Interfacing with the user(s) and sponsor(s) and all other stakeholders in order to determine their **needs**.
- Generating the highest level of system **requirements**, based on the user's needs and other constraints such as cost and schedule.
- Ensuring that this set of high level requirements is **consistent**, complete and correct.
- Generating a set of **acceptance** test requirements, together with the designers, test engineers, and the user, which determine that all of the high level requirements have been met, especially for the system-environment-interface.
- Performing **cost-benefit** analyses to determine whether requirements are best met by manual, software, or hardware functions; making maximum use of commercial off-the-shelf or already developed components.
- Perform a **partitioning** to allocate all present and foreseeable requirements into discrete partitions such that a minimum of communications is needed among partitions, and between the user and the system.
  Partitioning of large systems into (successive layers of) subsystems and components each of which can be handled by a single engineer or team of engineers or subordinate architect.
- Generating products such as **sketches**, models, an early user guide, and prototypes to keep the user and the engineers constantly up to date and in agreement on the system to be provided as it is evolving.
- **coaching** the design and implementation engineers, or subordinate architects, so that any problems arising during design or implementation can be resolved in accordance with the fundamental architectural concepts, and user needs and constraints.
- Ensuring that a maximally **robust** architecture is developed.
- Ensuring that all architectural products and products with architectural input are **maintained** in the most current state and never allowed to become obsolete.

SATURN 2010

SIOUX EMBEDDED SYSTEMS

"WHAT": What is an architecture?

"A software architecture is the **development product** that gives the **highest return** on investment with respect to quality, schedule, and cost."
(Software Architecture in Practice – Bass/Clements/Kazman)

Architecture:

Blueprint, foundation, vision, coherent form or structure, components connections and constraints, for the system and the project developing it.

➔ ***the conceptual glue***

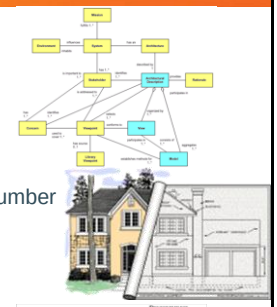
(Documenting Software Architectures – Clements/Bachman/Bass/Garlan/Ivers/Little/Nord/Stafford)

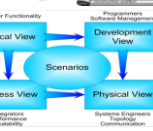
SATURN 2010

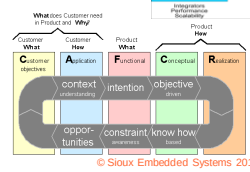
SIOUX EMBEDDED SYSTEMS

"WHAT" System Architecture: H2 document

- IEEE1471:
  Stakeholders have Concerns, from their Viewpoints ➔ Views
- Construction business:
  separate drawings per stakeholder ➔ bricklayer, electrician, plumber
- Kruchten, "4+1":
  separating the different parts and put a focus per view on an essential item:
  logical view, process, development, physical, use cases
- CAFCR – 5-view framework, Gerrit Muller:
  Customer objectives, **A**pplication view, **F**unctional, **C**onceptual, **R**ealization view



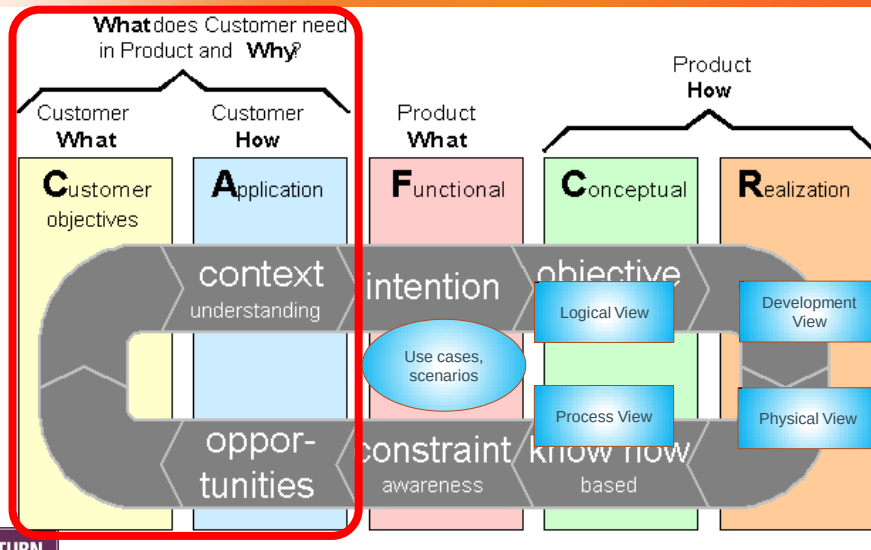




SATURN 2010

SIOUX EMBEDDED SYSTEMS

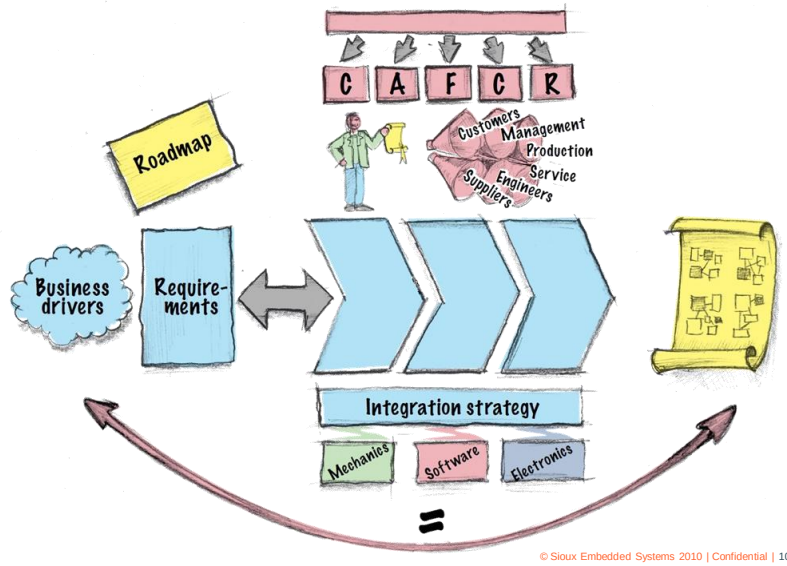
CAFCR and "4+1"



SATURN 2010

SIOUX EMBEDDED SYSTEMS

"HOW" System Architecting process



SATURN 2010

Agile?

- What is **agile** and why is it **important**?
- **Changes**:
  - Interfaces to the **environment** / other systems
  - **Customer** needs
  - **Technological** opportunities
  - Other and new **markets**
- BDUF *(– big design up front)*
  - A lot of work at least twice…
    - Once for creating the full design in an early stage
    - Then the update of all of it due to the external changes over time
  - *Agile/lean ➔ defer decisions to the latest possible moment*

SATURN 2010

Agile Manifesto

(www.agilemanifesto.org)

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

12 Principles behind the Agile Manifesto

(www.agilemanifesto.org)

1. Our highest priority is to satisfy the **customer** through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development.
   Agile processes harness change for the customer's **competitive** advantage.
3. Deliver working software **frequently**, from a couple of weeks to a couple of months,
   with a preference to the shorter timescale.
4. Business people and developers must work **together** daily throughout the project.
5. Build projects around motivated individuals.
   Give them the environment and support they need, and **trust** them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team
   is **face-to-face** conversation.
7. **Working software** is the primary measure of progress.
8. Agile processes promote sustainable development.
   The sponsors, developers, and users should be able to maintain a constant **pace** indefinitely.
9. Continuous attention to technical excellence and good **design** enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing** teams.
12. At regular intervals, the team reflects on how to become more **effective**, then tunes and adjusts its behavior accordingly.

SATURN 2010

Question:

"Architecting & Agile, friends or enemies?"

System Architecting

Creating and Building Structures or Systems

fit
balance
needs
technology

(The Art of Systems Architecting, Maier/Rechtin)

Agile Development

Speculate
Collaborate
Learn

decide without knowing the result
constructively on decisions made
re-decide to improve

(Jim Highsmith, 1999)

SATURN 2010

System Architecting and Agile

System Architecting

- Future
- Vision
- Direction
- Foundation
- Stable, robust
- Multi-disciplinary

Agile Development

- Short term
- Speed
- Changes in direction
- Adaptions
- Changes in solution
- Software engineering

SATURN 2010

System Architecting and Agile, complementary?

SATURN 2010

System Architecting and Agile

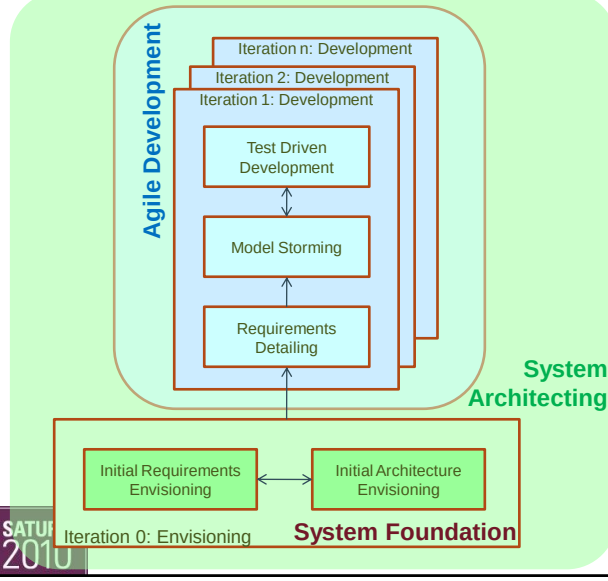
System Architecting	Agile Development
Future	Short term
Vision	Speed
Direction	Changes in direction
Foundation	Adaptions
Stable, robust	Changes in solution
Multi-disciplinary	Software engineering

Create a rough initial **foundation** based on the future **vision and direction**.
A stable and robust **multi-disciplinary system outline**, inspired by **customer needs** and **technological opportunities**, **revisited** on a regular basis.

then

Use a **time-boxed iterative approach** for detailing the **requirements**, **fit criteria**, **design** models and **realisation**.
This is derived from the multi-disciplinary system outline implemented **for all disciplines**.

Software Architecting & Agile Development



(Based on AMDD, Agile Model Driven Development Scott W. Ambler)

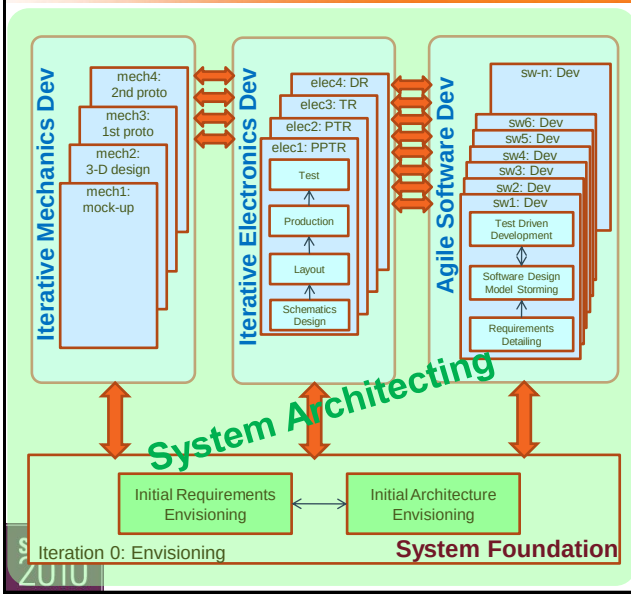
- Develop working system via a test first approach
- Details captured in the form of executable specifications

- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Identify the high level scope
- Identify initial 'requirements stack'
- Identify an architectural vision

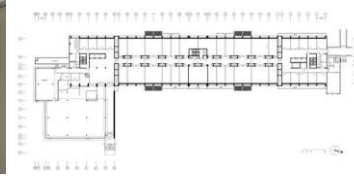
System Architecting & Multidisciplinary Iterative/Agile Development

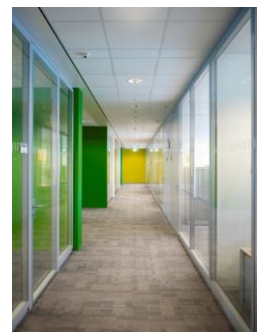


- Develop working system via:
  - Early user feedback
  - Early integration steps
- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later
- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration
- Identify the high level scope
- Identify initial 'requirements stack'
- Identify an architectural vision at system level

High Tech Campus 37, Eindhoven, The Netherlands








Philips Pronto (project name: Maestro)

Challenge @ start of 2005:

- **Next generation touch screen remote controls for high end home theater systems**
- **Roadmap** 2006 - 2010
- First product development, market introduction **Sept 2006** @ CEDIA show in Denver
- Introduce **System Architecting Process**

- My role: **System Architect** (multi-disciplinary and multi-site)

SATURN 2010

High End Home Theater System

- **Video**
  - TV, satellite, cable, terrestrial
  - Video, DVD, Blu-ray
  - (n)vod, time-shift recording, pvr
- **Audio**
  - CD, Audio Server, PC
  - Multi-room, centralized content
  - Controlled from multiple rooms
- **Lighting**
  - On/Off, dimming
  - Based on scenes
- ***Extensions***
  - *Home automation*
  - *Security control*

SATURN 2010

Market Characteristics

- Niche market
  → *low volumes*
- $10k - $70k … $ 300k …
  → *package deals*
- Specialized dealers
  → *not the main stream CE shops*
- Custom installer market
  → *professional installers, no hassle for end users*

SATURN 2010

Roadmap Continuation Challenge

SATURN 2010

‘Status Quo’, Technology

- Mitsubishi M32C processor
- embOS RTOS from Segger
- C based application
- Win32 based Editor

- Freescale MX1 processor
- Montavista Linux
- C based application, OpenPeak “IP”
- No editor

?

- Customer Installer market: full control of UI / graphics / navigation.
  ➔ Use an editor as for existing ProntoProNG version.
- M32C/embOS limits reached. No embOS support on more powerful processor.
  ➔ Use Freescale/Linux base
- Ethernet/WiFi is becoming defacto standard in houses
  ➔ Design new extender(s) based on Ethernet/WiFi communication

SATURN 2010

Pronto Product Family Line

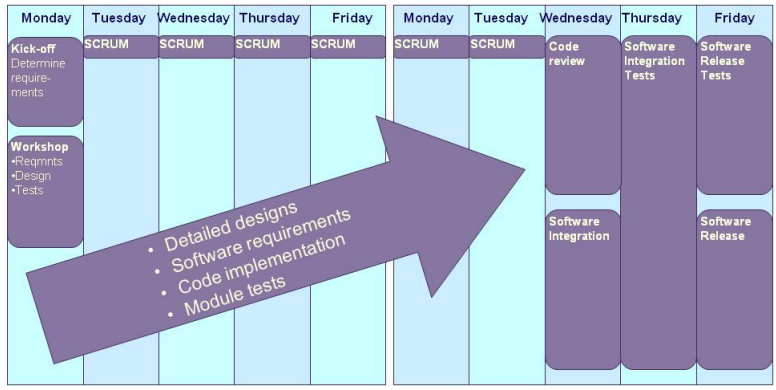
2006 2007 2008 2009

Approach of the Total Project

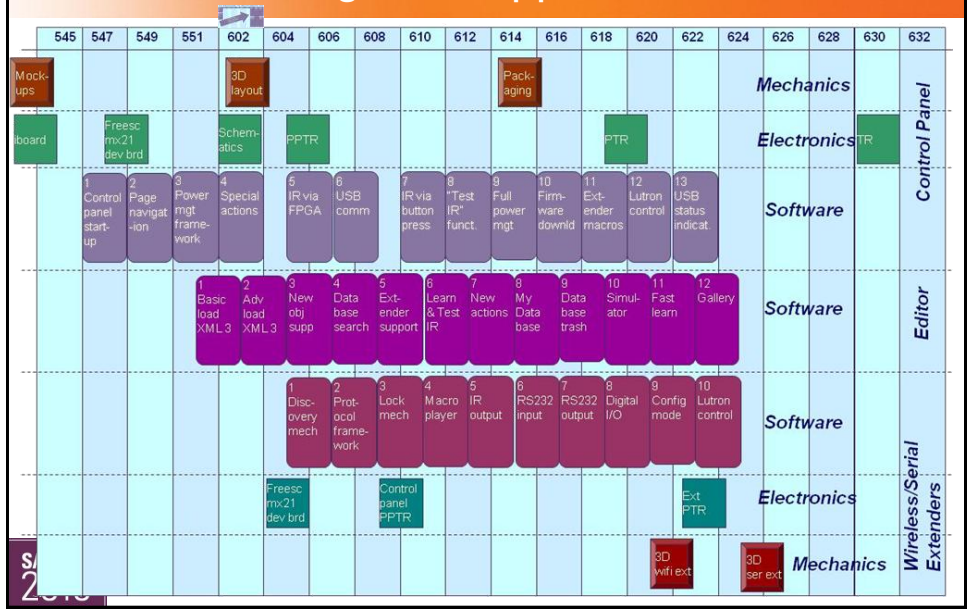
- Software Development Project:
  - 2-weekly increments
  - Daily Scrum
- 1st Pronto System Project (= panel, extender, editor)
  - Incremental development and integration
  - Multidisciplinary development
  - Multisite development
- 1st Pronto System Project lifecycle
  - Feasibility
  - Preparation
  - Development
  - Market Introduction and Production
- Overall Pronto Project Roadmap

➔ *overview on next couple of slides*

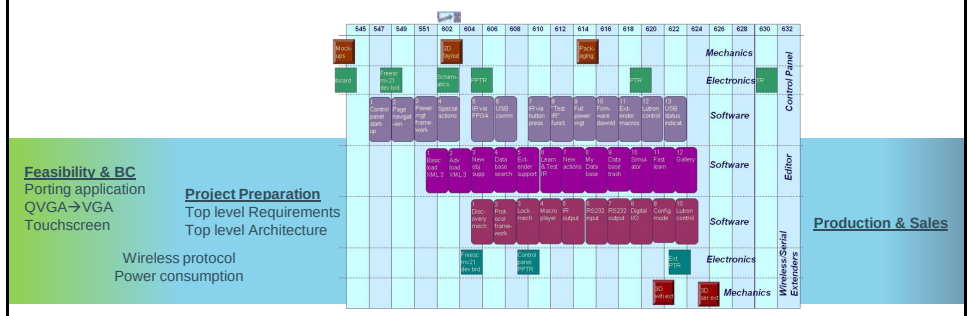
Project Approach: Software increment



Incremental development & integration approach



Pronto project



Pronto Roadmap

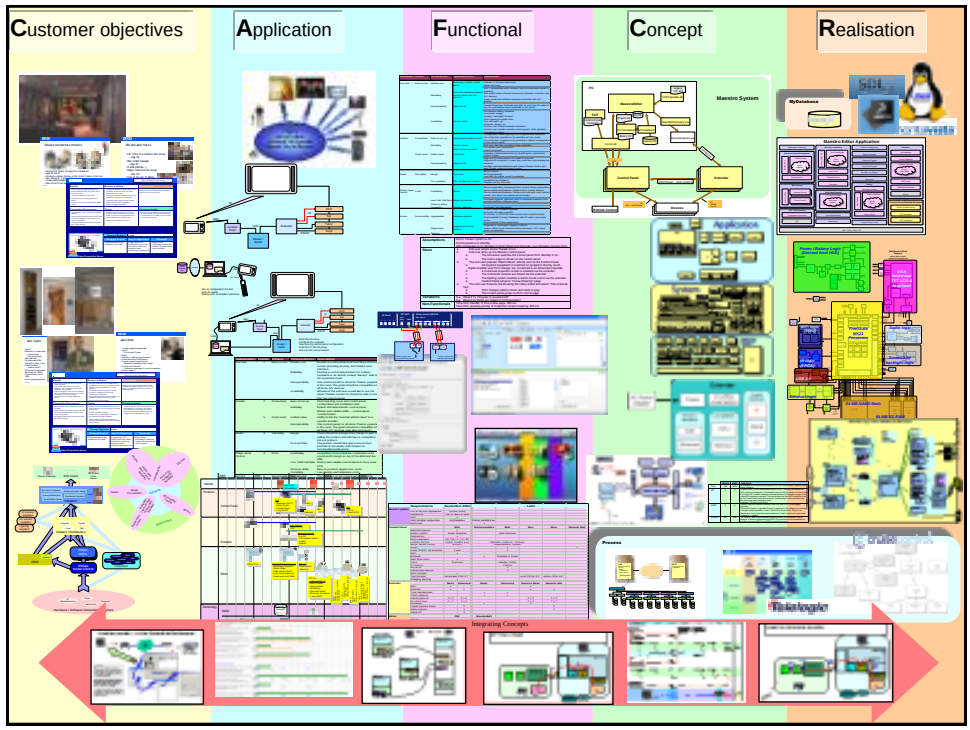
2005 2006 2007 2008 2009

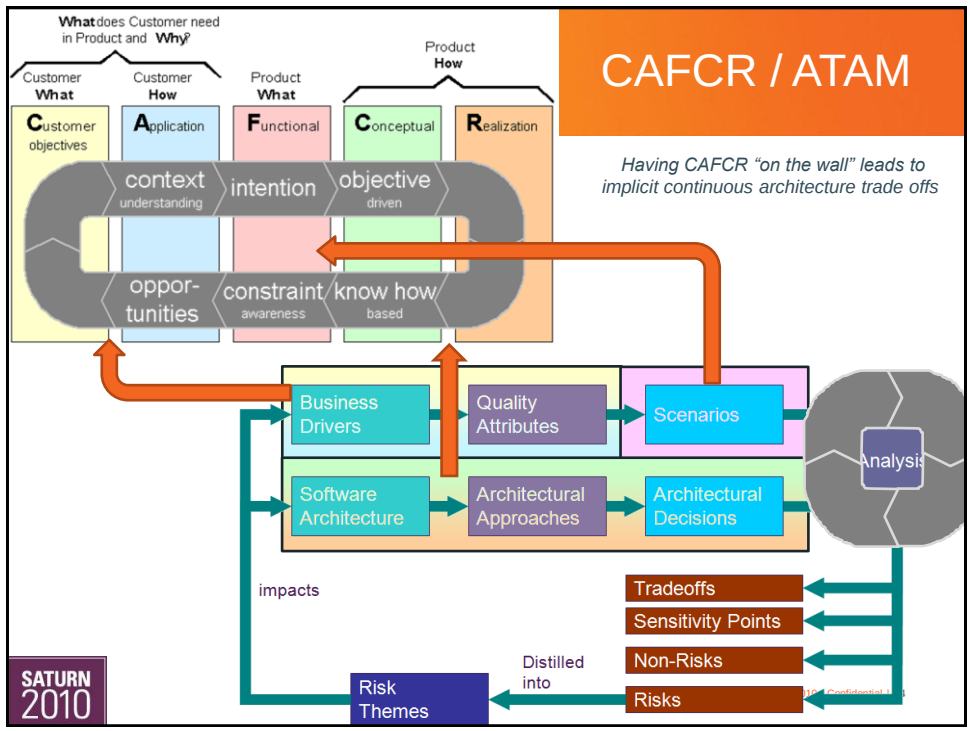
SATURN 2010

Pronto Roadmap, visualising architecting process

2005 2006 2007 2008 2009

SATURN 2010


Customer objectives
Application
Functional
Concept
Realisation


What does Customer need in Product and Why?
Customer What
Customer How
Product What
Product How
Customer objectives
Application
Functional
Conceptual
Realization
context understanding
intention
objective driven
oppor-tunities
constraint awareness
know how based
CAFCR / ATAM
Having CAFCR "on the wall" leads to implicit continuous architecture trade offs
Business Drivers
Quality Attributes
Scenarios
Software Architecture
Architectural Approaches
Architectural Decisions
Analysis
impacts
Tradeoffs
Sensitivity Points
Non-Risks
Risks
Distilled into
Risk Themes
SATURN 2010

Project and Technical challenges

- Feasibility phase
  - Screen size
  - Performance
  - Bus bandwidth
  - Porting application
- Project preparation
  - Software project size estimation
  - WiFi / Ethernet reliability
  - Power usage
- Development
  - Incremental approach
  - Integration strategy
  - Non-functional requirements
- Market Introduction and Production
  - Buy-in from market
  - Business case

SATURN 2010

➔ *Detailed in next couple of slides*

Feasibility Challenges

- Mockups
  - Mechanics
  - UI
    - Control panel graphics
    - Editor dynamics
- Technology
  - Performance
    - Memory usage
    - Bus usage
    - Processor usage

Feasibility projects

1. Pronto application **port** to iBoard (MXL-Linux) platform
2. iBoard hardware and BSP SW platform extensions: **VGA** LCD, USB
3. Extender **supplier** selection & evaluation
4. "MyDatabase": prototype for Editor
5. Communication with CE servers
6. CF format Extensions
7. RS232 feasibility
8. MaestroEdit project wizard
9. BoM & **power** calculations
10. **WiFi** feasibility – protocol design

What I learned

1. Early feedback from the market (end-users, installers) is essential:
   - The market determines the usefullness of a product.
   - Early involvement leads to buy-in.
   - It is impossible to define the success of a product behind the desk!
2. Prototyping technical issues is a must:
   - Separates practice from theory
   - Hottest issues in our case:
     - battery power usage
     - wifi reliability

SATURN 2010

Project Preparation Challenges

- Define:
  - Project Plan
  - System Requirements
  - System Architecture
- Effort estimations (software intensive)
  - Work break down
  - Development lead time: 6 months
  - Ramp-up of development teams within one month (2 dev sites)
- Feasibility, risks:
  - Battery power usage
    - Detailed calculations
    - Cross check with market
  - WiFi protocol:
    - TCP: not real-time, UDP: not reliable
    - ➔ define reliable protocol on top of UDP

What I learned

1. Don't invest time in detailing all req/arch before starting project ➔ market window.
2. Do invest effort in highest feasibility risks for final go/no-go of a project. Once defined, follow all results closely during the project
3. Have experts on board, in our case: using Linux ➔ you need a guru.
4. Multiple site development: solve the communication barrier by frequent and regular face2face meetings.

Development Challenges

- Agile approach, incremental working
  - Requirements at start of sprint
  - Focus on non-functional requirements
  - Software: continuous integration
  - System: early integration slots
- Technical
  - USB: interoperability
  - WIFI AP: interoperability

What I learned

1. Have a workshop at start of sprint: req / design / system interfaces / test
2. Have Continuous Integration and Test Driven Development as a standard wow.
3. Define non-functional requirements and have these tested as early as possible in the development cycle.
4. Stop functional or feature development as soon as NFRs are below target.
5. Find multi-disciplinary integration opportunities in the earliest possible phase.
6. Never rely on standardization: external products will never comply to the standard! You always must build in interoperability yourself!

Market Introduction and Production Challenges

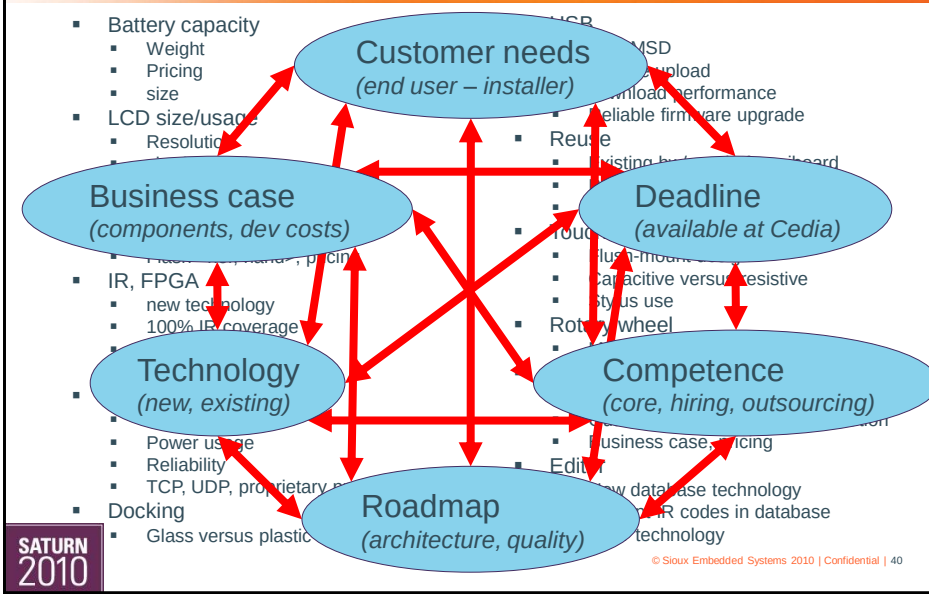
- Market Introduction
  - Market Introduction Plan
  - Phased approach
    - Teasers
    - Trainings
    - Beta releases to selective group
- Production
  - Testability in the factory

What I learned

1. Phased market introduction is a must to get buy-in.
2. Factory is not in sync with product development:
   - Input for testability from factory point of view is a must.
   - At the moment the testability requirements are needed as input, concerning stakeholders have very different priorities (production of previous products).
3. The necessity of convincing sales in selling solutions instead of selling products.

SATURN 2010

Technological choices → Continuing architectural challenges



SATURN 2010

SIOUX EMBEDDED SYSTEMS

Information radiators, dashboards, short feedback loops

Ger's statement:

IF you have to work in a project, where:

- the team > 1 person, OR
- the project is for a customer

THEN

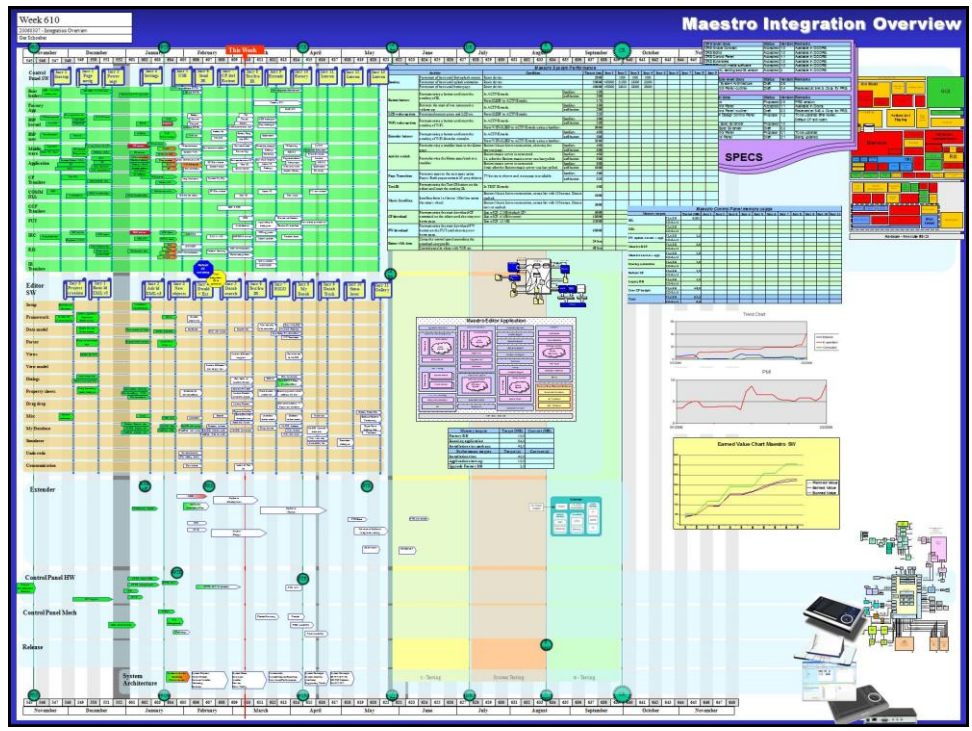
- miscommunication is the biggest risk!

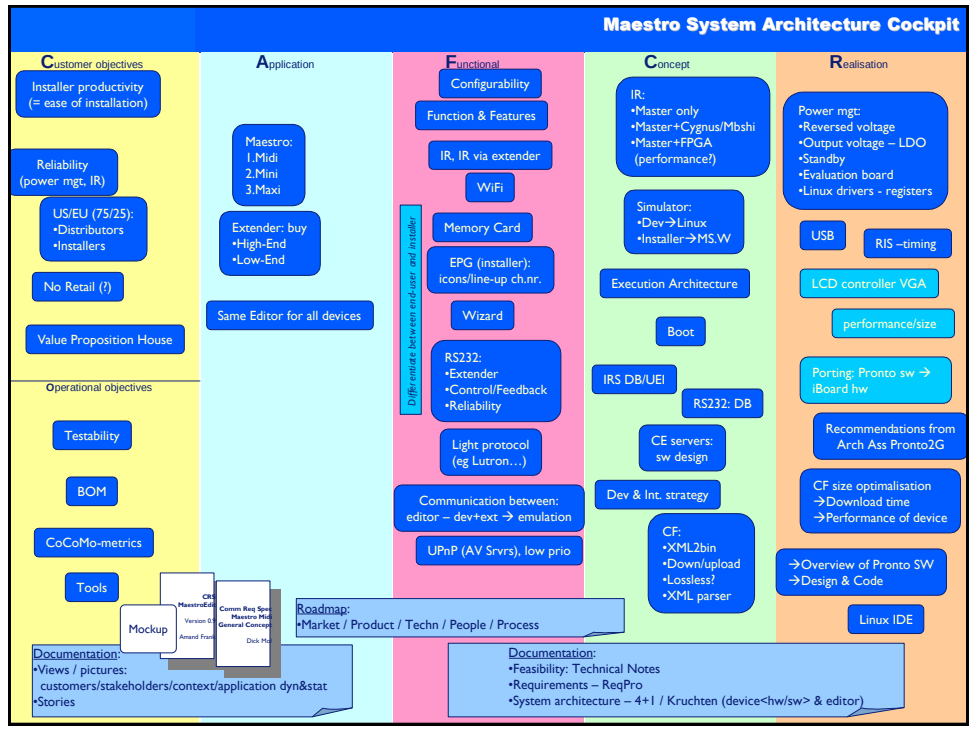
ENDIF

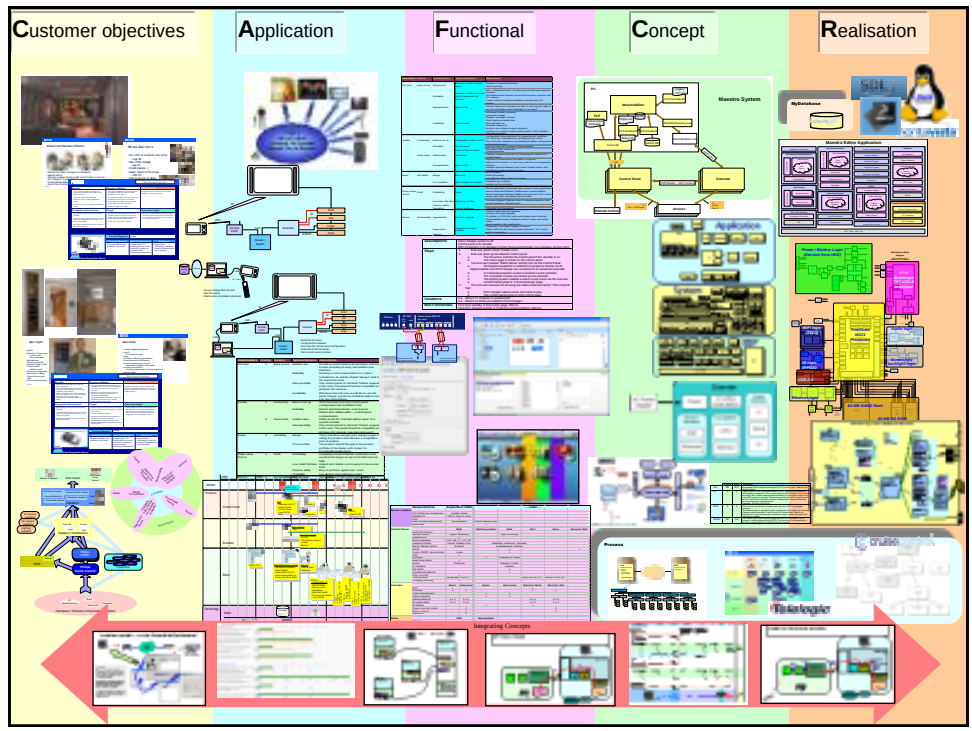
to tackle

➔ *Some information radiator examples on the next slides*

SATURN 2010




Maestro System Architecture Cockpit
Customer objectives
Application
Functional
Concept
Realisation
Installer productivity (= ease of installation)
Reliability (power mgt, IR)
US/EU (75/25):
•Distributors
•Installers
No Retail (?)
Value Proposition House
Operational objectives
Testability
BOM
CoCoMo-metrics
Tools
Mockup
Maestro:
1.Midi
2.Mini
3.Maxi
Extender: buy
•High-End
•Low-End
Same Editor for all devices
Configurability
Function & Features
IR, IR via extender
WiFi
Memory Card
EPG (installer): icons/line-up ch.nr.
Wizard
RS232:
•Extender
•Control/Feedback
•Reliability
Light protocol (eg Lutron…)
Communication between: editor – dev+ext → emulation
UPnP (AV Srvrs), low prio
Differentiate between end-user and installer
IR:
•Master only
•Master+Cygnus/Mbshi
•Master+FPGA (performance?)
Simulator:
•Dev→Linux
•Installer→MS.W
Execution Architecture
Boot
IRS DB/UEI
RS232: DB
CE servers: sw design
Dev & Int. strategy
CF:
•XML2bin
•Down/upload
•Lossless?
•XML parser
Power mgt:
•Reversed voltage
•Output voltage – LDO
•Standby
•Evaluation board
•Linux drivers - registers
USB
RIS –timing
LCD controller VGA
performance/size
Porting: Pronto sw → iBoard hw
Recommendations from Arch Ass Pronto2G
CF size optimalisation
→Download time
→Performance of device
→Overview of Pronto SW
→Design & Code
Linux IDE
Roadmap:
•Market / Product / Techn / People / Process
Documentation:
•Views / pictures:
customers/stakeholders/context/application dyn&stat
•Stories
Documentation:
•Feasibility: Technical Notes
•Requirements – ReqPro
•System architecture – 4+1 / Kruchten (device<hw/sw> & editor)

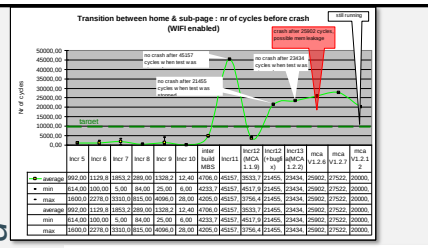

Customer objectives
Application
Functional
Concept
Realisation

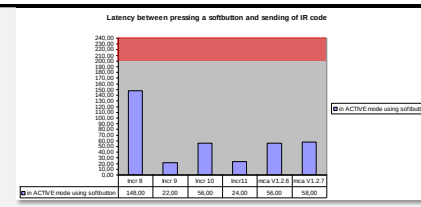
Lean CAFCR example

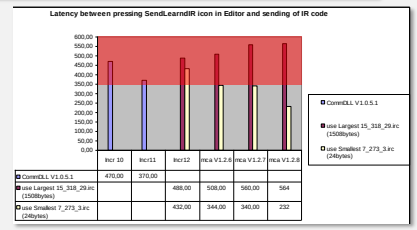
➔ *try at least to start with low-tech means*

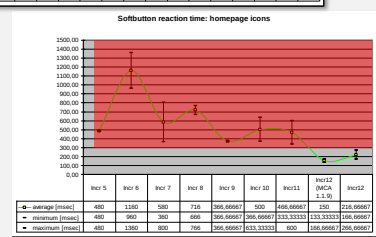
SATURN 2010

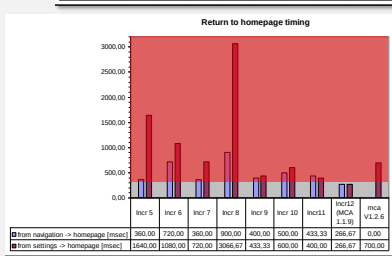
NFR dashboard

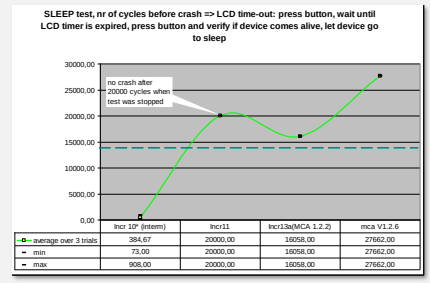












Lessons learned in general

DO's

- **Incremental/iterative** development with short **feedback** loops
- **Integration** as early as possible. This addresses the weak points in architecture/design (**interfaces**) in an early stage.
- **"Vertical" integration** strategy.
- Testing focus on **non-functional requirements** as early as possible.
- **Daily stand-up meetings** with the team, not only during development but in any phase of the project.
- Use **information radiators**, sheets on the wall for planning, status dashboard, architecture, business backgrounds.
- The **customer is never wrong**. Involve the customer whenever possible and use all the feedback you can get.
- **Integrate over the disciplines** as early as possible and as often as possible.
- Design a system from the point of view of the most important stakeholders (**decision makers** for buying the product).

DON'Ts

- Only **BDUF** and a lot of paper work at the start.
- Give priority to **functional aspects** over performance and 'ilities'.
- Write **documents for the shelf**.
- Have the **process in the lead**.
- **Communicate electronically** only.
- Start **without** an **architectural foundation** especially when working on a family of products.
- **Postpone integration** to the latest stage of the project.
- **Horizontal integration**: Integrate layer by layer.
- Have **user interfaces** designed by **technical people** only.

Question: *YES!*

"Architecting & Agile, friends ~~or enemies?~~"

System Architecting & Agile Development

SIOUX EMBEDDED SYSTEMS

References

Title	Author(s), Source
Agile Manifesto	*http://www.agilemanifesto.org*
Agile Modeling Effective Practices for Extreme Programming and the Unified Process	Scott W. Ambler *http://www.agilemodeling.com/essays/amdd.htm*
Architectural Blueprints The "4+1" View Model of Software Architecture	Philippe Kruchten *IEEE Software, November 1995, pp. 42-50*
CAFCR: A Multi-view Method for Embedded Systems Architecting Balancing Genericity and Specificity	Gerrit Muller *http://www.gaudisite.nl/Thesis.html*
Documenting Software Architectures Views and Beyond	Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford
Evaluating Software Architectures Methods and Case Studies	Paul Clements, Rick Kazman, Mark Klein
IEEE Std 1471-2000 Recommended Practice for Architectural Description of Software-intensive Systems	*http://www.iso-architecture.org/ieee-1471*
Lean Software Development An Agile Toolkit	Mary Poppendieck, Tom Poppendieck
Pronto	*http://www.pronto.philips.com*
Scrum and XP from the Trenches How we do Scrum	Henrik Kniberg
Software Architecture in Practice	Len Bass, Paul Clements, Rick Kazman
Software Architecture Organizational Principles and Patterns	David M. Dikel, David Kane, James R. Wilson
The Art of Systems Architecting	Mark W. Maier, Eberhardt Rechtin
The Toyota Way 14 management principles from the world's greatest manufacturer	Jeffrey K. Liker
Understanding A3 thinking A Critical Component of Toyota's PDCA Management System	Durward K. Sobek II, Art Smalley *http://a3thinking.com/*

SATURN 2010

SIOUX
EMBEDDED SYSTEMS

Source of your development.

www.sioux.eu
ger.schoeber@sioux.eu
+31 40 26 77 100

 51