An Open Market for Software Components

A Way to Increase Software Generation Efficiency by Some Orders of Magnitude

Hans van Leunen

Overview

- Current situation
- Why software components ?
- Why an open market ?
- How to design & build components ?
- How to design & build component based systems
- How to create an open market ?

Current Situation

- Viewpoint
- Before the dip
- What happens now
- After the dip

Viewpoint

- The viewpoint is taken from the eyes of the **high tech integrated circuit industry**
- Their customers are the system integrators

Before the Dip 1

- Requirements for skilled human resources in embedded software doubled each year
- It was expected that in a few years the **cost of software would outrun the income** on IC production
- Due to increase of complexity the robustness of the produced software **becomes questionable**

Before the Dip 2

- The industry must take measures to cope with expected call backs
- Buggy products, bought by early adopters will cause a buy refusal by the other part of the customers

What Happens Now

- Due to the long lasting dip in the market, lots of embedded software engineers are lead off
- These skilled human resources will start other jobs outside the domain of embedded software
- The pool of available skilled human resources for this industry branch is quickly diminishing

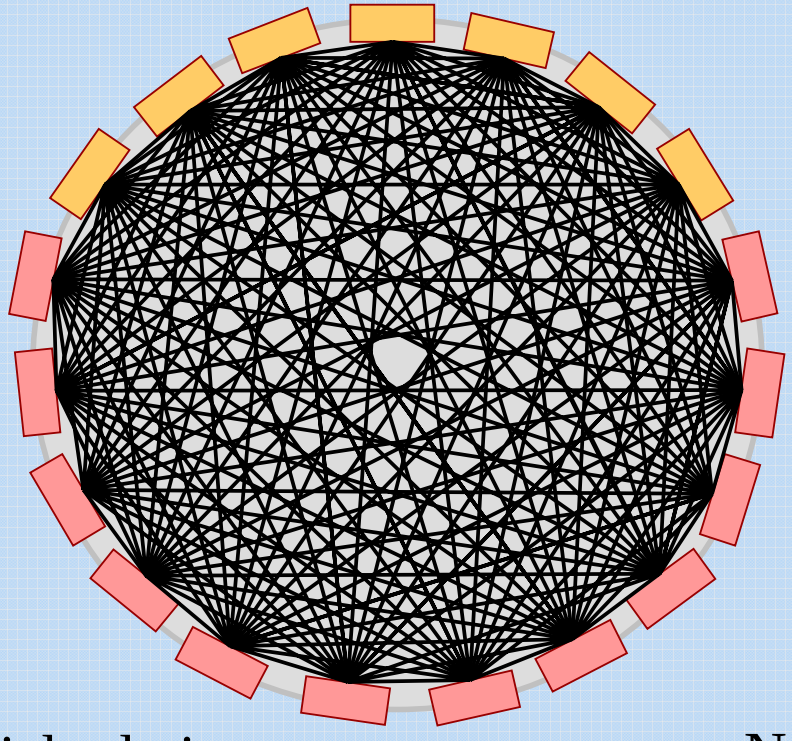
After the Dip

- Due to the smaller pool of human resources the point where **the requirements will surpass the availability of human resources** will come much earlier than was calculated before the dip
- Reaching that point will bring **structural obstructions** to high tech industry in stead of the the negative influences of the tidal economical waves that it encountered until now

Why Software Components ?

- Hard encapsulation
- Efficient reduction of complexity
  - Inside the components
    - More than two orders of magnitude
  - In the component based system
    - More than three orders of magnitude
- Guarantee of consistent behavior
- Hides intellectual property
- Market ready end-product

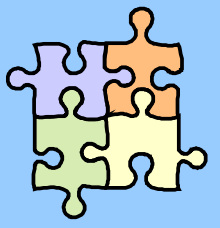
Relational Complexity in Monolithic System or Part



n(n-1)/2 potential relations

n = Nr of related items

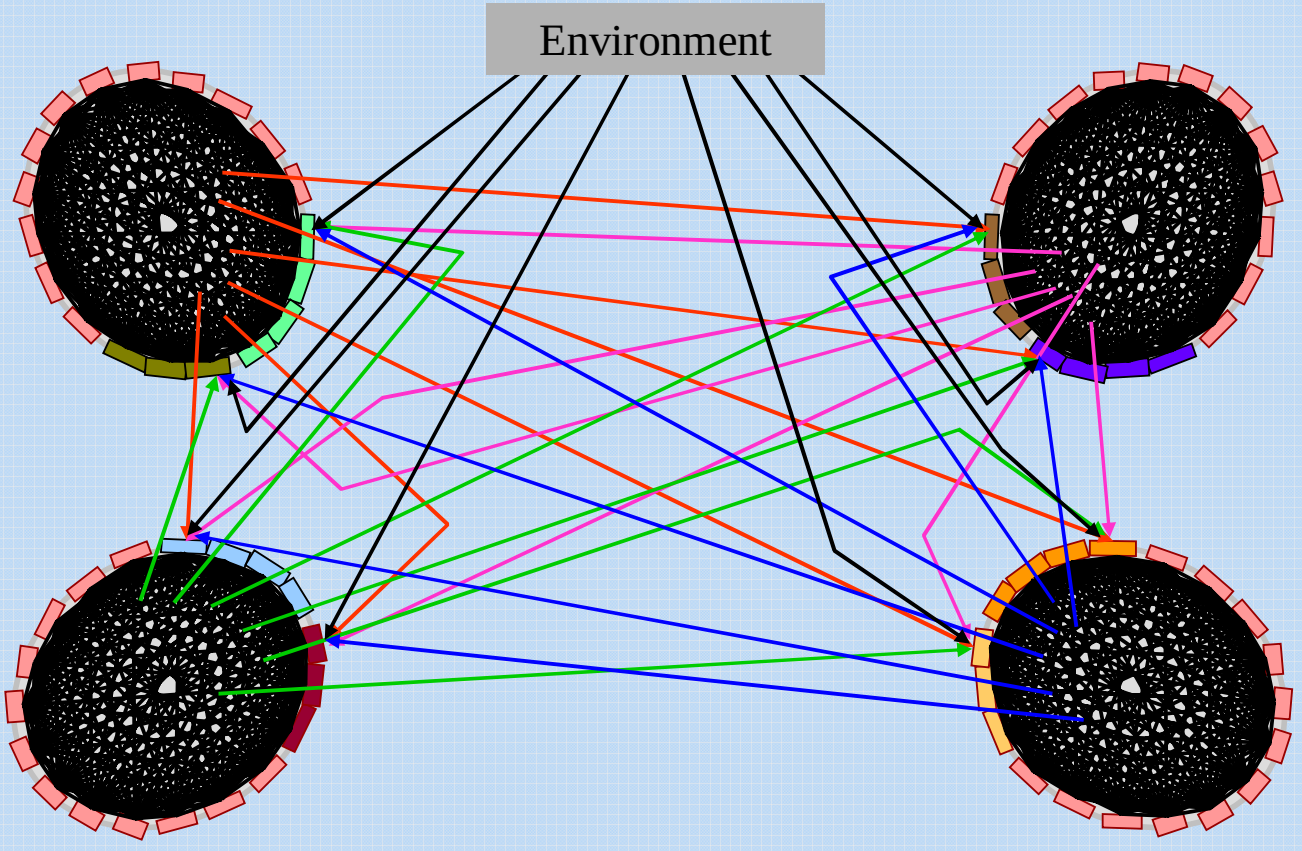
In numbers



- 100 items ⇒ 99 • 100 potential relations
- 1000 items ⇒ 999 • 1000 potential relations

- 10 modules containing 100 items ⇒ maximally 99 • 100 potential relations inside a module plus 9 • 10 relations between modules.
- Nobody sees more than 9990 relations!!!

Complexity in Component Based System



Easily two orders of magnitude better than monolithic case

Expert knowledge

- Components are excellent vehicles for the encapsulation of expert knowledge
- The system architect can apply this expert knowledge without the requirement that he must himself also be an expert in that domain

Why an Open Market ?

- Significantly increases reuse for popular design elements
- A single company can no longer do it all
- Competition lowers prices and increases quality
- Availability of more sources increases composition flexibility

How to Design & Build Components

- Choose a lean & mean component model
- Provide the design technology that let architects create skeletons of components
- Let domain specialists fill these skeletons with active code
- Combine sets of classes of these components in binary packages
  - Package on request of the customer
  - Package a coherent set of classes
- Publish the packages for sale

How to Design & Build Component Based Systems 1

- Provide a mechanism for selecting components
- Create skeletons of the components and check if they fit together
  - Static fit: require and provide interfaces
  - Dynamic fit: All real-time restrictions
  - Need scheduling/connection schemes
- Add a tailored infrastructure to the selected components

How to Design & Build Component Based Systems 2

- Use a flexible gluing technique that can cope with binary components
- Use a task scheduling technique that can cope with binary components
  - Paired tasks: primary & repair task (TAFT)
- Support static and optionally dynamic loading

How to Create an Open Market 1

- Publish formal specification documents on publicly accessible repositories
  - Packages
  - Component classes
  - Interfaces
  - Types
  - Other reusable design elements
- The repositories must have a standardized structure that provides
  - Navigation
  - Categorization

How to Create an Open Market 2

- Provide tools that retrieve formal specifications based on both
  - Document types
  - Categories
- Provide system configuration tools that support a solid licensing mechanism
- Multiple independent tool vendors must support the technology

All involved parties must be able to make a decent profit

- Arrange that standards enable cooperation between tool vendors that want to support this market
- Arrange that sufficient component suppliers publish their products on the repositories
- Arrange that sufficient system integrators make use of these services

Some details 1

- Components have special features
  - Require interfaces
  - Task scheduling interfaces
  - Notification interfaces
  - Hardware/software interfaces
  - Streaming interfaces
  - Paired tasks
  - Fixed and/or dynamic instances

Some details 2

Component based systems have special features

- For each package
  - Package manager
    - Class factory
- Tailored infrastructure
  - Relation manager
  - Task manager
  - Synchronization services

Current State 1

- Philips Semiconductors tried to establish a consortium of companies that will establish the proposed market
  - Several parties showed interest
  - PS built a relation with two tool vendors
    - I-Logix
    - TimeSys
- All long term investments in this technology are currently at zero level

Current State 2

- TWP created a demo-toolkit that shows the feasibility of most of the aspects of the proposed technology
- The first three tools support generic formal specification documents
  - A formal specification **document editor** that can create and edit document types and their instances
  - A **document exporter** that creates local repositories from existing documents and adds navigation and categorization utilities
  - A **document browser** that imports documents based on a set of selected document types and a set of categories

Current State 3

- The last two tools are dedicated to the generation and use of software components
  - A formal specification document editor that specializes on software components and component based systems
  - A system integration tool that uses retrieved and locally generated specification documents to produce prototypes and final versions of component based systems
  - The current version of the toolkit covers single-tasking systems.

Future

- The architecture of the demo-toolkit is already lead out to support real-time multitasking systems
- The full fledged demo-toolkit is planned to be ready before the end of 2003
- Tool vendors must adapt this technology and offer the production tools.
- It is expected that full acceptance of the technology will not occur before the first disasters due to lack of skilled human resources have happened in high-tech industry.

More info

- http://www.scitech.nl/inducom
- http://www.scitech.nl/inducom/repository