Highlights of our thinking on how to design, document, analyze, and implement architectures. Much of this is documented in more detail in recent papers.

CONVERGYS

Carnegie Mellon
Software Engineering Institute

Purpose

Software product line practice, to be successful, requires strategic insight and the ability to choose among competing alternative actions.

The bottom line for a choice is usually economic.

Therefore, the ability to predict the economics of alternatives emerges as a critically important capability.

Many economic models exist. Several are represented here.

We wanted to see them in action.

CONVERGYS

Carnegie Mellon
Software Engineering Institute

Panelists

Dale Peterson
Convergys, USA

Lamia Labed (Sana Ben Abdallah)
Ecole Nationale des Sciences de l'Informatique,
Campus Universitaire la Manouba, Tunisia

Jacco Wesselius
Philips Medical Systems, The Netherlands

Klaus Schmid
Fraunhofer Institute for Experimental Software Engineering
Germany

John McGregor
Clemson University, USA

CONVERGYS

Carnegie Mellon
Software Engineering Institute

Ground rules

1. Each panelist was given an initial scenario and three questions to answer. Each was allowed to ask whatever questions they wished. All questions and all answers were forwarded to all panelists.

2. Timing
   - This introduction: 5 minutes
   - Each panelist:
     - 10-minute overview of method and how it was used to solve the problem
     - 2 minutes of clarifying questions from audience
   - Questions and discussion: 30 minutes

CONVERGYS

Carnegie Mellon
Software Engineering Institute

The Scenario -1

An avionics manufacturer called Air-Bits has a product line of 11 products.

- 500KSLOC / 1000 modules
- 95% of SLOC in common between any two products
- 70% of *modules* in common between any two products

Air-Bits keeps a repository of 4000 modules.
New product is built via clone-and-own. They start with a similar product, copy it over, change the modules that need changing.

Question #1: Air-Bits has just received an award for a new system. Should they

- (a) clone-and-own the most similar system, and expect to re-program 300 of that product's modules, as they have in the past? Or,
- (b) comb their repository and look for the best fit for all 1,000 modules that will populate the new system? Here, the expectation is that they would only have to re-code 50 modules, being able to find the other 250 from *some* other product in their family.

CONVERGYS

Carnegie Mellon
Software Engineering Institute

The Scenario -2

Engineers are unhappy with the clone-and-own approach and want to re-engineer the repository, taking it from 4000 down to 1200 modules. They claim the cost savings in maintenance and testing will pay for the effort.

In an average year

- Each product comes out with a new version
- Changes to any one product touch 100 modules.
- 800 modules are changed throughout the year. Of these,
  - 200 are common across the whole family;
  - 200 are modules that are changed for one system only;
  - 400 reflect changes that have to be propagated to multiple (3, on average) cloned copies of a module.

Question #2: How long, if ever, will it take to recoup the re-engineering effort based on the difference in testing and maintenance cost?

Question #3: What will it cost to add a twelfth product to the family

- (a) if the engineers get their way?
- (b) if the product line continues down the current clone-and-own path?

CONVERGYS


Software Product Line Conference
Dale Peterson
September 28, 2005
CONVERGYS
Outthinking Outdoing
Copyright ©2005 Convergys Corporation

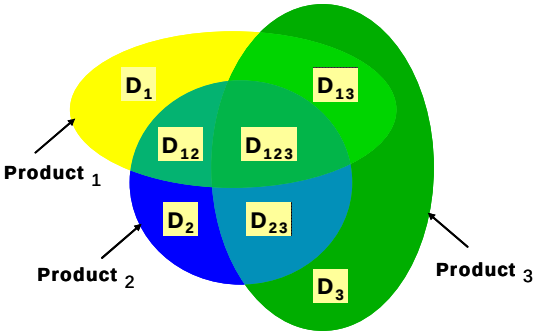
Model Overview

Motivation

- Relate SPL benefits to concepts of commonality and variability
- Incorporate impact of productivity

Approach

- Compare effective demand* placed on development groups for two scenarios
  - Independent
  - Software Product Line



Requirements Space
Partition Size – Function Points
Emphasis on *Future* Requirements
Proactive Product Line Planning

$$\text{Independent Demand} = D_1 + D_2 + D_3 + \mathbf{2}(D_{12} + D_{13} + D_{23}) + \mathbf{3}\, D_{123}$$

$$\text{SPL Demand} = D_1 + D_2 + D_3 + D_{12} + D_{13} + D_{23} + D_{123}$$

* Demand is measured in terms of function points/year, SLOC/year, …

7 SPLC 2005 D R Peterson

Traditional approaches to reuse economics do not properly account for the fact that requirements across a line of N products may apply to 1, 2, 3, …, N of the products.

The Venn diagram illustrates the case for N=3. Overlaps in requirements that involve all three products imply a greater inefficiency (in the Independent case) in the development process than requirements that are shared by only two products.

In order to capture the potential benefits of SPL, an understanding of these overlaps is required.

Model Overview . . .

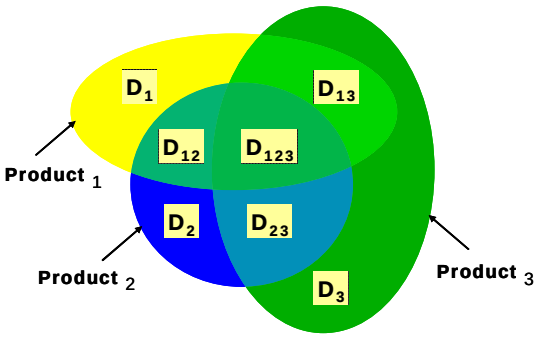
For N products, there are $2^N - 1$ partitions of the Venn diagram

- N = 11 => 2047 partitions!

Only require two parameters to capture relative demand

- Commonality **ω**
  - Fraction of demand involving 2 or more products
- Leverage **λ**
  - Average fraction of common demand relevant to a single product

Additional parameter to capture relative change in productivity **δp**



Solution Space
Partition Size - SLOC
Assumes an *Architecture*
Scalability, Performance, Cost, …
Constraints May Lower Size of Overlap

$$S^{SPL} = \left(\frac{Independent\ Productivity}{SPL\ Productivity}\right)\left(\frac{SPL\ Demand}{Independent\ Demand}\right)S^{Ind} = \left(\frac{1}{1+\delta p}\right)\left(\frac{1}{1+(N\lambda-1)\omega}\right)S^{Ind}$$

8 SPLC 2005 D R Peterson

There are two parameters that, together, uniquely determine the ratio of Independent demand to SPL demand: 1) commonality, and 2) leverage.

For the case of N=3 as illustrated above, the commonality is given by:
ω = (D12 + D13 + D23 + D123)/(D1 + D2 + D3 + D12 + D13 + D23 + D123)

The leverage (again for N=3) is given by:
λ = [2*(D12 + D13 + D23) + 3*D123]/[3*(D12 + D13 + D23 + D123)]

The leverage is a measure of how much of the common capabilities are used ("leveraged") by a given product, averaged across all products.

The second thesis behind the model is that transitions to an SPL require reengineering/re-development of the asset base in order to standardize on common functionality while simultaneously allowing for controlled variations in functionality. This transition will impact development productivity which in turn impacts the economics of the transition. Development productivity here is defined in terms of new/changed function points/SLOCS (or equivalent metric) per person-month (or year). It accounts for all effort associated with development, such as requirements, design, code, unit test, integration test, system test, documentation, configuration management, and project management.

Much has been written about the cost of developing for reuse, and the cost of reuse. However, these overheads may be offset by improved productivity resulting from a much more structured, modular asset base that is easier to enhance and maintain.

The parameter that is of relevance is the *relative* change in productivity, defined by:

A positive/negative value for δp indicates that development productivity is higher/lower in the SPL scenario than the Independent scenario.

$$p^{SPL} = p^{Ind} + \Delta p = p^{Ind}(1+\delta p)$$

Air-Bits

Observations

- 1000 “functions” implemented by 4000 modules => 4 clones on average
  - Maintenance nightmare!
- 6 Hours/SLOC @ 1680 hours per person-year => 280 SLOC/PY (!)
  - Development productivity is a major problem

The Air-Bits case study states that the effort to design, code and unit test changes to the existing code base is 6 hours per SLOC. Assuming 140 hours per person-month, that translates into 280 SLOC/person-year.

Question 1

Parameters

- F = Fraction of code modified
- T = Search time (hours) per module

Option a:

- Cost = 300 Modules x 500 SLOC/Module x 6 Hrs/SLOC x F = 900K x F Hrs

Option b:

- Cost = 50 Modules x 500 SLOC/Module x 6 Hrs/SLOC x F
  + 250 Modules x 4 Searches/Module x T Hrs/Search
  = (150K x F + 1000 x T) Hrs

Option b is preferred if

900K x F > 150K x F + 1000 T => T < 750 x F Hrs

The rationale for introducing the parameter “F” is that the Air-Bits case study does not say explicitly how many lines of code are modified in Option a (300 modules are “re-programmed”). If all three hundred modules are completely re-written, F would have a value equal to 1.

Question 1 . . .

Assume

- 5% of total code base is modified = 25K SLOC => F = 1/6
- T = 40 Hrs

Then

- Option a cost = 150K Hrs
- Option b cost = (25K + 1000 x 40) Hrs = 65K hours
- Option a – Option b = 85K Hrs

Option b also has the advantage of minimizing module proliferation

Recommendation: **Option b**

According to the Air-Bits case study, any two products have 95% code commonality. This suggests that the code bases for two products differ by .05 x 500K SLOC = 25K SLOC. We therefore assume that in Option a, 25K SLOC are modified across the 300 re-programmed modules. Then:

F = 25K SLOC/(300 x 500 SLOC) = 1/6

This value of F gives 150K hours for code modifications in Option a.

In Option b, only 50 modules are re-programmed instead of 300, so Option b requires (25/300) x 150K hours = 25K hours for code modifications.

Option b also requires searching the repository for the best fit for the remaining 250 modules. Since there are 4000 modules in the repository, and since each product uses on average 1000 modules, there are 4000/1000 = 4 versions of a given module on average. This results in 4 x 250 = 1000 searches.

Using a conservative assumption of 40 hours per search, Option b requires 40,000 hours to find the 250 modules. Total Option b effort is then 25K + 40K = 65K hours.

Question 2

To quantify benefits of transition to SPL

- Make conservative estimates
- Should perform (downside) sensitivity analysis
- Need to consider both Development (design, code, unit test) and Test (integration, system, …)
  - Development and Test require separate analysis
    - Some common development already
    - Test needs to be done with distinct module combinations

In working question 2, I've made the assumption that the scenario calls for a complete transition to a Software Product Line approach, even though that is not an explicitly stated objective. The rationale for this assumption is that it if an organization is going to make a major investment in reengineering their asset base, it would not be with the intent of maintaining a clone and own approach.

SPL benefits for development and test need to be separately assessed since there is some common development performed in the baseline scenario which needs to be factored into the Independent demand. On the other hand, we can expect that testing (integration/system/stress/…) will need to be done on a per product basis since each product consists of it's own unique set of modules, therefore there is no reuse of testing work across the products in the Independent case.

Secondly, there is new code that is written in addition to code modifications. Productivity numbers are not available for new development (the 6 hours per SLOC figure is much too low for new development). Therefore, new code is not factored into the development demand, only code modifications. I.e., the SPL benefits do not include any savings associated with new code development.

Question 2 . . .

Development

- Consider annual *change* activity

$$D_{Dev}^{Ind} = (1.5K\ Common + 4 \cdot 9K\ Cloned + 3.5K\ Unique)\frac{SLOC}{Yr} = 41K\frac{SLOC}{Yr}$$

$$S_{Dev}^{Ind} = \frac{D_{Dev}^{Ind}}{p_{Dev}^{Ind}} = \frac{41K\ SLOC}{Yr} \cdot \frac{6\ Hr}{SLOC} \cdot \frac{1\ Person - Yr}{1680\ Hr} \cong 146\ Persons$$

$$D_{Dev}^{SPL} = (1.5K\ Common + 9K\ Common + 3.5K\ Unique)\frac{SLOC}{Yr} = 14K\frac{SLOC}{Yr}$$

$$S_{Dev}^{SPL} = \frac{p_{Dev}^{Ind}}{p_{Dev}^{SPL}} \cdot \frac{D_{Dev}^{SPL}}{D_{Dev}^{Ind}} \cdot S_{Dev}^{Ind} = \frac{1}{(1+\delta p_{Dev})}\frac{14}{41} \cdot S_{Dev}^{Ind} = \frac{50\ Persons}{(1+\delta p_{Dev})}$$

$$\delta p = 0 \quad \Rightarrow \quad S_{Dev}^{SPL} = 50; \quad \Delta S_{Dev} \equiv S_{Dev}^{Ind} - S_{Dev}^{SPL} \cong 96\ Persons$$

The Independent development demand (for code modifications) can be computed in a straight-forward manner from the Air-Bits case study, which, together with the stated productivity, can be translated into a staffing requirement.

Similarly, the SPL development demand can be computed, which turns out to be approximately 1/3 of the Independent demand (14/41). Assuming no productivity improvement associated with the SPL transition (which as previously stated is a very conservative assumption), we obtain the SPL staffing requirement and thus the staff savings.

Question 2 . . .

Test

$$D_{Test}^{Ind} = (11 \cdot 19K\ Common + 4 \cdot 9K\ Cloned + 6K\ Unique)\frac{SLOC}{Yr} = 251K\frac{SLOC}{Yr}$$

$$S_{Test}^{Ind} = \frac{p_{Dev}^{Ind}}{p_{Test}^{Ind}} \cdot \frac{251K\ SLOC}{Yr} \cdot \frac{6\ Hr}{SLOC} \cdot \frac{1\ Person - Yr}{1680\ Hr} \cong \frac{p_{Dev}^{Ind}}{p_{Test}^{Ind}} \cdot 896\ Persons$$

$$\frac{p_{Dev}^{Ind}}{p_{Test}^{Ind}} = \frac{1}{4} \quad \Rightarrow \quad S_{Test}^{Ind} \cong 224\ Persons$$

Likewise the Independent testing demand can be computed from the information provided in the case study. Productivity numbers for the testing function are not provided, therefore we introduce a parameter which is the ratio of development productivity to test productivity. We assume a value of ¼ for that ratio (based on past experience and industry statistics). This gives us the Independent staffing requirement.

Question 2 . . .

Test (cont'd)

- Commonality: $\omega_{Test} = \frac{(19K\ Common + 9K\ Common)}{(19K\ Common + 9K\ Common + 6K\ Unique)} = \frac{28}{34} \cong .82$
- Leverage: $\lambda_{Test} = \frac{\left(19K \cdot 1 + 9K \cdot \frac{4}{11}\right)}{(19K + 9K)} \cong .80$
- Productivity:
  - Assume δp = -1/2 (Very conservative)

$$S_{Test}^{SPL} = \left(\frac{1}{1+\delta p_{Test}}\right)\left(\frac{1}{1+(N\lambda_{Test}-1)\omega_{Test}}\right) S_{Test}^{Ind} = \left(\frac{1}{1-\frac{1}{2}}\right)\left(\frac{1}{1+(11\cdot.80-1)\cdot.82}\right) 224 \cong 60$$

$$\Delta S_{Test} \equiv S_{Test}^{Ind} - S_{Test}^{SPL} \cong 164\ Persons$$

Using the information provided in the Air-Bits case study, it's a straight-forward calculation to derive the SPL testing commonality and leverage parameter values.

Assuming there is a 100% overhead in testing for reuse (fairly conservative assumption), we obtain the SPL test staffing requirement and test staff savings.

Question 2 . . .

Summary

$$S^{Ind} = 146 + 224 = 370$$

$$S^{SPL} = 50 + 60 = 110$$

$$\Delta S \equiv S^{Ind} - S^{SPL} = 370 - 110 = 260$$

Question 2 . . .

Should Air-Bits transition to SPL?

Approach*

- E_T total effort (person-years) to make the transition
- T_T transition interval (years)
- L fully loaded cost per person per year
- C total transition cost $C = L \cdot E_T$
- T_P planning horizon (years)
- B total benefits $B = L \cdot \Delta S \cdot (T_P - T_T)$

* For simplicity, assume non-discounted cash flows

The cost benefit analysis should include discounted cash flows, however, for simplicity we have ignored the discounting (if it doesn't prove in without discounting, it won't prove in with discounting since costs are incurred prior to benefits).

Question 2 . . .

Yes, if… $B - C > 0$

Or … $T_P > T_T + \frac{E_T}{\Delta S}$

Assume

- Transition effort is dominated by developing/re-engineering asset base
- Code base commonality: ω = .95
- Size of code base: 1200 Modules x 500 SLOC/Module = 600K SLOC

$$E_T = \frac{600K\ SLOC}{p_T}$$

- Transition productivity: p_T = 1.5FP/PM @ 100SLOC/FP => 1800 SLOC/PY

$$E_T = \frac{600K\ SLOC}{1800\ SLOC/PY} \cong 333PY$$

While there are organizational costs to consider in the transition, the assumption is that those costs are a second order effect due to the size of the redevelopment/reengineering effort.

The size of the new code base is 600K SLOC. An assumption of 1800 SLOC/PY to develop that code base is conservative, even if we assume there is a 100% overhead in developing for reuse.

Question 2 . . .

So …

$$T_P > T_T + \frac{333}{260} \cong T_T + 1.3$$

For T_T = 3 Yrs, T_P = 4.3 Yrs –high risk investment

However, if cut total time and effort by 50%, T_P = 2.1 Yrs

- E.g. mine existing assets to decrease time and cost

Many assumptions

MUST DO SENSITIVITY ANALYSIS

We had to make a number of assumptions to arrive at the payback point.

The next logical step is to perform “what if” analysis by changing the assumptions and determining how sensitive the answer is to those changes. Parameters for which a small change in value yields a significantly different answer are of the most interest.

Question 3

Independent scenario

- Assume option b from Question 1 for code modification, add testing

$$E_{Dev}^{Ind} = 65K \cdot Hrs \frac{1\,PY}{1680\,Hrs} \cong 39\,PY$$

$$E_{Test}^{Ind} = \frac{p_{Dev}^{Ind}}{p_{Test}^{Ind}} \cdot (1+\alpha_{\mathrm{Re}gression}) \cdot 39\,PY \cong 20\,PY \quad (\alpha_{\mathrm{Re}gression} = 1)$$

$$E^{Ind} \cong 59\,PY$$

SPL scenario

Assume ω = .95

$$E^{SPL} = \frac{(1-\omega)\cdot(1+\alpha_{\mathrm{Re}gression})}{p^{SPL}} 600K\ SLOC \cong 33\,PY$$

Savings

$$\Delta E = E^{Ind} - E^{SPL} \cong 26\,PY$$

The commonality figure of .95 refers to the commonality across the entire code base, and is based on the information provided in the Air-Bits case study.

SOCOMO-PLE, The 4C Model: Application to AirBits Inc.

Lamia Labed* and Sana Ben Abdallah*
University of TUNISIA, RIADI-GDL laboratory (ENSI)
Ali Mili*, NJIT, College of Computer Sciences, University Heights

SPLC Panel,
Rennes, September 28, 2005

*Lamia.Labed@isg.rnu.tn, *sana.benabdallah@riadi.rnu.tn,* mili@cis.njit.edu

CONVERGYS

Background: Premises of 4C

- Four main stakeholders: Corporate Management, Domain Engineering Team, Application Engineering Team, Component Engineering Team.
- Each Stakeholder has a strategic decision to make: Introduce Reuse, Launch a Product Line, Develop an Application, Develop a Component.

Premises of 4C

- All Stakeholder decisions may be quantified by ROI formulas.
- Making Reuse Happen: Ensuring all ROI are positive.
- Better: Optimizing Corporate ROI under the condition, all ROI are positive.
- Linear Optimization.

CONVERGYS

Question #1

- In the 4C model, gains in quality are quantified by reduced maintenance costs, exactly the question asked here.
- The question is ‘’should they’’: The answer depends on who ‘’they’’ are (corporate management, AE team). We assume AE.
- 4C quantifies AE costs using three parameters: black box reuse, white box reuse, and custom code.
- 4C also quantifies library operations.

4C's Interpretation of Options AE cycle

- Option (a)
- Black box reuse: 70%;
- white box reuse: 30%.
- We neglect effort to identify best fit.

- Option (b)
- Black box reuse: 70%
- White box reuse: 25%
- Custom code: 5%.
- Library search: 1000 searches.

ROIa for Option (a)

- Investment Cost = IC = Cα(2005) = 700 * BP + 300 * WP = 448,5 PM
- Episodic Benefice = Bα(2005) = DCA + SCA = 104 604 PM

- DCA : Development Cost Avoidance = 1 104 PM
- SCA : Service Cost Avoidance (cost of avoided maintenance by reusing a
- component) = 103 500 PM (assuming an error rate = 1,5 and error cost =
- 100 times of the development cost).
- NPV = Bα(2005) - Cα(2005) = 104155,5 (just for the current year)
- ROIa = NPV / IC = 232,230

ROIb for Option (b)

•Investment Cost = IC = Cα(2005) = 700 * BP + 250 * WP = 438,15 PM

•Episodic Benefice = Bα(2005) = DCA + SCA = 99373,8 PM

DCA : Development Cost Avoidance = 1048,8 PM

SCA : Service Cost Avoidance (cost of avoided maintenance by reusing a component) = 98 325 PM (assuming an error rate = 1,5 and error cost = 100 times of the development cost).

NPV = Bα(2005) - Cα(2005) = 98935,65 (just for the current year)

ROIb = NPV / IC = 225,803

Results

- ROIa > ROIb so,
- Option (a) is better in our application cycle.

CONVERGYS

Question #2

- Here we must take the standpoint of corporate management, as the re-engineering decision, and costs, are up to them.
- All the ROI's are subject to two strategic parameters: discount rate (d) and investment cycle Y.

Question #2

- We let $ROI_A(Y)$ and $ROI_B(Y)$ be the corporate ROI's under options (a) and (b).
- Question #2 can then be formulated as: what is the smallest Y such that
  - $RE+ROI_B(Y) \leq ROI_A(Y)$,
- where RE are the re-engineering costs.

Question #2, Results

- We need more details about SD which is Start Date of the PLE initiative and other information about certification and library insertion, architecture evolution cost, domain analysis cost, infrastructure investment, number of librarian for managing the repository, etc.
- We don't have an equation for Reverse Engineering cost. If we consider it as maintenance described in the scenario, we evaluate 34 KSLOC to be developed (it is long to give the details), 204 staff-hours.

Question #3

- The 4C model has equations for estimating the cost of an application as part of computing the AE's ROI.
- We estimate it under the (b) option.

Question #3, answer

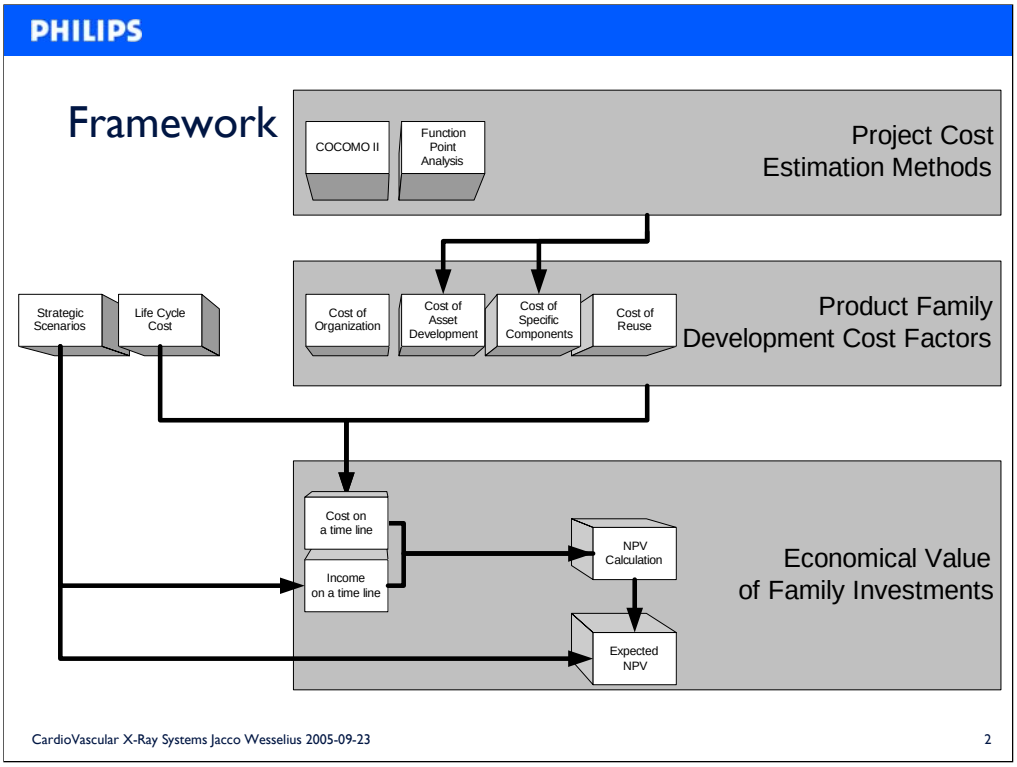
- Solution in slide 7 : cost = 438,15 PM

PHILIPS

Panel:
A Competition of SPL Economic Models

Jacco Wesselius (Jacco.Wesselius@philips.com)
CardioVascular X-Ray Systems
Panel Discussion SPLCe2005 (Sept. 28, 2005)/Rennes (France)
2005-09-23

CONVERGYS



The economical model presented in our SPLC2005 paper does focuses on translating expected cash flow into an estimate of the expected economical value.
As indicated in this picture, we identified three types of economical models:

1) Models like COCOMO and Function Point Analysis → effort estimates for defined tasks based on historical data
2) Models like those presented by Dale Peterson, and Klaus Schmid/John McGregor focusing on estimating the re-use costs/benefits involved in product line development (I was not aware of the model of Lamia Labed and Sana Ben Abdallah while preparing for this SPLC2005-panel discussion)
1) Models like ours which use these cost/benefit estimation models to estimate the economical value.

Our model adds three aspects:
1) All costs estimated using the other models should be projected in time: cash flow discounting (e.g. using NPV) is essential
2) Apart from development costs, other costs should be taken into account: life cycle costs
3) Since the future is uncertain, future cash flows are uncertain → capture this using strategic scenarios

This picture was included in my SPLC2005-talk, but it was not in the (short) paper. It was made after submission of the paper and will be part of a longer paper, which will probably be a chapter of the third ITEA-families project book.

PHILIPS

Scenarios

- Architectural Scenarios
  - A series of investments in product line assets
  - The investments are placed in time
- Strategic Scenarios
  - A series of events that influence the value of the investments
    - New or change market demands
    - Initial delays caused by investments in the architecture
    - Changes to product/feature/quality value
    - Emerging new technologies
    - Organizational changes
    - *etc.*
  - The events are placed in time
- For each architectural scenario, the value depends on the occurrence of future events

See my paper

CONVERGYS

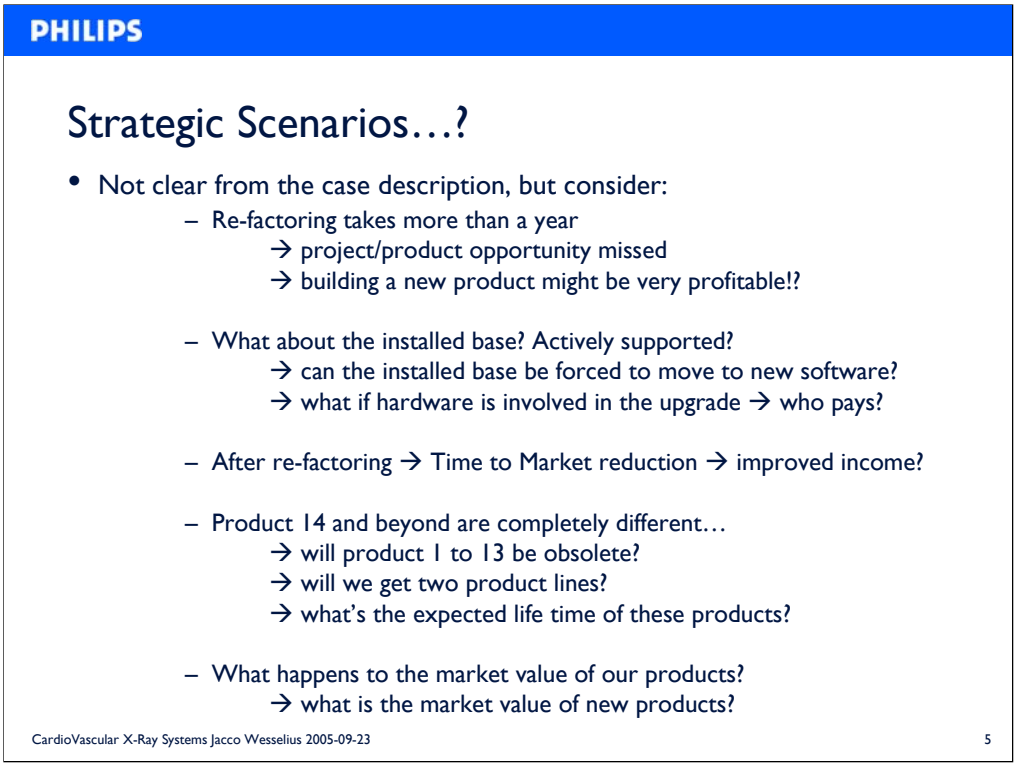
PHILIPS

Architectural Scenarios

- Do re-factor the 4000 modules
- Don't re-factor the 4000 modules

- The associated cost → use existing cost models
- But take care: put the cost in time → NPV

$$NPV = \frac{cashflow}{(1+discountRate)^{time}}$$



This slide contains a brief overview of aspects that were not considered in the case description.
Many sources of future cost/income should be considered before judging the economical value of the product line transition.

The case description provides data for estimating the cost saving based on a transition to product line development.
As such, it provides the data for an attempt to apply the second type of modeling (see my first slide).
Applying the first type of modeling is not needed, since all the basic effort estimates have been provided in the case description.

An attempt to apply my model resulting in repeating the exercise Dale Peterson and John McGregor did. Repeating that would not have made any sense.
To make the next step (cost estimates → economical value estimates) would have required much more data.
To give an idea of the aspects that came to mind when considering the case description, this slide provides an overview of some building blocks for strategic scenarios which would have a major impact on the overall expected value estimation.

As a consequence of this, I did not present an answer to the questions in the case description.

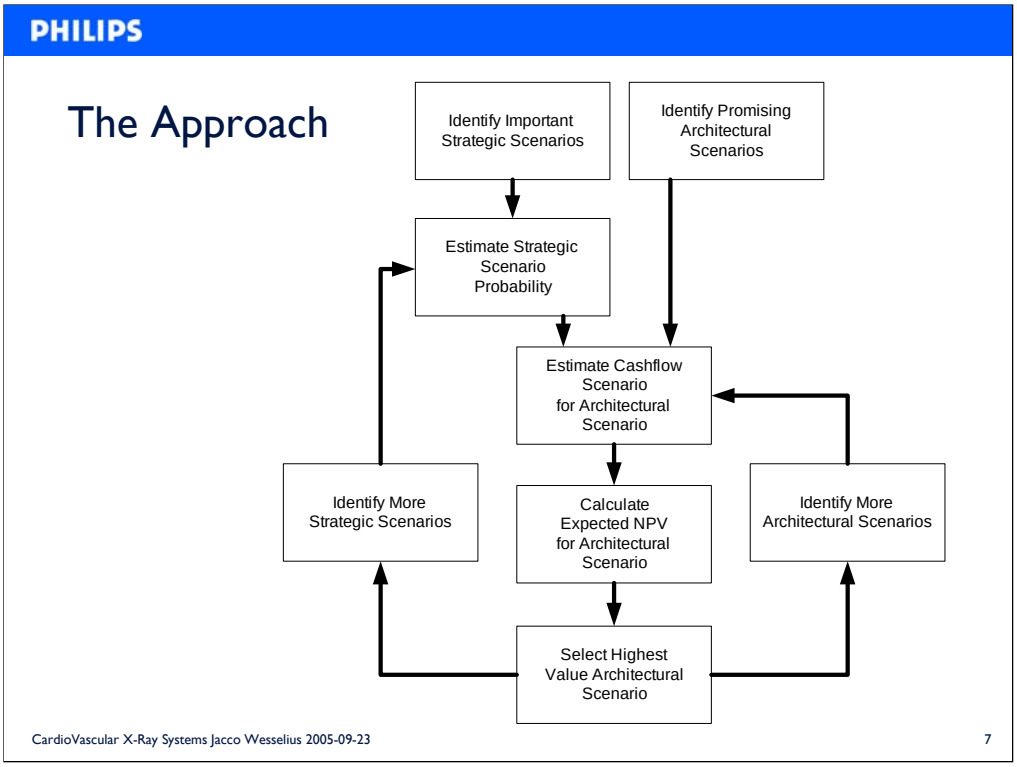
PHILIPS

Expected NPV != NPV

$$NPV_{Expected}(ArchScenario, StratScenario[1..n]) = \sum_{i=1}^{n} NPV(ArchScenario, StratScenario[i]) * probability(StratScenario[i])$$

See my paper

CONVERGYS



See my paper


PHILIPS
http://home.planet.nl/~jacco.wesselius/SPLCe2005slides costmodels.ppt
http://home.planet.nl/~jacco.wesselius/SPLCe2005paper.pdf

A Comprehensive Modelling Approach for Product Line Economics

Fraunhofer

Institut
Experimentelles
Software Engineering

Klaus Schmid
klaus.schmid@iese.fraunhofer.de

CONVERGYS

Key Characteristics of the Modelling Approach (I)

- Open model aiming at introducing all kinds of economic issues into a model
  - **Level I:** basic qualities: cost, time, defects, etc.
    user perception (market value)
  - **Level II:** Discounting (time-value of economic attributes)
  - **Level III:** Value of alternatives, flexibilities, etc. (options)

- *In our example: only costs are used!!*

Key Characteristics of the Modelling Approach (II)

- Economic Modelling is CONTEXT-Dependent!
  - Strong reliance on a GQM-like approach
    (what is relevant in a specific context? How do costs interact?)
  - Reliance on successive refinement of basic values and formulas
    (e.g., cost of reuse)

IESE

Question I

- Goal Formulation
  - **Minimize** the **costs** for **Air-Bits Avionics Product Line** from the viewpoint of **product line management** at **Air-Bits, Inc**.
- Refinement
  - **costs:** maintenance costs + development costs
  - Scenarios: clone and own single product vs. best-fit
- Result
  - development costs are significantly lower for best-fit (~6 times)
    reduction in maintenance costs (depending on how handled)
    improve this advantage

CONVERGYS

Question I

- Dev. Effort per Line: effL 6*(1+2)*0,75 = 13,5 PD
- Dev. Effort per module: effM = 500 * effL = 6750 PD

- **Cost for development in Clone-and-own scenario:**
- Identify most similar product (incl. 700 modules): 1day
  Reprogram 300 modules: = 300*effM = 2.025.000 PD
- Teff= 2.025.001 PD

- **Cost for Development in clone as much as possible scenario:**
- Identify most similar product to get 700 modules: 1 day
- Identify 250 further modules from other products: 1 day per module = 250 PD
- Reprogram 50 modules: = 50 *effM = 337500 PD
- Teff = 337801 PD

- Maintenance (if new modules are cloned = roughly identical)

- *=> Searching longer wins*

Question II / III

- Result – Question II
  - What: Recovery of reengineering will take roughly > 60 years!
  - Why:
    - extreme numbers in terms of reengineering effort
    - comparably low savings in testing only
    - only reduction in clone copies considered
      (e.g., reductions in defects, etc. are not taken into account)
- Result – Question III
  - What: Clear advantage in new development based on reengineered approach!
  - Why:
    - extremely low number of modules that need to be coded anew
    - productivity is extremely low

IESE

Conclusion

- Our model is more a framework that provides a goal-oriented way to analyze a situation
  - model leads to many questions, supporting the understanding of the situation, e.g., the goal
- No “pre-given” solutions – thus, no preconceptions that do not fit to the situation -> However increases effort required for model development and tuning
- Supports a wide range of characteristics that were not an issue here (e.g., time-value, user-utility, flexibility, etc.)

CONVERGYS

The Structured Intuitive Model for Product Line Economics (SIMPLE)

John D. McGregor
Clemson University
SEI

https://simple.sei.cmu.edu

SIMPLE

- 2 cost functions with product line scope
- 2 cost functions with product scope
- A set of benefit functions

$$C_{org}(t) + C_{cab}(t) + \sum_{i=1}^{n}(C_{unique}(product_i, t) + C_{reuse}(product_i, t)) + \sum_{j=1}^{nbrBen} B_{ben_j}(t)$$

The guiding principle for SIMPLE is to be simple. We do that by separating the specification of the cost functions from their implementation.

Product line costs

- C_{org}
  - The costs of modifying an organization for product line practice
- C_{cab}
  - The costs of creating the core assets

Product costs

- C_{unique}
  - The cost of building the unique portion of a product
- C_{reuse}
  - The cost of locating and readying for use the required core assets

Benefits

- We have not enumerated a set of benefits as we have for costs
- A benefit must be chosen so that “double counting” does not occur.
- In most uses of SIMPLE a cost reduction would not qualify as a benefit.
- An increase in quality would.

Uncertainty

- An analysis using SIMPLE handles uncertainty by considering multiple scenarios
- Each scenario proposes a different set of events
- The analyst can present a range of results where the probability that the scenario will occur relates directly to which scenario should guide action.

Time

- Time is handled in SIMPLE using whatever mechanism the modeler wishes.
- Some companies would use Net Present Value but others use other measures
- The analyst provides a cost function for each cost under the broad categories.
- For example, in C_{org} the analyst might include a $C_{training}$ with a series of values that increase by 5% each year to account for price increases from the vendor.

The challenge

- Train A leaves New York at 2 pm traveling west at 30 mph while train B leaves Chicago at 3 pm traveling east at 40 mph. At what time will there be a really big wreck?

Question 1

- Given a goal of reducing maintenance costs, should they:
  - clone-and-own the most similar system, and expect to re-program 300 of that product's modules, as they have in the past? Or,
  - comb their repository and look for the best fit for all 1,000 modules that will populate the new system? (They will start with a similar system first, giving them the first 700 modules of their new system right away. They will search the repository looking for the best fit for the other 300 modules.) Here, the expectation is that they would only have to re-code 50 modules, being able to find the other 250 from *some* other product in their family.

Question 1

- C_{org} is assumed to be zero since a product line organization exists
- C_{cab} is irrelevant since it is constant across both options
- For the Clone and Own option
  - C_{unique} cost function – 900000 hours (300 modules * 3000 hours/module)
  - C_{reuse} cost function – 0 hours
- For the Search and Find option
  - C_{unique} cost function — 150000 hours (50 modules * 3000 hours/module
  - C_{reuse} cost function – 112500 hours (250 modules *(.15 * 3000 hours/module))
- Option 1 requires 900000 hours
- Option 2 requires 262500 hours

.15 is a very conservative estimate of the cost of locating the 250 modules

Question 2

- How long, if ever, will it take to recoup the re-engineering effort simply based on the difference in testing and maintenance cost between the two approaches? Assume every product comes forth with a new version every year.

Question 2

- C_{org} is assumed to be zero since a product line organization exists
- C_{unique} and C_{reuse} can be ignored to answer the question since we are limited to test and maintenance costs; however since the smaller asset base would be easier to use C_{reuse} would be less for the reengineered core asset base.
- C_{cab} is broken down into three categories for this question: C_{test}, $C_{maintenance}$, and $C_{reengineering}$

Question 2 - 2

- Compute $C_{reengineering}$
  - Worst case for $C_{reengineering}$ is 3,600,000 hours + regression testing costs for 11 products (Assumes writing all 1200 modules from scratch)
  - Best case, ALL products share the 700 modules and the rest of the core asset base is shrunk into 500 new modules at a cost of 500* 3000 hours == 1,500,000 hours
- Compute test and maintenance costs under current and reengineered asset bases
  - Assumption – same percentage of new asset base is touched by maintenance in both approaches so the reengineered scenario affects 360 modules
  - There is a savings of 800 – 360 modules per year = 440 modules in the reengineered option

Question 2 - 3

- Compute savings and divide $C_{reengineering}$ by the amount of savings/year to get years to payoff.
  - Using just the basic maintenance and feature costs and assuming that the smaller core asset base would lower maintenance costs the savings is 121500 – 36450 = 85050 hours/year in savings for $C_{maintenance}$
  - Based only on worst case reengineering and maintenance savings the reengineering pays off in only 42 person years! For best case, 17.6 years

Question 3

- What will it cost to add a twelfth product to the family if (a) the engineers get their way? and if (b) the product line continues down the current clone-and-own path?

Question 3

- C_{org} is assumed to be zero since a product line organization exists
- C_{cab} is ignored since it is assumed to exist in either form
- Option (b) is the same as in Question 1 == 900000 hours
- Option (a) can assume that C_{reuse} is reduced to .05 == 187500 hours

References

- "Computing Return on Investment for Software Product Lines" Guenter Boeckle, Paul Clements, John McGregor, Dirk Muthig and Klaus Schmid, IEEE software, v 21, n 3, May/June 2004.
- **"The Structured Intuitive Model for Product Line Economics (SIMPLE)"** Paul C. Clements, John D. McGregor, Sholom G. Cohen, CMU/SEI-2005-TR-003, 2005.

Carnegie Mellon
Software Engineering Institute

Results

	Q1 A: clone B: mine	Q2 Payback time	Q3 Cost of P12
Dale	B	2.1-4 yrs.	26PY
Sana	A	?	438PM = ~36PY
Jacco	?	?	?
Klaus	B (x6)	>60 yrs	316kPD = ~158PY
John	B (x3.4)	17.6-42yrs	187500PH = ~94PY

CONVERGYS

Carnegie Mellon
Software Engineering Institute

Conclusions

Product line economic modeling is a journey, not a destination…

CONVERGYS