Six Easy Software Pieces

Some Industrial Challenges in Software Variability

What are the challenges?

What is the domain?

What is the “product”?

What are the variations?

What are we using?

6 Easy Software Pieces

Wrap-up

What's the Medical Domain?

The medical domain focuses on products that keep people alive:

- Drugs
- Devices – pacemakers
- Treatment – monitors, drug delivery systems
- Doctors

In order to be cost effective the whole system needs to be managed:

- Electronic health records
- Treatment information
- Drug control
- Scheduling systems

The medical domain is both tightly controlled and very conservative.

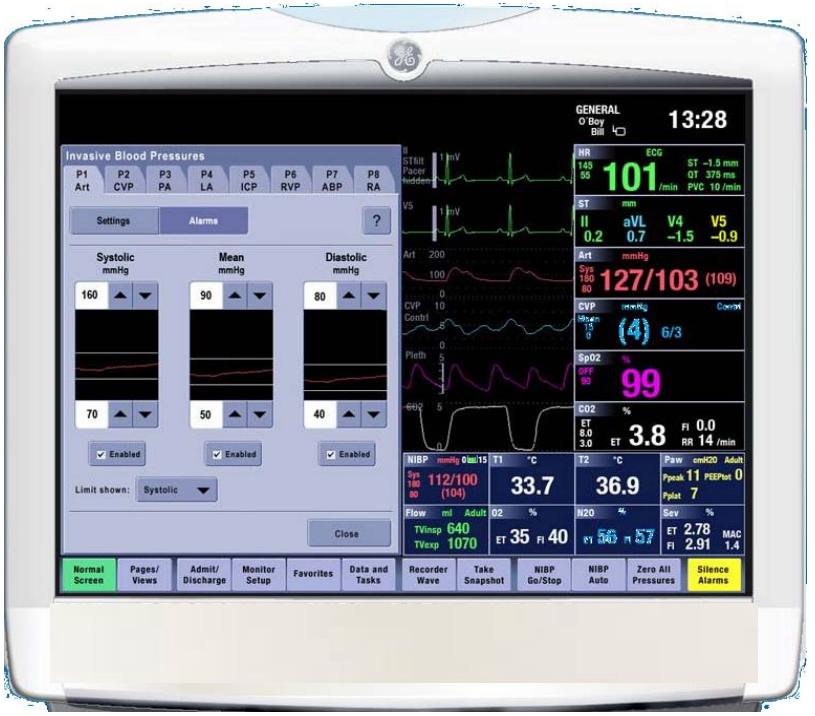
What's the Patient Monitoring Domain?

A standard set of features:

- ECG, Temp, CO2, SpO2, ImpResp, SvO2 etc.
- Variations according to context – OR, ICU, NICU, etc.

Market is split along cost lines

- Low-end, Value, High-end



A good candidate for a scalable product line, since the physiological parameters do not change (much)

What's the GE Product Domain?

GE Produces:

- Back end systems
- Delivery systems
- Monitoring systems – low to high end.
- Diagnostic systems – MRI, Ultrasound

In each market

- Total cost of ownership is the key
- Quality is crucial
- Service is important
- Products have long lifecycles

For care related systems regulatory approval is required – globally.

What are the domain's Product dimensions?

Variable Cost:

Scalability means changing

- Hardware – Flexibility and performance
- UI technology – Devices and cost

Fixed Cost:

Scalability means changing

- Context features – Hi-Resolution trends
- Parameter sets - OR vs. ICU
- Options

Quality is controlled, changes occur at different points in the product lifecycle.

What does Product management need?

Product managers need to understand where the software is coming from, meaning

- Clear policies – how their product relates to the product-line is need
- Clear variations – they need to see their variations quickly

Division managers need to understand how the product line evolves and why, meaning

- How adding/removing variation helps product managers
- How product-line changes makes money

Products pay for software, showing how software variability helps the products is a key activity

What's our Current state-of-the-art?

Current tools have not varied much since the 1980's:

- Software configuration management - ClearCase
- Compilers – in our case C++
- Makefiles – ClearCase
- Requirements management - DOORS

Design methods have had an impact

- OO design - UML
- OO Frameworks
- Architecture driven development

Processes have changed

- Agile processes are now used.

This is a standard set of software tools, supporting standard product development

I: Merge variability management methods with industrial tools

Industrial developers use industrial tools

- For companies to integrate variability management then it needs to work with industrial tools

The basic tools exist, but need some variation management spin

- OO Design – aggregation/inheritance/parameterisation
- Configuration tools – branch variations
- Compiler – conditional compilation
- Make systems – conditional linking
- Distribution– system configuration
- Configuration – run-time parameterisation

II: Write a Bestseller!

Industrial developers need guidance

- Provide a book of best practices
- All link to the book

A list of methods and when they should be used:

- Trade-offs between variation mechanisms
- Relation to other SW dev.
- Merging of methodologies

Some adoption plans and surveys exist, but an objective view of advantages and disadvantages of methods and when/how to use it is needed.

III: Parallel development of product-line and variations

Methods/patterns of coordinating changes to variations and the common product-lines are needed.

- Every variation needs to be maintained and evolves with the products.
- Every variation evolves with the product-line.
- So Product-line software development is like hitting a range of moving targets from a moving platform

The tools for handling variation exist, they need to be expanded to handle variations of variations.

IV: Variable variation points

Software is not fixed, so why are variations points?

- Industry needs ways to insert and remove variations points easily.
- (Tool integration would be nice too)

The tool:

- Insert a variation point – compiler or makefile directive
- Delete a variation point – remove a directive and associated files
- Diff the variations – compare and merge variations

The concept of anytime variability exists, managing mechanisms are needed

V: Verifying the Product-line

Industrial verification of a single product is difficult:

- Every feature needs to be verified
- Every product variation needs to be verified
- Features need to be traced to code

Tools for reusing

- Software verification plans
- Software tests
- Recording verification
- Generating verification plans
- Tracing verification to requirements to code

Single product tools exist, they need to be extended for variation management

VI: Matching Software Management with Product Management

Features are developed for products, so for every feature we have the following questions:

- Is it part of the product-line or one-off?
- Is this a variation or a common feature?

- When to freeze for products?
- Release management for multiple product simultaneously?
- What products are possible in the current range?
- Balancing product development and product-line development

Solutions for single products over time exist, multiple product support is need.

Wrap-up

Industrial development need

- Industrial product-line methods
- Captured best practices

Industrial development needs

- Verification techniques for software variations
- Methods to manage product in parallel with software

Industrial development needs

- Variability point management
- Methods to management parallel development of the product-line and products


Q&A