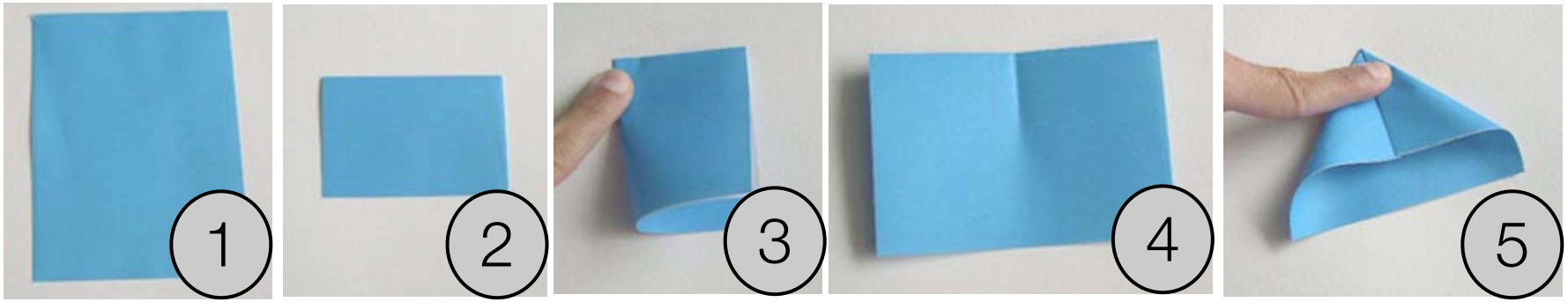
Configuration Languages?
Find out what, why and how in 5 steps.

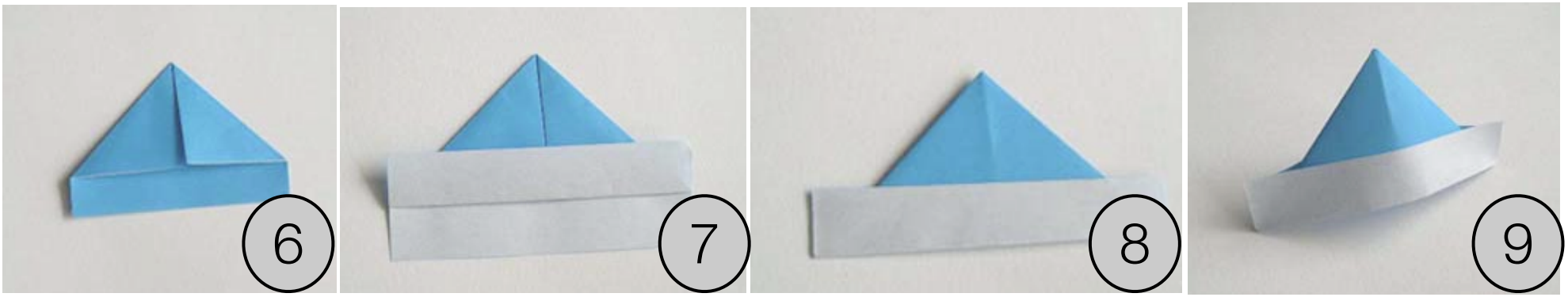
Dr. Thomas Cleenewerck
Dr.-Ing. Sebastian Günther

Vrije Universiteit Brussel
VariBru

The power of configuration languages

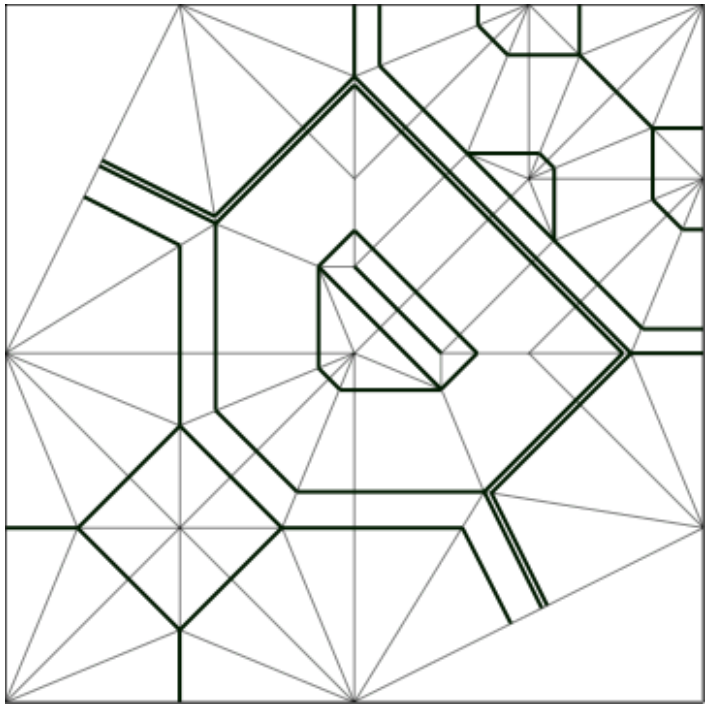
Evolving from basic instructions..



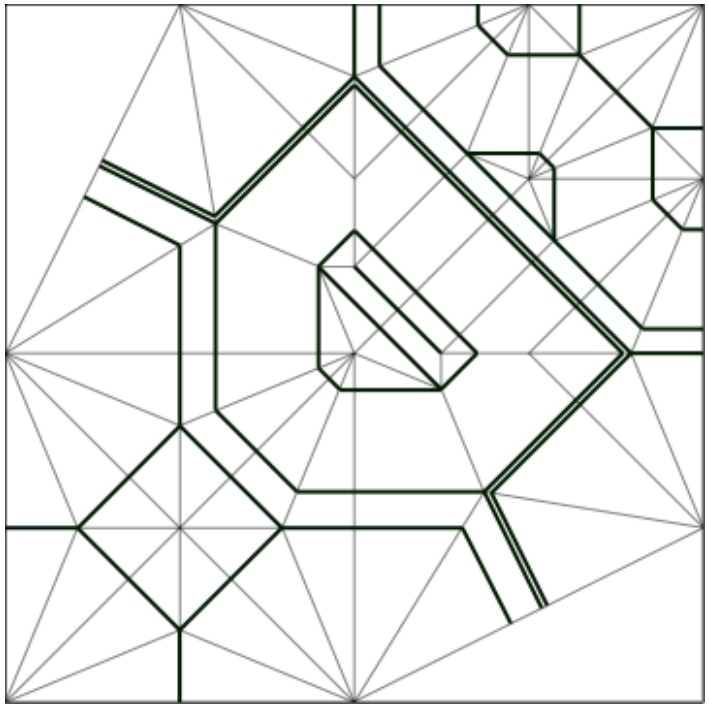


Illustrated origami instructions www.origami-instructions.com

To a powerful notation

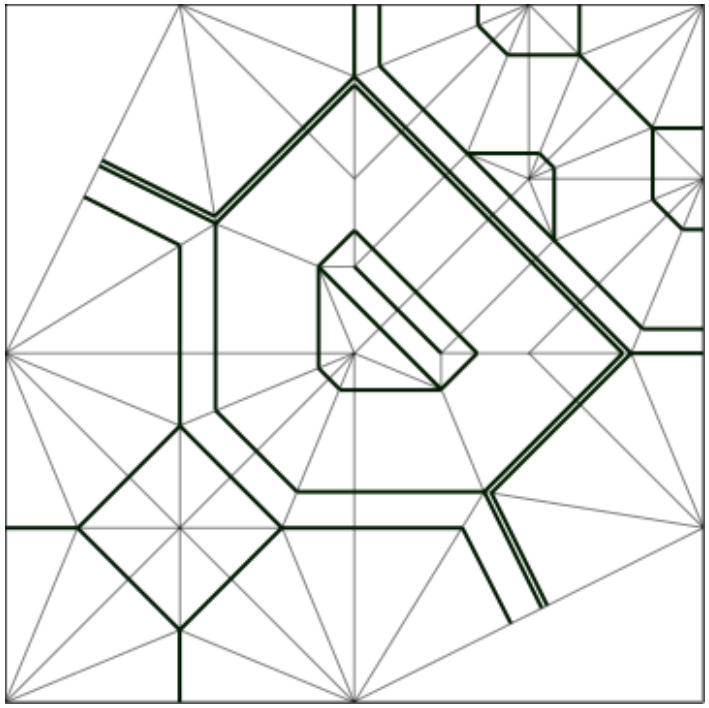


Fold any figure with paper using Crease Pattern



Fold any figure with paper using Crease Pattern

It has become a **language** to **configure** a sheet of paper to any shape



Fold any figure with paper using Crease Pattern

Applying this power in software
via a configuration language
yields...

pricing rules:

totalprice += 23 * workshop1

(bad ∈ articles) totalprice -= bad.price/2

email rules:

subject = '*varibru*' => inbox

...

new customer-specific functionality

...

unit conversions:

m -> km : x/1000

km -> m : x*1000

report data:

event:group:concert event:locaton:balzaal

Step 1:

Turn your configuration challenge into an opportunity

cost reduction

"how to reduce costly in-depth programmer expertise"

cost reduction

"how to reduce costly in-depth programmer expertise"

better products

"how to offer better products"

cost reduction

"how to reduce costly in-depth programmer expertise"

better products

"how to offer better products"

faster delivery

"how to avoid long development cycles"

cost reduction

"how to reduce costly in-depth programmer expertise"

better products

"how to offer better products"

faster delivery

"how to avoid long development cycles"

grow by serving more customers

"how to scale your development team"

cost reduction

"how to reduce costly in-depth programmer expertise"

better products

"how to offer better products"

faster delivery

"how to avoid long development cycles"

grow by serving more customers

"how to scale your development team"

improve market position by stronger customer intimacy

"how to provide customer-specific functionality"

cost reduction

"how to reduce costly in-depth programmer expertise"

better products

"how to offer better products"

faster delivery

"how to avoid long development cycles"

grow by serving more customers

"how to scale your development team"

improve market position by stronger customer intimacy

"how to provide customer-specific functionality"

increase market share by distribution channels

"how to outsource customer tailoring"

Step 2:
Identify configuration problems in your product

adequate parameters?

varying antenna length:
cleary this antenna does not fit
customers needs

adequate user interface?

time consuming
hard to get it right

adequate process?

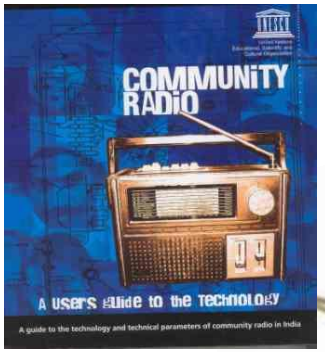
complex rebuilds that are
easy to corrupt

parameters are inadequate

skins, accessories

predefined functionality

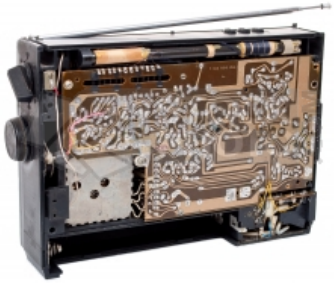
user interface is inadequate



step by step manual

non reproducable

process is inadequate



technical no business

laws required to compose parts

Step 3:
Learn why **not** to vary radios
like we vary software...

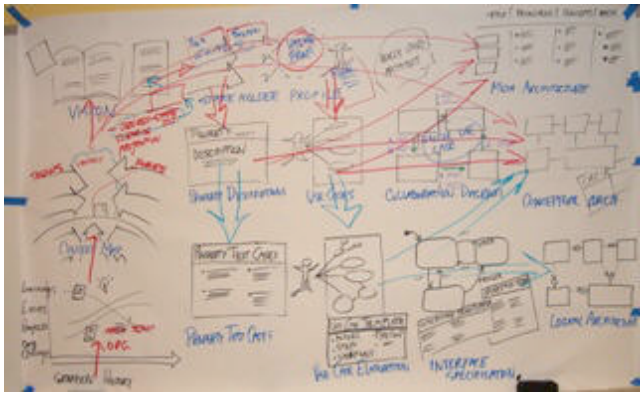
adequate parameters?

the software does not fit
customers needs

adequate user interface?

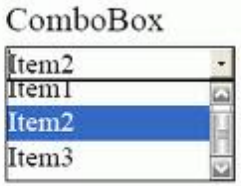
time consuming
hard to get it right

adequate process?



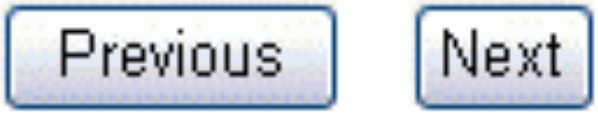
rebuild process is complex
and easy to corrupt

parameters are inadequate



parameters, yes-know

user interface is inadequate

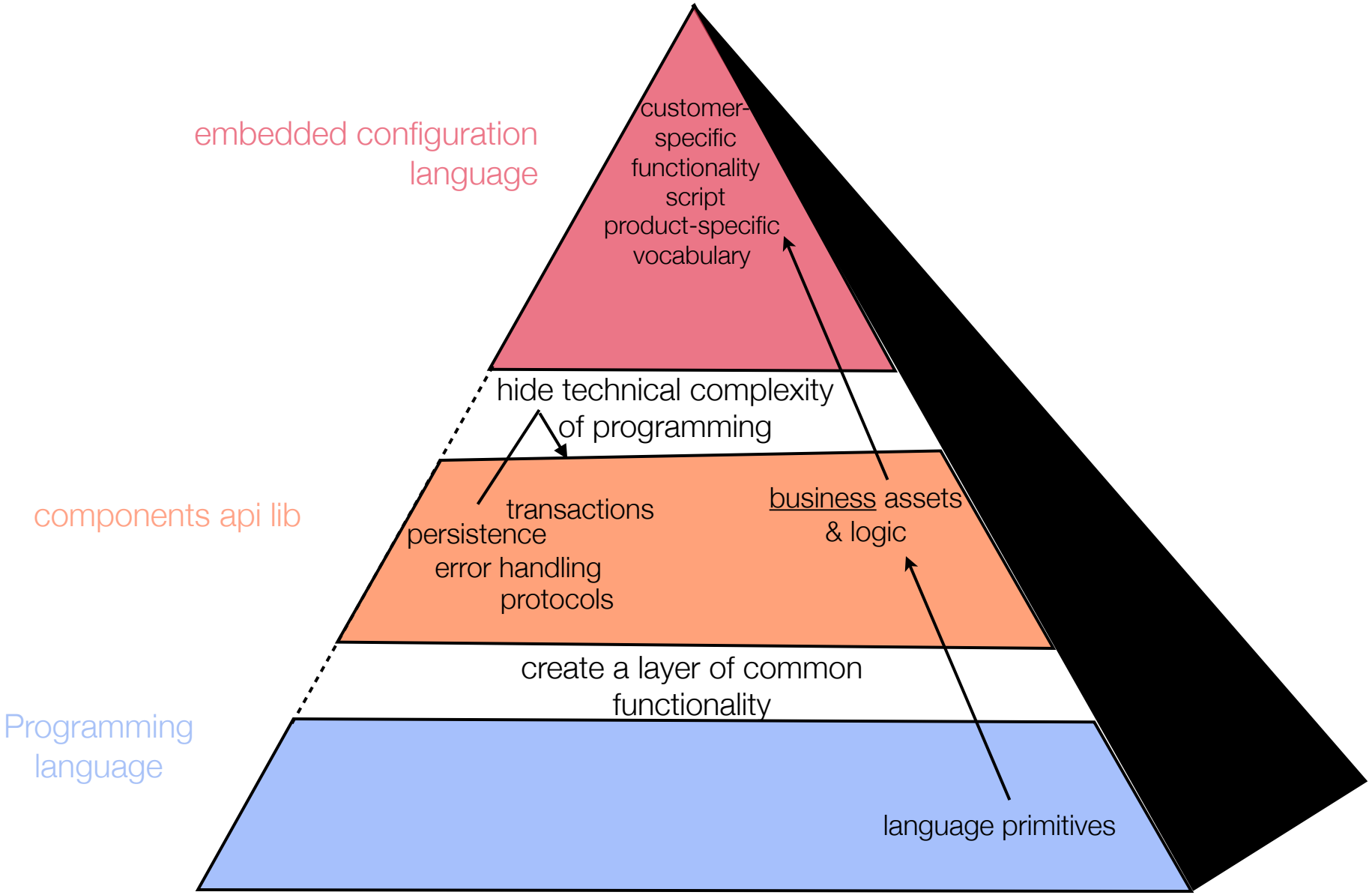


step by step
wizard

process is inadequate

components, libraries, ...

Step 4:
How to vary with configuration languages


embedded configuration language
customer-specific functionality script product-specific vocabulary
hide technical complexity of programming
components api lib
transactions persistence error handling protocols
business assets & logic
create a layer of common functionality
Programming language
language primitives

Step 5:
Enjoy the value of
configuration languages

high cost

"in-depth programming expertise is not cheap"

cost reduction

"no longer require in-depth programming expertise"

cost reduction

"no longer require in-depth programming expertise"

rigid products

"product is not flexible"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

slow delivery

"requests for custom functionality is a time consuming business"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

cannot easily serve new customers

"every customer requires detailed HR planning"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

grow by serving more customers

"no need to scale the whole team"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

grow by serving more customers

"no need to scale the whole team"

loosing market position by weak customer intimacy

"customers favor a more flexible and tailored solution"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

grow by serving more customers

"no need to scale the whole team"

improve market position by stronger customer intimacy

"ship new behavior and respond to specific requests"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

grow by serving more customers

"no need to scale the whole team"

improve market position by stronger customer intimacy

"ship new behavior and respond to specific requests"

no partner to increase market share

"every new customers requires your time"

cost reduction

"no longer require in-depth programming expertise"

better products

"offer richer functionality"

faster delivery

"configurations are concise and easy to write and adapt"

grow by serving more customers

"no need to scale the whole team"

improve market position by stronger customer intimacy

"ship new behavior and respond to specific requests"

increase market share by distribution channels

"subcontract integrators to serve customers"

Summary

what: A thin layer tailored on top of your business assets to script customer-specific functionality using your product-specific vocabulary. It hides the complexity of programming.

how: Embed in your programming language a new language abstraction layer on top of your business assets. It's cheap, light weight and requires no special programming skills.

benefit: Adapt your product without in-depth technical expertise. It's quick, less error-prone, easy to write and easy to change.

dr. Thomas Cleenewerck
tcleenew@vub.ac.be

Dr-Ing. Sebastian Günther
sgunther@vub.ac.be