SPA 2010 – End users as software engineers – good idea? Yes But, No But: RAW WRITE UP

Why are you here – scenarios attendees interested in:

- We develop tools (mechano kit) for other developers to build tools for other end users. “End-end-users” kit content.
- End User Systems – something built by end users that is to be “productionised” – an MI system developed by end-users / the business.

===

These Days Anyone Can Build An Application	
No	**No But**
They cannot build the application framework	..everyone can have a great idea for an application
Some people are too stupid/incompetent to code	..we shouldn’t assume they can’t understand technical issues
Some applications are limited in scope and interoperability	End users can customise and alter existing applications
The tools they use build the application	Some basic skills are required but it is not too hard to develop them
Security can become an issue	Although a lot of the barriers to software have been removed, some people lack the intrinsic ability to do it
Anyone can build a mess but an application needs to be properly crafted	
Many people do not have the motivation	
You need to be able to develop a complete development environment	
Language is too great a barrier	
Every “language” created for this has failed in some way historically	
Some languages / environs are just too darn tricky w/o training & experience	
Yes	**Yes But**
Kids are growing up more skilled and less technophobic	They need to have the motivation to deal with the technology
Free technology and free advice available to everyone	Users overwhelmed by choice of tools
Yes and people do it for fun	End users assume they cannot build applications
Availability of very good sample code	Only small application
Even a spreadsheet is an application and anyone can do that	That doesn’t make it good application (x2)
The tools are much more available to all	The application really has to want to be built
It’s not rocket science	If you assume that an application is any piece of running software
Support environments help and check code produced	It depends what you call an application
Online communities make it much easier	

Yes: Debate notes made:

- Empowerment - Lowering of barriers to entry
- Components are there, Knowledge is there, Communities are there
- Wide range of technologies for all sorts of development for fun or for business

Formal Software Development Education Is Necessary For End User Developers		
No		**No But**
You learn by doing (x2)		Experience vs Education – experience is more valuable
Why try to control it?		Software development literacy is desirable
Doesn’t it risk us limiting creativity by imposing standards / beliefs		It should be available to end users who wants some
Tools – drag and drop / metaphors		There may be costs to not doing so (reliability, maintainability)
It’s obviously NOT the case		
Most of what I know is self-taught		
If it works it works, if it ain’t broke don’t fix it (it being the end user software development process		
Yes		**Yes But**
It would increase their productivity		Education doesn’t support the necessary creativity
End users need aspirations		What is the benefit?
There are risks!		Why bother?
Education is necessary for everyone		Which Software development training is suitable for end users?
Yes for business applications – risk reduction		Only for critical or v complex applications
Education about risks of software development		Do we have suitable education
		I coded without training but I do better since training and mentoring

Brainstorming – why do they do it, why should we care.... Grouping the brainstorm...

Why they do it: Filling a Gap
To support specific scientific tasks
They see a market opportunity
I know my requirements and existing application doesn’t meet it
My needs are not met by IT department => DIY
Current software doesn’t meet needs
No specific apps for what they want to do
Why they do it: It’s FUN
It appears easier than doing something else
They’re techies at heart
Do end users see it as software? Or just something to get the job done
They care
Why they do it: Domain Knowledge / communicating requirements
Barrier to “having knowledge to be able to make the system” is too much for external developers
Communicating requirements is too difficult
Feel they understand more than an “outsider”
The ultimate “on site” customer
They don’t know what they want until they have written it
They don’t know what they want – try it out
Software that only works on bespoke hardware
Why they do it: IT not accessible / available
Can’t wait for IT department to do it
Can’t afford IT people to do it
No one else will do it for me x 3
I can’t get anyone in to do it
Why we care: IT Professionalism
Not tested thoroughly -> Risk of failure -> Risk of business impact
We care because we have professional responsibilities
We lose prestige
They might make bad code available to others

Why we care: Support and Maintenance
We care because we will have to make it REALLY work
Lack of IT support
We get asked to adopt and support their code
We care because we will end up maintaining it
We get called in when it falls over
How many end-users -> Impact and Support?
If original “developer” leaves, who looks after it?
Their software linked to & impacts on our systems

Why we care: IT Opportunity To Be GAINED
So we can produce software that facilitates it
There’s a revenue stream in correcting their mistakes

Why we care: IT Opportunity LOST
We lose business
Re-inventing the wheel
They buy our stuff (??)

Problem Tree: **Lack / Difficulty in Communication of Domain Knowledge**

Fear of appearing stupid	It's complex knowledge Understanding domain requires years of study I might be exposed as NOT having domain knowledge There is no "Domain" I don't really understand the domain (yet) Lack of formal education
Pain of making implicit knowledge explicit	Domain experts can't communicate with (normal!) people Unconscious processes I don't want to share knowledge I don't think people will be able to understand Prejudice Peer status: You're only worth talking to if you know what I know Closed vocabulary / no shared language Don't know my requirements yet -> knowledge is OUTPUT of the development (as well as the input) False / Hidden assumptions

IMPACT

Encourages exploratory development
If requirements / development not shared – get single point of failure
Missing opportunity to share
Only expert can use it
Can only be supported by expert
No documentation
No quality review (ISO9000)
Usability of system
Closed shop, closed system
No community – missing opportunity to share
Availability, performance and supportablity
No audience
No one knows you are an expert, no external recognition
No validation and verification

Possible SOLUTIONS (very brief – running out of time)

System documentation should be created and reviewed
System / code should be documented as it develops
Tools to support documentation
Documentation done by apprentice
Standards
Classify risks
End user code is just a prototype
IT Buddy
Pair Programming
Quality Gates
View IT as an enabler
Encourage participation
Developed code ranges from DIRTY on one extreme to PROFESSIONAL at other end