Role	***Sub-role***	***Technologies***	***comments***
Software Architecting			
		multiple paradigms	
			generalist
			not a single paradigm purveyor
		whiteboards	
		diagrams	
		code snippets	
		pseudo code	
		DSLs	
			shows ability to view problem from many angles
		Drakon	http://drakon-editor.sourceforge.net/
		StateCharts	
		UML	
		indirection	
	UX Architect		
		UX design is a divide-and-conquer activity	
			design a rudimentary piece of the UX
			test it (for UX-ness, not for robustness) before proceeding
			final design will be a composition of the various pieces
			once designed, Engineering makes it practical and robust
		Humane Interface (Jef Raskin)	
Engineering			
	Realization Engineering		
			define first-cut of realized architecture
			iterate design with Architect until realizable and all I’s dotted and T’s crossed
	Correctness Engineering		

		proofs, etc.	
	UX Engineer		
			define parameters & timing for responsiveness
			usability testing
			feedback to UX architect
	Error Handler Engineer		analyze testability of product (& suggest changes)
			create procedures / scripts for Q/A
		Throw / catch	
		Signals	
		Events	
		A.O.C. (Aspect Oriented Programming)	
	Maintenance Engineer		
		Refactoring	
		D.R.Y.	
	Optimization Engineer		
		Profiling	
			remove Architectural indirection if appropriate
	Security Engineering		
	Test Engineer		
		Incoming Test	
		Black Box Testing	
		White Box Testing	
		Q/A	
		Scripting	
		Back-to-back testing	
		Sikuli	
	Release Engineer		
		CD	
		Dashboards	
		CI	
Implementation			
	Q/A		

	Maintenance Testing	
		Hardware production test used HP Trace Analyzers that would generate a GUID for every test (including sequencing over time) for a “golden unit” (known to be good), when GUID didn’t match in production unit, then further testing was used to determine where the fault was (kind of a Canary CI, replacing Unit test with faster/cheaper tests which signalled Go/no-go only)
Teaching Software to Children		
	different set of concerns than providing tools to Professionals	
		Rhetorical Question: would you drive across a bridge designed by a gifted child?
Software for Business		
		Rhetorical Question: would you drive across a bridge designed by a Professional who isn’t an Engineer? E.g. a Dentist?
	Word Excel Visio Scapple Scrivener	
Software for Domain Experts (not Programmers)		

	HyperCard		
	VisiCalc		
	people with expertise who see a need and want to learn “just enough” programming to fill that need	e.g. accounting software	
		absolute addressing	
		grid layout (VisiCalc)	
		fixed layout (HyperCard)	
		few options	
		“obvious”	
Software Designs Based on Existing Paradigms			
	transitional (only)		
		will be supplanted by designs based on computing-driven Paradigms	
	desktop		
	filing cabinet		
	typewriter		
	TV schedule		New paradigm: Netflix
	magazine articles		New paradigm: blogs
	typewriter keyboard		New paradigm: tablet, phone
	house phone		New paradigm: iPhone
	retail		New paradigm: Amazon
	libraries		New paradigm: internet
			New paradigm: what is the new O/S? Do we need an O/S?
	expensive all-in-1 computers		New paradigm: IoT
	desk calculator		
	piano		loops
	recording soundboard (e.g. mimiced by GarageBand, ProTools)		

audio	new paradigm: video+audio, YouTube
whiteboard	New paradigm: ?
office	New paradigm: WFH
house	New paradigm: condo
automobile	New paradigm: bicycle, fat bike, ebike, public transit

counter e.g. everything is Haskell – no
c.e.g. everything is an Object – no
c.e.g. everything is <xxx> - no

earliest drafts tested by Architect and Engineers ; later drafts
tested by Customer (Stakeholder)

test suitability of all bought-in technologies (e.g. code from GitHub)

devise ways to break product

large systems can feed inputs to same kinds of systems