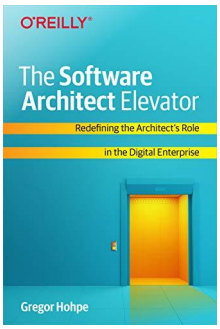

YOUR KINDLE NOTES FOR:

The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise

by Gregor Hohpe

Free Kindle instant preview: https://a.co/6fUlijK

123 Highlights

Highlight (Yellow) | Location 51

> It's about designing software and solving problems within a specific organizational context, and being aware of what's happening around you, so that you can successfully navigate and influence that context where necessary. It's crucial, therefore, that architects realize they need to communicate and influence at different levels, with different audiences, both inside and outside of their immediate team environment.

Highlight (Yellow) | Location 74

> being comfortable with ambiguity is one of the most important attributes of a good architect — asking awkward questions, as my team member illustrated, being another!

Highlight (Yellow) | Location 77

> connecting, explaining, questioning,

Highlight (Yellow) | Location 80

> I believe that architects make two things that are of vital importance and in short supply: they make sense and they make decisions.

Highlight (Yellow) | Location 94

> The job of architects now is to create the conditions for speed and dynamism within their organizations: to satisfy the design goals of change velocity and service quality at the same time (and to help people understand that these goals are not in conflict; see Chapter 40).

Highlight (Yellow) | Location 141

> reason about large-scale architectures; how to ensure your architecture benefits the business strategy; how to leverage vendors' expertise; and how to communicate critical decisions to upper management.

Highlight (Yellow) | Location 166

> To transform an organization, you don’t need to solve mathematical equations. You need to move people, and that’s why you need to be able to tell a good story and paint a compelling vision.

Highlight (Yellow) | Page 3

> architects deal with nonrequirements. This term doesn’t indicate things that aren’t required; rather, it refers to requirements that aren’t stated anywhere. This includes context, tacit assumptions, hidden dependencies, and other things that were never spelled out.

Highlight (Yellow) | Page 3

> if an IT system can still absorb high rates of change after many years, the project team probably included a good architect.

Highlight (Yellow) | Page 3

> Architects “connect the dots”

Highlight (Yellow) | Page 3

> Architects see trade-offs

Highlight (Yellow) | Page 4

> Architects look beyond products

Highlight (Yellow) | Page 4

> Architects articulate strategy

Highlight (Yellow) | Page 4

> Architects fight complexity

Highlight (Yellow) | Page 4

> Architects deliver

Highlight (Yellow) | Page 4

> Staying grounded in reality and receiving feedback on decisions from real project implementations is vital for architects. Otherwise, control remains an illusion

---

Highlight (Yellow) | Page 7

> departments speak different languages, have different viewpoints, and drive toward conflicting objectives.

Highlight (Yellow) | Page 8

> The worst-case scenario materializes when people holding relevant information or expertise aren't empowered to make decisions, whereas the decision makers lack relevant information.

Highlight (Yellow) | Page 11

> I was once criticized by the engine room for pushing corporate agenda against the will of the developers while at the same time corporate leadership chastised me for wanting to try new solutions just for fun. Ironically, this likely meant I found a good balance.

Highlight (Yellow) | Page 12

> blowing the whole thing up just leaves you with a pile of rubble, not a lower building.

Highlight (Yellow) | Page 16

> The enterprise architect is sometimes seen as such a person — the all-knowing decision maker.

Highlight (Yellow) | Page 17

> The role of the gardener is to trim and prune what doesn't fit and to establish an overall balance and harmony in the garden, keeping in mind the plants' needs.

Highlight (Yellow) | Page 20

> instead of the superhero we need "super glue" architects — the guys who hold architecture, technical details, business needs, and people together across a large organization or complex projects.

Highlight (Yellow) | Page 21

> In a Constantly Moving World, Your Current Position Isn't Very Meaningful

---

Highlight (Yellow) | Page 22

> the only system that wouldn't benefit from architecture is one that doesn't change at all.

Highlight (Yellow) | Page 22

> the rate of change is a major driver of architecture's value and architectural decisions.

Highlight (Yellow) | Page 25

> Systems with fewer interdependencies — for example, because they are modular and cleanly separate responsibilities — localize changes and can therefore generally absorb a higher rate of change.

Highlight (Yellow) | Page 26

> If fear slows you down, confidence should speed you up.

Highlight (Yellow) | Page 27

> Digital companies only know one speed: fast.

Highlight (Yellow) | Page 28

> architects should be part of a trusted but diverse network of experts, which can provide unbiased information.

Highlight (Yellow) | Page 30

> To receive a warmer welcome from IT staff, one should therefore be careful with the label enterprise architect.

Highlight (Yellow) | Page 31

> Business architecture translates the structured, architectural way of thinking (Chapter 8) that’s guided by a formalized view of components and interrelationships into the business domain.

Highlight (Yellow) | Page 32

> Enterprise architecture is the glue between business and IT architecture.

Highlight (Yellow) | Page 32

> Alas,

Highlight (Yellow) | Page 32

> in the old days, when everything was running on paper instead of computers, companies also didn’t have a separate “paper” department and a CPO — the chief paper officer.

Highlight (Yellow) | Page 33

> In the digital age, IT is a competitive differentiator and opportunity driver, not a commodity like electricity.

Highlight (Yellow) | Page 38

> When asked to characterize the seniority of an architect, I apply a simple framework: a successful architect must stand on three “legs”: Skill The foundation for practicing architects. It requires knowledge and the ability to apply it to solve real problems. Impact The measure of how well an architect applies his or her skill to benefit the company. Leadership Determines whether an architect advances the state of the practice.

Highlight (Yellow) | Page 38

> Skill is the ability to apply relevant knowledge which can relate to specific technologies (such as Docker) or architectures (such as microservices architectures).

Highlight (Yellow) | Page 39

> Impact is measured by the benefit achieved for the business, usually in form of additional revenue or reduced cost. Faster times to market or the ability to incorporate unforeseen requirements late in the product cycle also positively affect revenue and therefore count as impact. Focusing on impact is a good exercise for architects to not drift off into PowerPoint-land.

Highlight (Yellow) | Page 39

> rational and disciplined decision making (Chapter 6) as a key factor in translating skill into impact.

Highlight (Yellow) | Page 39

> The leadership leg acknowledges that experienced architects do more than make architecture. Mentoring junior architects can save a new generation of architects many years of learning by doing.

Highlight (Yellow) | Page 39

> Skill without impact is where new architects start out as students or apprentices.

Highlight (Yellow) | Page 39

> Impact without leadership is a typical place for architects who are deeply ingrained in projects but “don’t get out much.” Such architects will plateau at an intermediate level, which is bad for them and their employer. The architects will likely hit a glass ceiling in their career because they won’t be able to see beyond their current environment.

Highlight (Yellow) | Page 40

> Many companies are penny-wise and pound-foolish by not placing sufficient emphasis on nurturing their architects. They fear that any distraction from daily project work will be unproductive. However, they miss out on growing world-class architects.

Highlight (Yellow) | Page 40

> IBM distinguished engineers and fellows are expected to demonstrate giving back to the community both internally (e.g., via mentoring) and externally (e.g., via conference presentations or publications).

Highlight (Yellow) | Page 40

> leadership without (prior) impact lacks foundation and might be a warning signal that an architect has become an ivory tower resident with a weak link to reality.

Highlight (Yellow) | Page 41

> As architects, we know that scaling vertically (getting smarter) works only up to a certain level and can lead to a single point of failure (you!). Therefore, you need to scale horizontally (Chapter 30) by deploying your knowledge to multiple architects. The scarcity of good architects makes this step more important than ever.

Highlight (Yellow) | Page 42

> Lastly, sharing openly and demonstrating thought leadership offers another huge benefit: it can give you access to a powerful community of other thought leaders, which in turn makes you a better architect.

Highlight (Yellow) | Page 45

> can’t judge a decision by the outcome alone, simply because you didn’t know the outcome when you made the decision.

Highlight (Yellow) | Page 46

> Humans are terrible decision makers, especially when small probabilities and grave outcomes like death are involved.

Highlight (Yellow) | Page 46

> “The law of small numbers”; people tend to jump to conclusions based on sample sizes that are way too small to be significant.

Highlight (Yellow) | Page 47

> confirmation bias

Highlight (Yellow) | Page 48

> prospect theory: when faced with an opportunity, people tend to favor a smaller but guaranteed gain over the uncertain chance for a larger one:

Highlight (Yellow) | Page 48

> loss aversion.

Highlight (Yellow) | Page 48

> priming, can influence decisions based on recent data we received.

Highlight (Yellow) | Page 49

> Decision analysis helps us think rationally

Highlight (Yellow) | Page 49

> A one-in-one-million chance of dying is called one micromort.

Highlight (Yellow) | Page 49

> The amount you are willing to pay to avoid this risk is called your micromort value.

Highlight (Yellow) | Page 50

> The best overview of models and their application I have come across is Scott Page's Coursera course on Model Thinking

---

Highlight (Yellow) | Page 52

> Many IT decisions — especially those related to cybersecurity risks or system outages — share similar characteristics of small probability but severe downsides.

Highlight (Yellow) | Page 52

> separating likelihood from impact and baselining probabilities can help remove emotion, resulting in more rational decisions.

Highlight (Yellow) | Page 52

> what's the best decision? It's the one that you don't need to take!

Highlight (Yellow) | Page 52

> eliminate irreversibility in software designs."

Highlight (Yellow) | Page 52

> In a well-designed software system, decisions aren’t as final as when taking deadly pills from a jar.

Highlight (Yellow) | Page 53

> I often introduce myself as a person who knows the right questions to ask.

Highlight (Yellow) | Page 54

> It’s also helpful to remind your counterparts that you are not challenging their work or competence, but that your job requires you to understand systems and problems in detail so that you can spot potential gaps or misalignments.

Highlight (Yellow) | Page 54

> An architecture review is not only looking to validate the results but also the thinking and decisions behind it all.

Highlight (Yellow) | Page 55

> Unstated assumptions can be the root of much evil if the environment has changed since the assumptions were made.

Highlight (Yellow) | Page 55

> “You didn’t come here to make the choice, you’ve already made it. You’re here to try to understand why you made it”

Highlight (Yellow) | Page 55

> Asking questions in traditional organizations might not get you insights but defensiveness to cover up the lack of decision discipline.

Highlight (Yellow) | Page 56

> she likes to have the architecture team involved because, “we have nothing to sell, no one can fool us, and we take the time to explain things well.”

Highlight (Yellow) | Page 60

> It’s simply a matter of whether you consciously choose your architecture or whether you let it happen to you. History has shown that the latter approach invariably leads to the infamous Big Ball of Mud1 architecture, also referred to as shantytown.

Highlight (Yellow) | Page 61

> Generally, good architecture buys you flexibility. In a rapidly changing world, this seems like a smart investment.

Highlight (Yellow) | Page 61

> Architecture is a matter of trade-offs: there rarely is one single “best” architecture.

Highlight (Yellow) | Page 61

> Architects should also strive for conceptual integrity, that is, uniformity across system designs.

Highlight (Yellow) | Page 62

> The real world must deal with many of the same issues faced by large enterprises: lack of central governance, difficult to reverse decisions, enormous complexity, constant evolution, slow feedback cycles.

Highlight (Yellow) | Page 64

> One of my favorite tests for architecture documentation is whether it contains any nontrivial decisions and the rationale behind them.

Highlight (Yellow) | Page 64

> The structure of the components of a system, their interrelationships, and principles and guidelines governing their design and evolution over time.

Highlight (Yellow) | Page 64

> The set of design decisions about any system that keeps its implementors and maintainers from exercising needless creativity.

Highlight (Yellow) | Page 67

> Significant architectural decisions may look obvious in hindsight, but that doesn’t diminish their value.

Highlight (Yellow) | Page 67

> Assessing

Highlight (Yellow) | Page 68

> when discussing architectures, let’s talk about what isn’t obvious.

Highlight (Yellow) | Page 68

> A good test is whether the chosen option also has downsides — decisions without downsides are unlikely to be meaningful.

Highlight (Yellow) | Page 70

> “one of an architect’s most important tasks is to eliminate irreversibility in software designs.”

Highlight (Yellow) | Page 71

> being able to defer a decision while fixing the parameters has value.

Highlight (Yellow) | Page 71

> An option is defined as “the right, but not the obligation, to execute a financial transaction at fixed parameters in the future.”

Highlight (Yellow) | Page 72

> An option allows you to defer a decision: instead of deciding to buy or sell a stock today, you can buy the option today and thus acquire the right to make that decision in the future.

Highlight (Yellow) | Page 72

> The financial industry knows quite well that deferring decisions has value, and therefore an option has a price,

---

Highlight (Yellow) | Page 73

> Essentially, you’re paying for the option with increased complexity.

Highlight (Yellow) | Page 75

> The more uncertain about the future I am, the more value I derive from deferring a decision.

Highlight (Yellow) | Page 75

> The business not wanting to be involved in technical decisions leads to suboptimal decision making because IT alone can’t judge the value of an option. Instead, it’s the architect’s job to translate technical options into meaningful choices for the business.

Highlight (Yellow) | Page 75

> Architects and project managers typically work under different time horizons and thus value the same option differently.

Highlight (Yellow) | Page 77

> Both Agile methods and architecture are ways to deal with uncertainty, meaning that working in an Agile fashion allows you to benefit more from architecture.

Highlight (Yellow) | Page 77

> in times of high uncertainty, as we are facing them today both in business and technology, the value of architecture options also increases. Businesses should therefore invest more into architecture.

Highlight (Yellow) | Page 78

> systems thinking emphasizes behavior

Highlight (Yellow) | Page 81

> Gerald Weinberg2 highlighted the importance of thinking in systems by dividing the world into three areas: organized simplicity is the realm of well-understood mechanics, such as levers or electrical systems consisting of discrete resistors and capacitors. You can calculate exactly how these systems behave. On the other end of the spectrum, unorganized complexity doesn't allow us to understand exactly what's going on, but we can model the system as a whole statistically because the behavior is unorganized, meaning the parts don't interrelate much. Modeling the spread of a virus falls into this category. The tricky domain is the one of organized complexity, where structure and interaction between components matter, but the system is too complex to solve it by using a formula. This is the area of systems. And the area of systems architecture.

Highlight (Yellow) | Page 82

> Bounded rationality,

Highlight (Yellow) | Page 82

> the tragedy of the commons

Highlight (Yellow) | Page 83

> The

Highlight (Yellow) | Page 84

> Most of what users see from a system are events: things happening as a result of the system behavior, which in turn is determined by the system structure, that is often invisible.

Highlight (Yellow) | Page 84

> humans are particularly bad at steering systems that have slow feedback loops,

---

Highlight (Yellow) | Page 87

> most complex configuration really is just programming, albeit in a poorly designed, rather constrained language without decent tooling or useful documentation.

Highlight (Yellow) | Page 89

> The best abstractions are therefore those that solve and encapsulate the difficult part of the problem while leaving the user with sufficient flexibility.

Highlight (Yellow) | Page 89

> If an abstraction takes away too many or the wrong things, it becomes overly restrictive and no longer applicable. If it takes away too few things, it didn’t accomplish much in terms of simplification and hence isn’t very valuable.

Highlight (Yellow) | Page 91

> If the algorithm is predefined and you supply only a few key values, it may be fair to call this configuration.

Highlight (Yellow) | Page 92

> Rather than trying to anticipate changes for configuration, you may want to invest in your tool chain to allow incremental, rapid deployment.

Highlight (Yellow) | Page 94

> Corporate IT lives among zombies: old systems that are half alive and have everyone in fear of going anywhere near them.

Highlight (Yellow) | Page 96

> Systems fall into the state of legacy because technology moves faster than the business:

Highlight (Yellow) | Page 97

> The key driver for managed evolution is to maintain agility in a system.

Highlight (Yellow) | Page 99

> many IT departments spend more than half of their IT budget on “run” and “maintenance,” leaving only a fraction of the budget for “change” that can support the evolving demands of the business.

Highlight (Yellow) | Page 100

> One item routinely missing from such “features” lists is planned obsolescence: how easy is it to replace the system? Can the data be exported in a well-defined format? Can business logic be extracted and reused in a replacement system to avoid vendor lock-in?

Highlight (Yellow) | Page 104

> “automate everything and make those parts that can’t be automated a self-service.”

Highlight (Yellow) | Page 104

> Just like test-driven development is not a testing technique (it’s primarily a design technique), automation is not just about efficiency but primarily about repeatability and resilience.

Highlight (Yellow) | Page 104

> repeatability and traceability: wherever humans are involved, mistakes are bound to happen, and work will be performed ad hoc without proper documentation.

Highlight (Yellow) | Page 105

> Automation is hugely liberating and hence speeds up work significantly.

Highlight (Yellow) | Page 106

> once set a rule that no infrastructure changes could be made from a user interface but had to be done through version-controlled automation. This put a monkey wrench into many vendor demos.

Highlight (Yellow) | Page 111

> economic benefits of a software-defined infrastructure are too strong for anyone to put a stop to it.

Highlight (Yellow) | Page 113

> Software-defined infrastructure therefore isn’t just about replacing hardware configuration with software, but primarily about adopting a rigorous development life cycle based on disciplined development, automated testing, and CI.

Highlight (Yellow) | Page 113

> Over the past decades, software teams have learned how to move quickly while maintaining quality. Turning hardware problems into software problems allows you to take advantage of this body of knowledge.

Highlight (Yellow) | Page 120

> Platform standards essentially split the IT into two parts: a lower layer that standardizes those elements that are unlikely to form a competitive differentiator and an upper layer of in-house-developed software that provides direct business value and competitive differentiation.

Highlight (Yellow) | Page 121

> my favorite definition of software architecture: design decisions that keep implementors from exercising needless creativity

---

Highlight (Yellow) | Page 126

> Creating an accounting system yourself is in most cases as valuable as creating your own electricity. It’s important to have such things, but they won’t give you any competitive advantage.

Highlight (Yellow) | Page 127

> IT architects in large enterprises must therefore develop their own, balanced worldview so that they can safely navigate the treacherous waters of enterprise architecture and IT transformation.
