case studies from practice

34508 characters in 4844 words on 917 lines

Florian Moser

June 5, 2018

1 company tools

1.1 challenges in companies

decide without complete facts
meet impossible deadlines
finish a half-our argument in one minute
archive something with a team of disagreeing individuals

development
ensure compatibility with other products
maintain and extend a large code base
deal with ambiguous/conflicting requirements
question solutions which are "quick and easy"

strategy
difficult to establish in large companies
invest time to ensure all branches follow same strategy
establish consensus about objectives & approach

legacy
superseded SW/HW which is difficult to replace due to wide use
old systems may be badly written, but not necessarily
old systems battle-proven, therefore relatively stable

assess reliability of statements
hearsay, alternative facts
unjustified belief, opinion
factually, statistically, justified opinion
definition, proven theorem

technical problem motivation
build a tree, from left to right
start with technical problem
describe business problem(s) it leads to
describe possible solution(s) for each business problem

1.2 information formats

in general
present only short list of options, only structured data
don't describe analysis approach / previous work
answer "so what" / motivation as early as possible

decision based presentation
provide base to make a decision
present short list of options
describe implications of these options
provide single recommendation & its rationale

take-away based presentation
max 3 key take aways

1.3 prepare presentation

find out about audience
talk to stakeholders in advance (find out view points, expectations)
think about already brought up questions / issues
consider conflicting interests from different stakeholders

define objectives
work out what is expected to result from meeting
define decisions to be taken, take aways to be remembered
place all further material, math in appendix

develop structure
possibly present initial situation
top down (start with decision, then arguments, reasoning)
or tell a story & align with examples

create presentation
develop presentation based on storyline
each slide one core statement with 3 min presenting time
fast sketches by hand before spending time on perfecting them
not necessarily self-contained, focus on content, not design

finalize
proofread on printouts, verify objectives are reached

1.4 decompose complex questions

issue tree (disaggregation)
if given a problem ("why budget overrun")
tree from problem statement to issues, sub-issues
mutually exclusive, collectively exhaustive, relevant (MECERE)
slow but steady approach, clearly see most important issues
"budget overrun" $\rightarrow$ unit cost too high; scope too big
"any decision" $\rightarrow$ benefits; cost; risk; strategic flexibility
"MVP" $\rightarrow$ full design/implementation $\rightarrow$ functional, reliable, usable, UX

option tree
if given a desired outcome ("deal with bad PM")
adhere to CERE, ME is nice to have
slash non-feasible, build business case for others
easily compare available solutions
"cost too high" $\rightarrow$ "let people go", "stop business"

hypothesis tree (hypothesis driven)
if given an opinion ("budget overrun because bad PM")
identify & collect arguments needed to prove hypothesis
test hypothesis fast, for complex projects
fast to do, but only if hypothesis is broadly correct
"more budget bad" $\rightarrow$ costs unclear; requirements impossible
"reduce scope good" $\rightarrow$ delivery time; cost; risk; easier to change

1.5 big projects

problems
incomplete, changing requirements
insufficient user involvement with unrealistic expectations
insufficient planning, resources & management support

evaluate bad performing projects
base decision on how much more time / budget needed
forget about sunk cost

cost
solution deliver cost (design, realization, testing)
ongoing activities (operation costs, maintenance)

improve profitability
reduce cost (per unit / resource, of employees, of operations)
increase revenue (more customers, sell more units, increase price)

any (IT) decision
benefits (revenue increase, cost & risk reduction)
cost (development, maintenance, operations)
risk (technical, business)
strategic flexibility (decision, production)
implicit feasibility (higher cost & risk)

2 business tools

2.1 analyze company / products

SWOT
strengths (internal, helpful)
weaknesses (internal, harmful)
opportunities (external, helpful)
threats (external, harmful)

business model canvas results
key partners (interdependencies)
key activities (focus of business)
value propositions (customer value)
customer relationships (communication, trust)

channels (sell, communicate)
customer segments (target group)
revenue streams (billable services)
cost structure (expenses)

2.2 create new product

startup vs internal development
startup can scale up/down fast with entirely new mindset
is free from legacy business, more pressure for fast MVP
but higher cost, knowhow transfer difficult, ownership of staff unclear
internal can provide fully staffed, interdisciplinary teams
has connection to customers for early feedback, field tests

establish vision
develop vision of what to build before starting other things
may externalize to startup to enable new thinking models

interview customers
not asking for needs (only improved versions of products are recommended)
asking for pains to discover real problems, focus product on that

define target customers
large customers for large cash flow from only a few customers
but have high negotiation power, long requirements list
small customers easier to get to use MVP without reputation
are easier understood, accept outsourced, clouded application

calculate price of product
higher than production cost
lower than customer willingness to pay (perceived benefit, alternatives)

allocate resources
there can also be too much, scaling up / scaling down needs time
at first start with a few people
develop as a big team, then scale down again

build MVP
functional, reliable, usable, emotional design
build good first impression

3 problem solving tools

iterative approach (redefine problem after analysis if needed)
be critical to proposed problem statements

problem statement
agree on question (establish shared focus, assure actionable results)
define underlying problem (describe as-is, find drivers, causes & effects)
find decision makers (key stakeholders & the ones who decide in the end)
establish decision criteria (provide foundation to make decision)
set solution constrains (ethical, moral, strategic limits to solutions)
set solution scope (define out of scope items, like international units)

investigate
potentially build hypothesis tree to correct arguments, focus investigations
analyze data, collect experience, interview stakeholders, read reports
be respectful (mistakes happen, don't judge by degree/background)

analyze & structure problem
build issue tree to understand components of problem (what, how)
think from business perspective (cost, risk, revenue, flexibility)
make each problem component intellectually manageable / solvable
build a common understanding, clarify priorities / responsibilities
summarize key learnings, start over if problem definition needs review

synthesize results
define evaluation/success criteria (feasibility, costs, risks, payoff)
ask "so what" / "what must be done", assume company perspective
build MECERE option tree (always include do nothing)
analyze options using facts (stating any additional assumptions taken)
check implementation feasibility & risk, slash options not worth pursuing
analyze viable options detailed (using success criteria, cost analysis)
do technical evaluation (compatibility, availability)
do functional evaluation (functionality offered)
do integration evaluation (integrations needed)

formulate recommendation
propose best option, may goes beyond problem statement
propose next steps to be undertaken

4 dormakabra

sells locks

4.1 SWOT results

strengths (internal, helpful)
design portfolio, identity, long-lasting customers
experience, use of market potential, premium technology
complete business offering (technology, firmware, installation)

weaknesses (internal, harmful)
development costly & customer driven, inefficient software platform
local focus, no global price, no multiplication of output

opportunities (external, helpful)
system integration, parter distribution, home automation

threats (external, harmful)
rising complexity, partners collaborate with competition
market saturation, growing competition

results
usecase specific UI, package based offering, health check (opportunities)
solutions instead of products, consulting, design, UX (against threats)

4.2 business model canvas results

key partners
home automation
data center, development partners

key activities
sales, ordering, shipping, repair
support, communication, training
operations, delivery, production

value propositions
planing, easy install, integrations
complete system status, access control
continuous product improvement

customer relationships
business infrastructure
trust in brand and solution

channels
local partners, installers

customer segments
residential, small/medium, partners

revenue streams
planning, hardware sales, installation, configuration, support
cloud usage, maintenance, credential management, special solutions

cost structure
sales, support, marketing, cloud operations
process adaptation, new products, material

4.3 analysis results

established vision
provide parters & customers with access control management
integrate planning, sales, ordering, CRM & support

customer pains
needs to be AC expert, but high complexity & flawed technical support
fears proprietary solutions
hates slow delivery time, low delivery quality
fears breakdown of security

4.4 kaba exivo

shared vision to solve pains rather than detailed requirements

develop new market
definition & setup of new market as difficult as development
subscription plan for recurring revenue & high customer binding
technology enabled new market, but internal restructuring needed

develop new platform
access control as a service
for small practices (lawyers, doctors; need to comply with law)
early & often usability tests with end users
focus on business value rather than fancy technology
develop internally instead in startup to avoid reintegration issues

risks
high investment to until customer value generated
balance brand value & early market release
established business processes difficult to adapt

timeline
envisioning (create core-team, establish vision, scope, requirements)
start (scale up for MVP, agree on architecture, technical concepts)
stabalize (scale down team, finish the project, freeze features / scope)
operations (start production / maintenance / support)

impact on stakeholders
employees need new skills (release cycles, security, user-centric design)
partners need to adapt workflow to new platform
customers need to be persuaded of new pricing, are more dependants
managers must take new end-to-end responsibility, new procedures
managers must manage touch points of new system with old business
managers must support development approaches (fail early, new tools, reporting)
hr/recruiting must attract new employees from different industry
sales needs to rethink measures of success

5 teralytics

big data firm, collects usage of phone data

5.1 architecture

aggregate data
anonymize data (still in RAM)
clean up noise
estimate location based on events
interpolate traces
aggregate & extrapolate
prepare results for dashboard

dashboard
visualizes data
records any query
stop execution if too much info revealed

VPN
to teralytics office (ISO certification)

5.2 provide data

aggregated by zip code, by weekday, by month, by day

legal issues
set minimal bound according to regulations
respect general data protection directive GDPR, contry specific
allow opt-out (usa with webpage, ignore foreigners)

hosting issues
AWS (very expensive due to constant, predictable load)
therefore do hosting by yourself even if non-core business
cluster at teralytics (easy physical access, faster connection)
but legal issues EU vs CH, who pays cluster
cluster at peppermobile (responsible for security, maintenance)
but slow processes, can't extend old machines

pricing
calculate delivery of service costs (lower bound)
calculate customer benefits, costs (higher bound)
compare with other products

trial concept
big enough to show benefit, but not to reveal everything
contracts to make usage illegal outside trial, be prepared to sue

contract versions
fremium (more functions, services if customer pays)
license based contract (pay per view)

6 hotelcard

hotel-halbtax
quite successful, offerings of all variety
customer buys member fee, hotels pay no commission

6.1 win-win

hotel
higher occupancy rate
no booking portal commission
more revenue from secondary services
free publicity
self-controllable availability

guest
overnights at half price
large offering, high availability
unlimited usage

6.2 market platform

supplier with overcapacity, clients with frequent usage
sustainable high profit margin

network effect
value for each participant increases
competitors have hard start

introduction strategies
letter of intent (register if 150 parties agreed)
partnerships with TCS, SBB, tamedia
exclusivity (regional, limited in time)

6.3 find first 150 hotels

cover all regions, types of hotel

letter of intent with exclusivity
first signing right for 14 days
get exclusive regional access for limited time
hotels promise to create profile if 150 hotels found

6.4 sales strategy

sales partner like SBB, TCS

offer
via company newsletter/magazine
to clients, members, employees
discount at first year, automatic renewal

digital marketing
emails to clients/potential clients
sales portals like DeinDeal
SEO, SEA (+advertising)

media 4 equity (M4E)
tamedia takes 20%, pays off with spare advertising space
various print & digital media

product variants
personal hotelcard (double room booking)
company hotelcard (transferable)
vouchers (as a gift purchasable)

estimated market saturation
1 mio (+1 mio with surrounding countries)
maximize renewal rate with newsletters, inspiration
aquire new customers for less than 95 CHF

7 startups from ETH

7.1 focus types

technology (AdNovum)
programming languages, compilers
software engineering, data modelling
IT security
information retrieval

function (CodeCheck)
marketing (big data, CRM)
human resources (eLearning)
finance (payments)

industry (Avaloq)
aviation, banking, clothing
energy, health, construction

platform (Doodle)
B2B, B2C

7.2 startup summary

leader
software guy, passion, stamina, fun
is a role model for employees
team builder & capable of learning
focused and competencies in various areas

team
search for complementary skills & passion (even lawyers)
discuss ideas openly with others
define clear responsibilities
team up engineering with sales & UX
don't delegate recruiting to HR or keep under performing employees

financial
we build it, they finance it
we own it, they get the right to buy it
preserve reputation of customer
don't develop product without market

process
have a stable vision but flexible strategy
be patient, build your brand
dare to make decisions
pay attention to timing
don't be shy or stubborn

product
build high end software, platform, eShop, SAAS
focus on specific product for specific, big market
focus on urgently needed pain killer with highest value add
find customers & smart plan for initial population (avoid investors)
make it alternative-less, multipliable, high margin, scalable
look for n:m market (even better if n:n,)
multiply business model at other branches
try to stay at service level
don't be replaceable, too local, nice-to-have

sell
B2B easier than B2C, to business easier than to engineering
peer2peer advertising
don't spend unwisely on advertising

8 cost effectiveness and software metrics

8.1 productivity of software development

function points / person months inversed exponentials
10 points / months at 100 points
4 points / month at 2000 points
2 points / month at 8000 points
1 point / month at 16000 points, and decreasing further
comparison applies within same company or controlled environment

8.2 function point (FP)

metric to asses quality and productivity of software
convert requirements to FP, then estimate cost with past metrics
works well for comparable projects (same company, technology)

usage
time (person months needed for large project)
cost (total cost of project)
schedule (time of completion)
deliveries (intermediate releases)

benefits
technology independence (high vs low power language accounted for)
variability (to any kind of software)
completeness (all activities measured)
analysis (over multiple projects, technologies)
estimate at start of project (unlike LOC)

functional classes
external inputs (values in input mask)
external outputs (report)
internal logic files (states)
inquiries (query to db)
external interface file (like database)

example external input
data element type (how many inputs)
file type reference (how many groups)
then use table to estimate FP

8.3 FP results

relations
1P is small code change, 100 LOC
10P is small application, 1k LOC, 5K
100P is application, 12k LOC, 100K, 9months
1k P is commercial, 100k LOC, 1m, >1y
10k P for system, 1m LOC, 10m, 5y, 100people
100k P for OS, 12m LOC, 1bio, 8y

observations
maintenance fixed at 10%-15% of project development effort

low language levels
more logic mistakes per statement
more LOC per function point
more defects per function point

changes with FP (1 $\rightarrow$ 10000)
management effort (10 $\rightarrow$ 16)
error correction (15 $\rightarrow$ 35)
paper work (5 $\rightarrow$ 31)
coding (70 $\rightarrow$ 18)
changing requirements (0 $\rightarrow$ 45)
poor project quality (0 $\rightarrow$ 90)
observed project delay (0 $\rightarrow$ 80)
project cancellation (0 $\rightarrow$ 40)

explanations why big is bad
implementation of new functionality more difficult
large retesting efforts after each new change
maintenance effort bigger (missing support, libraries stop working, ...)
architecture degrades (unused code, now obsolete or unfit patterns)
knowledge gets lost (employee leaves, forgets)

conclusions
productivity decreases with size of system
synergies of projects must be high to be integratable in single product
flexibility often bad (FP overhead but hard to predict)
marginal requirements increase cost, risk exponentially

8.4 estimate new project

find efficiency at current FP level
calculate person months
take square roots for team size / duration
read out LOC / FP for the specific language

argue
assume 30% for coding & 30% error correct
if another language, compare LOC/FP and change PM accordingly
if new language to learn, include performance reduction 10%

9 AIP case study

want to replace old system, which has proven itself
currently most money spend on maintenance

9.1 AIP II

solves future problems with parametrization
new architecture & new technologies

but unrealistic
API was designed to be flexible as well
huge risk & not doable in this time
maintenance increases substantially

9.2 AIP II shared

collaboration with another company
productivity decrease through larger project

but collaboration difficult
scope creep (incentive to add more of own functionality)
difference in company culture (decision taking harder)
system integrator in strong position (can play off companies)

9.3 AIP renovated

renovate AIP to remove duplication, architecture issues
remove unnecessary functionality
increase flexibility & parametrization at selected times
include new business functionality

9.4 bad behaviour

from user
marginal additional features cause cost, time overruns
requesting flexible solution instead of fixing requirements
forgetting about increased complexity while trying to use synergies
bias towards description of requirements by other people

from staff
forgetting about increased complexity while trying to use synergies
building flexible solution, but not good in anticipating future requirements
blindly fulfilling all requests (not in company's best interest)
requesting rewrite instead of maintaining properly

9.5 conclusions

as few features as possible
all have to be maintained and increase complexity
synergies with other projects have to be high to be useful
flexible requirement connected to large overhead
marginal requirements increase cost, time overproportionally

communication
failures due to lack of user involvement, business buy-in
plan well, make assumptions explicit, keep up to date

legacy systems
can often still fulfil business needs
should run as short as possible at the same time as replacement
but avoid big bang replacements due to high risk

requirement engineering hard
old staff left the firm
documentation incomplete, misleading, motivation low
source code only real requirement, but hard to analyze

10 digital strategy

10.1 attack points

customer facing to increase revenue
non-customer facing to reduce cost
new products & services (like new possible products)
new business models (like online shop)
new customer needs (like cloud services)

10.2 objectives

value created/delivered with digital channels
key performance indicators KPI to measure success of strategy

10.3 roadmap of initiatives

potential digital initiatives with their respective impact & complexity
conclude coordination needed, prioritization or projects

10.4 organization

governance structure to enable timely delivery

10.5 workflow

10.5.1 framework & inventory

analyze digital portfolio (past, present, future)
gather end-user expectations & pain-points
understand the value chain & customer journeys
do interviews with employees & clients
use as-is assessments, value chains, customer journeys

value chain
how company creates value to its customers
analyses departments such as communications, marketing, product, customer care

detailed persona
personification of a typical customer
socio-demographic information (age, marriage, location, occupation, salary)
personal quote (describing individual situation)
motivations (work, learning, social, shopping, fun)
expectations (list of needs, pains), the core information!
behaviours (usages of devices, apps, websites, spare time)
influences (of other products, services, peoples)

customer journey
analyze customer touchpoints with brand/product before/after purchase using persona
discover (push info, ads / friends), explore (pull info, call / webpage), buy (purchase), engage (maintenance)
collected touchpoints describing actions of user, used channel, emotional response
emotional responses can be frustration, anxiety, satisfaction, happiness

10.5.2 digital benchmark

market analysis & trend forecasting
benchmark current experiences & get ideas from competitors
assess real & perceived limitations
do workshops, project team research
use benchmark templates

benchmark template
describe idea & solution, assess benefits for target group
describe quantified benefits for company (internal) & customers (external)
estimate cost & feasibility
place project in framework (impacted steps for target group)

10.5.3 ideation

organize, participate in ideas generation effort
classify ideas based in defined framework
select ideas based on strategy, feasibility, expected ROI
do workshops, project team research
use priorization quadrants, business cases

solution targets
web, eCommerce, mobile app, sale points, call centers
collect initiatives for all solution targets

priorization quadrants
grid with (technical complexity, customer value) as axis
foundational (low, low) critical for business
quick wins (low, high) for first priorities
optimizations (high, low) for probably not valuable ideas
future end state (high, high) for long-term initiatives

10.5.4 roadmap

assess, organize selected ideas into reasonable, feasible projects
quantify projects
create actionable roadmap
do final workshops, project team analysis
use prioritized portfolio, business case

10.5.4.1 business case

define business problem
about the project & key questions to be resolved
deliverables, objectives, outcomes

identify target customer needs
where does it fit in customer journey
how does customer do it now, how will it change

outline benefits
address issues, ensure alignment to overall strategy
increases in revenues, reductions in cost
customer loyalty & perception changes

estimate cost
total investment (effort + expenses), timeline & different options
assumptions & likelihood of estimates

identify risks
risk when doing projects vs when not
technology risks, technology changes
how to cover if over budget, time

calculate returns
potential return (best/worst estimates)
resources needed to maximize returns

define success metrics
KPI's affected, and their best/worst estimates
metrics to top-line (revenue) & bottom-line (profit) business gains

10.6 breitling

10.6.1 product

price to different socio economic background of clients
product features important
price compromised of value associated & material value

10.6.2 market

informed buying, decision emotional
complex market with different brands, models
customers often online, susceptible for input from environment

10.6.3 non-digital customer journey

discover
advertisement seen (not personalized)

talks to friends (small amount if people reached)
visits shops (only can try out part of collection)

explore
tries on watch (found by chance)
talks to wife (printed, not personalized catalogue)
studies brochure (no filtering)

buy
buy model (purchase rethought if model not available)

engage
maintenance (no news from brand)

10.6.4 digital solutions

10.6.4.1 online catalogue

filtering & recommendation system
responsive design
availability of watches at the retailer stores

benefits
personalized catalogue to customer
insight on customers preferences
better control on authorized retailers

cost drivers
high technical feasibility of catalogue
recommendation system
integration with retail systems

10.6.4.2 online community

sharing of information & opinions platform for customers
co-creation process participation
direct link to customer care for support / news

benefits
informed, structured customers under the influence of the brand
improved customer loyalty
direct link to online community

cost drivers
moderation, customer relationship management
integration with customer care & product development

10.6.4.3 augmented reality

mobile app to try on watches
connected to online catalogue

benefits
try-on of models without having to visit local shop
more visibility of products for customer

cost drivers
new technology (make or buy)
3d high definition models of all watches needed

10.6.5 digital journey

discover
online advertisement (personalized)
forum visit (sees comments, opinions, live chat)
schedules meeting online (guided to breitling, model available)

explore
sales app (clerk shows not available models)
breitling app (oriented to different decision makers)
augmented reality (helps visualize)

buy
online reservation (model ready, end-to-end support)

engage
continuous usage of breitling app (interactions, continuous news)

11 deliotte

11.1 engagement approach

gather data
incident containment and recovery
analysis
prevention & planning
post-incident review/delivery

11.2 incident prepare

incidence will happen, better be prepared for it

escalation criteria
scope (#users, #customers)
criteria (tier 4-1 assets, recoverability, impact)
score / criteria determine step to escalate on

escalation ladder
IT help desk
local/operational response
cyber incident response team
emergency management team
executive committee
board of directors (are informed, but don't act)

cyper security for C level
CxO is held accountable
CEO faces risk of complete business disruption
CFO faces risk of significant losses & high recovery costs
CIO needs to ensure IT runs smoothly
chief strategist may loses strategic plans (aborted acquisitions, ...)
head of marketing has to ensure brand is not abused
general council concerned with lawsuits, IP protection, prosecution

11.3 incident response

gather emergency management team
physical security (law enforcement)
information technology (internal IT)
information security (security specialist)
communications (PR)
legal & compliance (lawer)
chief of staff
human resources

containment/eradication
short term (kill threat asap, preserve evidence, assess/contain impact)
long term (learn from incident, mitigate risk of recurrence)
prepare for breach notification
e.g. disconnect compromised computers, block malicious communication

recovery
get assets operational again, monitor closely
evaluate containment plan, possibly refine
inform internal/external, compile report
coordinate post-breach actions
e.g. verify, update, restore systems, document decisions

post-breach
incident triage compromise 10% of impact
stop attack, communicate, assess impact within days
fix infrastructure, legal issues, manage relationships within a year
repair business processes, invest in defence within years

11.4 incidence good practice

preparation
develop standardized approach to incident response (policies)
ensure consistent results prior, during an incident (playbook)
training, awareness, preemptive / reactive controls (discipline)

detection
identify, validate, report incidents
confirm incident type & initial classification

analysis
determine mechanism, root cause, scope, scale, impact
develop incident remediation plan

containment/eradication
contain, eradicate, recover from incident, notify stakeholders
reduce effects, prevent escalation, ensure business as usual/service level
preserve evidence, document actions & timeline

reporting
produce post incident report
include all activities undertaken, lessons learned

11.5 cyper-attack

visible impacts
deterioration of public relation
costs of technical, legal cleanup
costs caused by disruption of services

post-breach protection/cyber security improvements

invisible impacts
costs of regulatory improvements (GDPR)
loss of IP
loss of potential clients, devaluation of brand
increase of insurance premium
organisational changes when executives change

12 credit suisse

12.1 about

global wealth manager, big investment banking sector
want to be more flexible, global

12.2 history of banking

1960
use of electronic booking machines in branches, ATM's

1970
centralized computing, automation of processes

1980
widespread ATM, new products like LSV

1990
electronic exchanges, internet based products

2000
online banking, 7x24 processing, large # of products

2010
mobile banking, algo trading, blockchain technologies

12.3 history of banking systems

1980
several thousand users sharing platform
single computer handles all transactions
batch processing to distribute payload
highly tuned applications enable high transaction volume

evolution
no more staff for manual transactions
more products, automation, integration, requirements
explosive growth in IT staff, unknown with whole architecture
continuous tweaks of architecture, experienced staff leaves
first attempts to replace systems fail
new complex interfaces, must features, failed migrations
can't find new staff

current status
40 year old applications till running on newest mainframe technologies
fit for purpose, highly scalable, rich applications
but difficult to manage (high complexity, lack of qualified staff)
lot of lost knowledge (business processes, requirements, code)
not fit for real-time processing, micro service

bottom line
legacy application huge asset & burden

12.4 credit suisse setup

swiss banking IT platform (SBIP)
shared with other divisions of credit suisse group
large set of functionalities, old

history
2003 SOA architecture
2006 One Bank (consolidation in SBIP)
2015 decoupling of security processing from SBIP
since 2016 redesign

problems with migration
analysis took to long (results outdates)
limited people with strong business / IT skills to architect complex systems
reverse engineering impossible through continuous changes
design deadlock because of too many dependencies
persistence on detailed upfront analysis
too high risk of implementation, migration
focus lost after restructuring of organisation
unable to obtain budget, too low priority
market events driven cost/investment rationalization

optimized mainframe
rich business functionality
high scalability (optimized code, timely distributed load)
highly integrated, single instance system
but decreasing supply of engineers, high platform cost

successor of mainframe
must have same rich business functionality
but be better platform

parallel distributed system approach
high scalability (distribution, parallelization)
business agility through segregated micro services
adaopt new technologies gradually
low platform cost

12.5 security settlement engine (SE)

manages delivery, receipt of securities (dematerialized & physical)
35 years old, some automation, very efficient
mission critical application, handling large volume of trades & events

problems
multifunctional components with minimal logging (design)
dead code, monolithic synchronous batch processes (technical)
high time to market, it just works, high post-incidence efforts (business)

future DLT platform proposal
replace central security depository CSD with DLT
CORDA as shared infrastructure with banks, CSD, other users
need consensus on ownership, governance, standards, investments, timeline
need business case with benefits, cost, transition risk

general approach
can't replace big bang, too risky
therefore introduce interface for new SE into other applications & old SE

approach
consolidate, standardize existing interfaces, solve bottlenecks
migrate database from IMS (hierarchical) to DB2 (relational)
decommission end-of-day processing, old business processes
start real-time processing

vision
settlement as a service, easy integration of new clients

next steps
micro-service architecture to replace mainframe step by step
but needs clearer interfaces, capability to onboard new regions / users
utilities for customization, integration into existing systems
but needs fast, cheap way to build integration layer with current system
DLT to replace middle man, messaging
but needs sync between DLT & existing database

12.6 migration

to stay relevant & competitive in fintech
rewrite (for new language, platform, apis; but not future ready)
replace (off-the-shelve with customizations, redesign business)
transform (green-field, reinvent business)

drivers
expensive hardware, software licenses
not aligned to IT or business perspective anymore
high operational risk, production stability, change efforts
end of life of architecture, technology
skills & people risks
new approaches (DLT, ML, micro services)

challenges
people (psychological bias against change, office push/pulls)
complexity (high volume, interface/integration/technical/maintenance sophistication)
financial (significant cost increase, multi-year commitments)
legacy knowledge (lost & forgotten, old philosophy & technology)
project management (different stakeholders, moving target, large project)

new technical approaches
utilities (existing standard products)
distributed ledger technology (DLT) to ease communication
machine learning, artificial intelligence for big data problems
micro service architectures to replace monoliths
agile development for faster execution, validation of requirements

12.7 lessons learned

mainframes

40yrs old systems still considered ”fit for purpose”
efficient code to operate on 40yrs old hardware with large business volume

migrations
can transformed, replaced, or rewrite
some attempts to collect requirements failed
need for coexisting platforms due to step-by-step migrations

stategies
micro services for flexibility, but difficult to decompose systems
DLT, but bottlenecks to work at scale of bank (solutions exist)
utility approach, i.e. solutions shared with others

DLT network challenges
business challenges (network effect, get & align goals of participants)
technical side (new & old coexist, DB synchronization, many interfaces)