- All probabilities are 50%. Either a thing will happen or it won't.
  – Colvard's Logical Premises
- The fastest I/O is the one that doesn't move any data.
  – Mark Erickson
- "I think she's dead. " - "No I'm not."
  – Monty Python's Flying Circus
- Take away my people but leave my factories and soon grass will grow on the factory floors. Take away my factories and leave my people, and soon we will have a new and better factory.
  – Andrew Carnegie

13 lessen in High Availability

26 september 2011

Johan Loeckx
Sectie Onderzoek

Smals
ICT for society

Agenda / aanpak

- Geörienteerd rond 13 "lessen", vragen stellen mag tijdens de presentatie!
- 3 grote delen:
  - #1 tot #5 – High Availability (HA) op hoog-niveau
    Wat is het, wat zijn de oorzaken en wat kan je eraan doen?
  - #6 tot #9 – De kern van hoog beschikbare systemen
    Wat maakt HA zo moeilijk, wat is het kernprobleem?
  - #10 tot #13 – Geavanceerde technieken
    Overzicht van geavanceerde tips en architecturen

Smals
ICT for society

High Availability is een business probleem!

Smals
ICT for society

#1 – It's the economy, stupid!

High Service Availability

Een service wordt "beschikbaar" genoemd
als het systeem een zinnig antwoord geeft
in een bepaalde tijdspanne

↓

Wat is een zinnig antwoord?

een foutboodschap?
een vorige waarde?
een benaderend resultaat?
een antwoord 1s na time-out?
een antwoord na twee retries?

Spreek dit af met de business stakeholders
tijdens de requirements analyse!

7

Smals
ICT for society

#2 – Stop thinking in "nines"

Metrieken van High Availability

Smals
ICT for society

Enkele definities…

#1 – Meet de downtime D tijdens een service window T

$$Availability = \frac{T - D}{T}$$

#2 – Vaak uitgedrukt als het "aantal negens" in het percentage (T=24/7)

Availability [%]	Downtime per year	Downtime per month	Downtime per week
95%	18.25 days	36 hours	8.4 hours
99% ("two nines")	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.9% ("three nines")	8.76 hours	43.2 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% ("four nines")	52.56 minutes	4.32 minutes	1.01 minutes

Smals
ICT for society

"Het systeem moet 99.9% beschikbaar zijn"

#1 – We zijn van 99 naar 99.5% gegaan, hoera, halfweg!

è helaas… dit was het gemakkelijke deel.
Het moet 5x beter!

#2 – Voor een offline batch process, is 99.9% wel nodig?

è bepaal beschikbaarheid & consistentie per use case

#3 – Eén kritische drop-out van 8u/jaar, of 1.5 min elke dag?

è specificeer de (on)gewenste patronen

Smals
ICT for society

99% vs. 99.9%
Niet echt hetzelfde…

#1 – Organizational shift

- Transparantie à correcte, volledige & up-to-date informatie
- Automatisatie à richting private cloud
- Governance à release, quality & capacity management, …

#2 – Operational shift

- Incidents zijn een no-go!!
- Operations schuiven richting development (DevOps)
- Systemen moeten zichzelf herstellen, zonder downtime

#3 – Statistics shift (P < 0.1%)

- Onwaarschijnlijke toestanden & situaties worden belangrijk
- Vele onbeschikbaarheden zonder downtime tot gevolg
- MTBF & MTTR worden irrelevant
  (Mean-Time-Between-Failure, Mean-Time-To-Repair)

Smals
ICT for society

Lifecycle van een outage / incident

Je krijgt 43 minuten, één keer per maand, om een incident op te vangen:

1. Detecteer het incident à afdoende monitoring
2. Toewijzen van de juiste mensen à beschikbaarheid van de teams
3. Analyse & localiseren v.h. probleem à in een complex omgeving
4. Isoleren van het probleem à e.g. een database down brengen
5. Toewijzen van resources à handleiding / backups zoeken, ...
6. Oplossen & documenteren
7. Verifiëren van de oplossing

Zelfs als elke stap 7 minuten duurt (niet het geval), haal je je 99.9% niet.

Outages / incidenten zijn
een no-go!

Smals
ICT for society

#3 – Who cares about a server?

De oorzaken van onbeschikbaarheid...

Smals
ICT for society

Korte Quiz

Wat zijn je zorgen omtrent beschikbaarheid bij 99.9% systemen?

- overgang naar een nieuwe versie?
- maintenance?
- server crashes?
- bugs?
- race conditions?
- network reconfigurations?

Smals
ICT for society

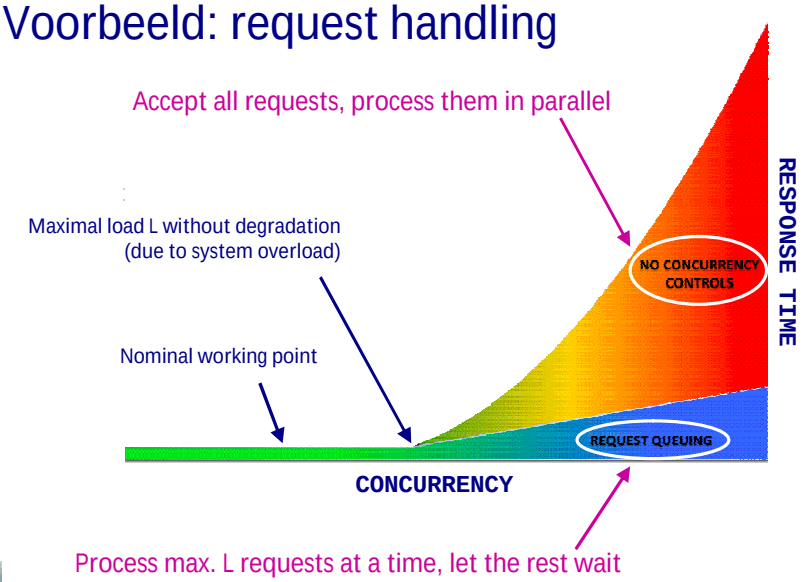
Korte Quiz

Wat zijn je zorgen omtrent beschikbaarheid bij 99.9% systemen?

- ~~overgang naar een nieuwe versie~~
- ~~maintenance~~
- ~~server crashes~~
- 'bugs' à ja, maar geen functionele
- race conditions → Sub-optimaal
  - Connection management
  - Memory management
  - Request handling
  - ...
- network reconfigurations

Smals
ICT for society

Voorbeeld: request handling



Smals
ICT for society

Hou een service / server in zijn nominaal werkingspunt, waar het gedrag voorspelbaar is.
(bv. om het verschil te merken tussen een server die dood is, of overbelast)

Oorzaken van onbeschikbaarheid

#1 – Gebrek aan governance

- Configuration errors
  Quota exceeded; network flux blocked; invalid certificate; wrong creds, ...
- Menselijke fouten
  Service vergeten te starten; de enige met kennis, is weg...

#2 – Hardware/software

- Software / Hardware falen
  Memory leak / Single-Point-Of-Failure (e.g. SAN, Reverse Proxy, LB, ...)
- Overgangseffecten
  Race condition, locked record, lost network packet

#3 – Externe factoren

- Externe service down
  Internet Service Provider, Rijksregister (RR/RN)
- Denial-of-service attack / Disasters

Bekijk de hele lifecycle van de service,
vanuit data, process & technisch perspectief

Smals
ICT for society

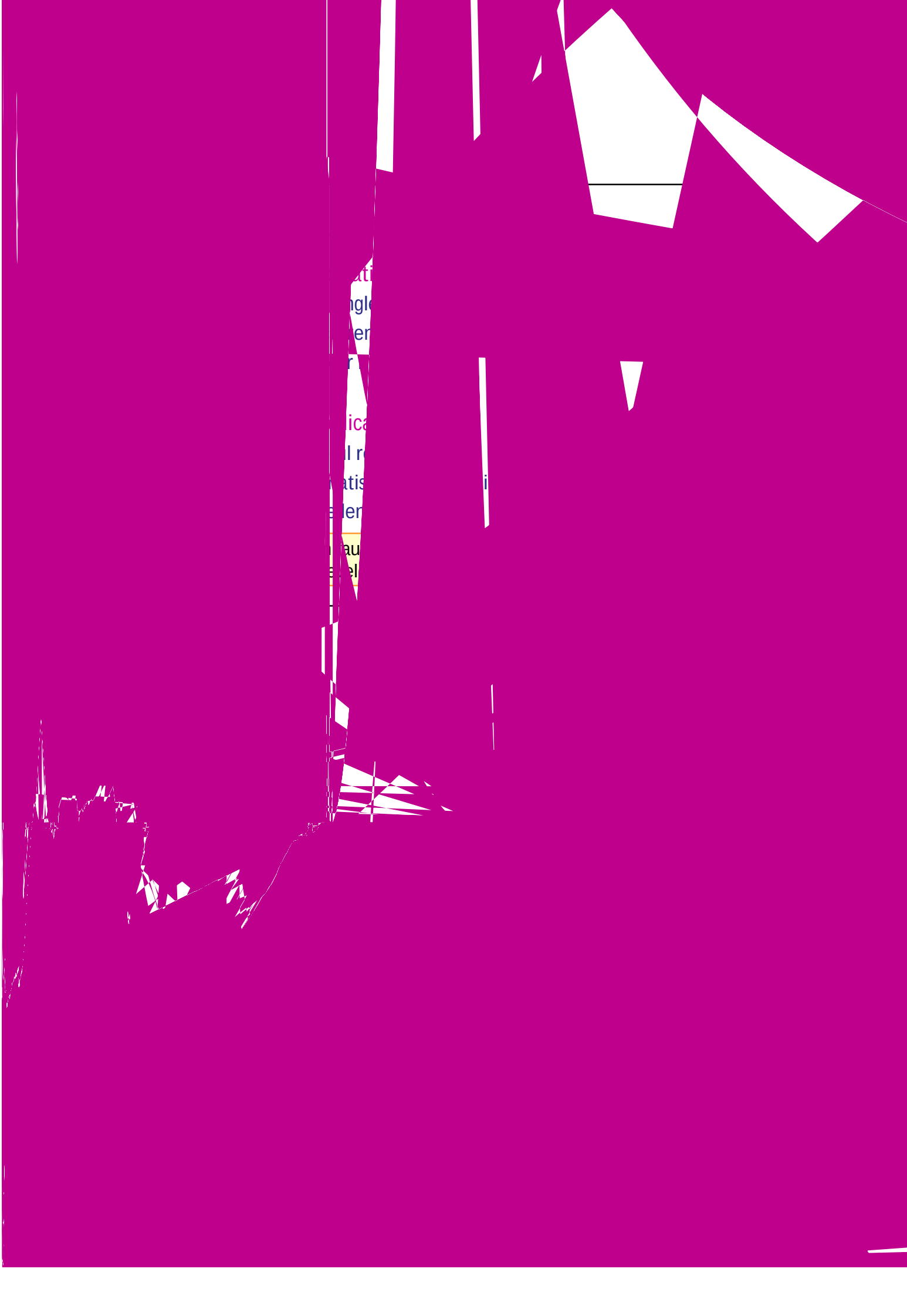
#4 – Hope for the best… but plan for the worst

Everything will happen

Smals
ICT for society

Stories

- Load Balancer detecteert dat de server up is, maar de service werkt niet. Omdat de Load balancer dat niet detecteert (application level), faalt de helft van de requests. Officieel is A=100%, realiteit=50%.
- Een connectie wordt geopend maar niet gesloten. Na 5s sluit de client de connectie. Bij hoge loads wordt het max aantal connecties bereikt en faalt de service (variant: object niet verwijderd).
- Netwerkcongestie of een overbelaste server veroorzaakt een time-out die een oneindige lus van verzenden / aanvragen van data in gang zet.
- Deadlocks: proces A lockt tabel 1 voor schrijven, proces B lockt tabel 2. Proces A vraagt toegang tot tabel 2, proces B tot tabel 1. Beide wachten onbeperkt tot de lock vrijgegeven is.
- Code aangeduid met `/* code not reached */` wordt uitgevoerd.



Smals
ICT for society

"One of the first systems our engineers built in AWS is called the Chaos Monkey. The Chaos Monkey's job is to randomly kill instances and services within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage. "

25

Smals
ICT for society

#5 – Learn from the best

De brandweer

26

Smals
ICT for society

Als luchtvaartmaatschappijen en de brandweer het kunnen, waarom u dan niet?

- Goed opgeleid & ervaren personeel
- Duidelijke verantwoordelijkheden & taken
- Vertrouwen in de werknemers (firefighters / pilots)
- Goed-gedocumenteerde procedures
- Flexibele & effectieve escalaties
- Voortdurende feed-back van onderuit
- Co-piloot è 4 ogen principe
- Inspectie van voorzieningen & procedures

Negeer lange-termijn/gevoelige problemen niet!

Smals
ICT for society

Mensen en processen

#1 – Transparantie

- Absolute noodzaak voor lange-termijn beschikbaarheid
- Ga tot op het bot: versta écht waarom iets faalt. Niet stoppen tenzij a) je logging moet verhogen b) je het probleem vindt.
- Documenteer voor als iemand het bedrijf verlaat / ziek wordt.

#2 – Governance

- Beheers Kwaliteit, Releases, Changes, leveranciers, etc...
- Detecteer problemen voor het te laat is (bv. memory leaks)
- Doe geen 'patch work' / bypasses (rebooten is GEEN oplossing)

#3 – Automatisatie

- Mensen maken fouten, zelfs ondanks het 4 ogen principe

Smals
ICT for society

Smals
ICT for society

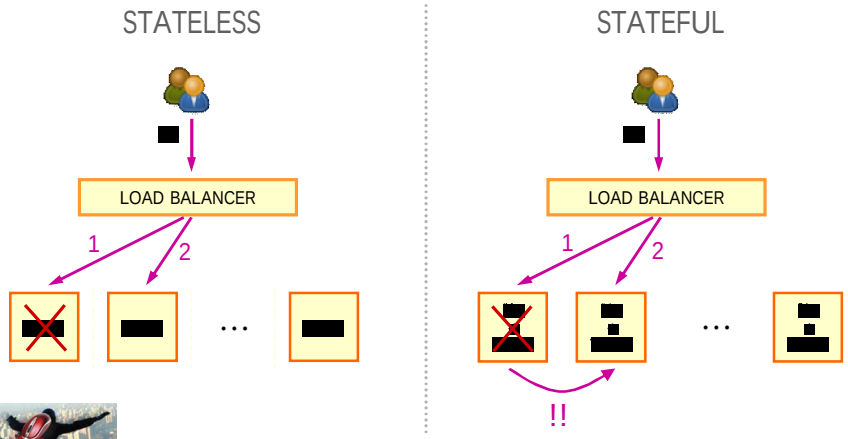
#6 – Understand your data

Availability-in-depth

Smals
ICT for society

Fundamenteel probleem van availability

Je gegevens krijgen waar de berekening plaatsvindt



Smals
ICT for society

Vluchtigheid & Lokaliteit

#1 – Vluchtigheid

- Hoe vaak verandert de data, waar en wanneer? (write)
- Begin de analyse reeds tijdens de business requirements
- Bv. session state/undo info vs. stored password

#2 – Lokaliteit

- Waar heb je de data nodig en wanneer? (read)
- Sterk afhankelijk van de architectuur (ook van de DB)!!!
- Bv. Sessie informatie enkel relevant voor de betrokken gebruiker

Dé uitdaging is gedeelde data die snel verandert!
Gelukkig is vluchtigheid & lokaliteit typisch een trade off…

Smals
ICT for society

Availability-in-depth

een holistische aanpak

#1 – Requirements analyse

- Bepaal de availability voor elke use case en gegevenselement
- Analyseer de vluchtigheid & lokaliteit van de data
- Analyseer de benodigde consistentie voor elk stuk data

#2 – Architectuur

- Behoud de lokaliteit & vluchtigheid van de data in het design
- Ontkoppel, keep-it-simple, elimineer alle SPOFs
- ACID vs. BASE afhankelijk van de gewenste consistentie

Smals
ICT for society

(ACID vs. BASE)

ACID: "traditional relational transactional systems"

- Atomic → ofwel slaagt/faalt de hele transactie
- Consistent → iedereen bezit dezelfde data
- Isolated → geen interactie tussen transacties
- Durable → fire-and-forget

BASE: "modern, custom, scalable & available systems"

- Basically → light-weight, simpel
- Available → steeds een antwoord klaar
- Soft-state → optimistische locking, conflicten kunnen
- Eventually Consistent → niet altijd dezelfde data

DNS, Amazon bookstore, web caches,...

Smals
ICT for society

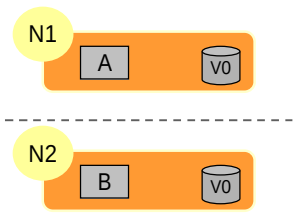
#7 – CAP is GOD

Consistentie & Availability zijn een trade-off

Smals
ICT for society

Het CAP theorema

In een gedistribueerd systeem,



gegeven:

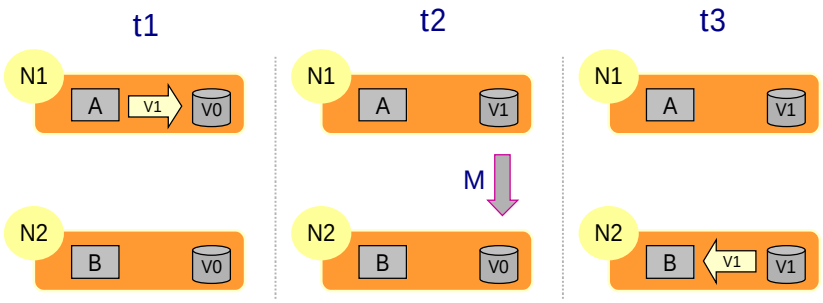
- Consistency: antwoorden van N1 en N2 zijn steeds gelijk
- Availability: ik krijg steeds een antwoord, van N1 en N2
- Partition tolerance: ongevoelig aan netwerk onderbrekingen

geldt het volgende theorema:

Consistency, Availability and Partition tolerance kunnen niet alle drie te allen tijde gegarandeerd worden

Smals
ICT for society

#1 – Alles draait zoals het moet



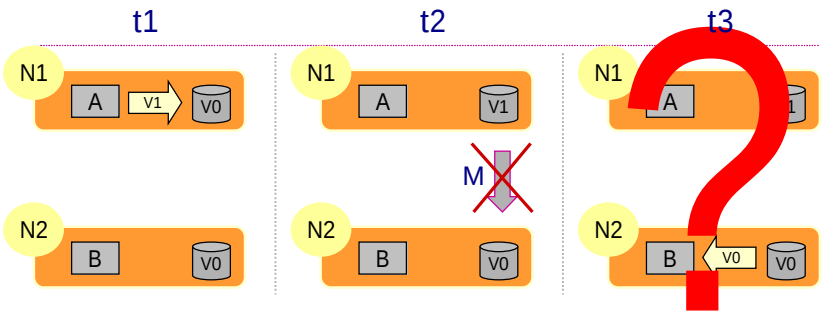
Het CAP theorema is irrelevant
als alles draait zoals het moet…

#7 – CAP is GOD

39

Smals
ICT for society

#2 – Het netwerk faalt



Transactie gaat door è Available maar inconsistent
Transactie annuleren è Unavailable maar consistent

40

Smals
ICT for society

Gevolgen

Als u consistency wilt, moet u ofwel:

- Availability laten varen, of
- Falen als er een netwerkprobleem is

Wenst u availability, dan heeft u de keuze:

- Consistency laten varen, of
- Falen als er een netwerkprobleem is

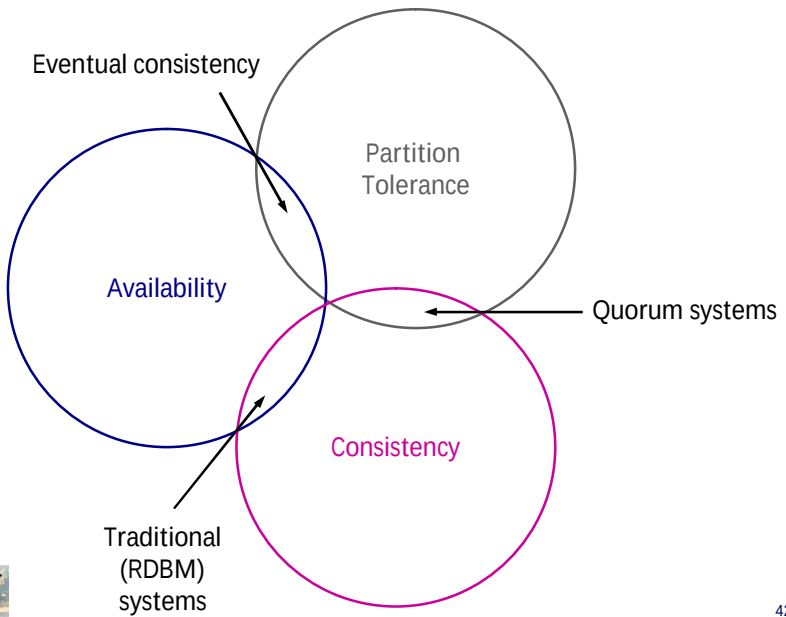
Netwerkproblemen treden op, dus ofwel:

- ~~Loose Availability~~
- Loose Consistency

Smals
ICT for society

De CAP trade-off

Eric Brewer, 1997



Smals
ICT for society

#8 – You have 234 ~~233~~ friends

Leven met inconsistentie

Smals
ICT for society

Leven met inconsistentie

#1 – De oplossing: "eventual consistency"

- In het voorbeeld, return V0, update N2 a.s.a.p.
- "Optimistic locking", los de weinige conflicten later op
- Typisch een kwestie van seconden

Varianten: session consistency, Read-your-writes consistency, Monotonic read/write consistency, …

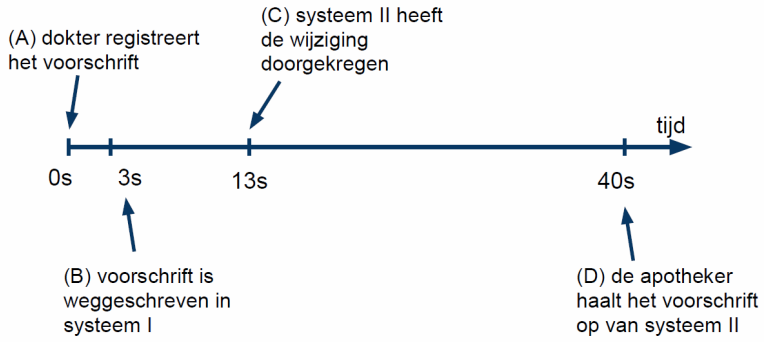
#2 – Vaak stelt dit geen probleem

- De business laat het toe
  (bv. Amazon # copies left, Facebook #friends, …)
- Business time en IT time zijn twee concepten!

Smals ICT for society

Business time <> IT time

#1 – Business time



#2 – IT time

- Milliseconden
- Synchrone transacties

Smals ICT for society

"Wij aanvaarden geen inconsistentie!" maar wacht:

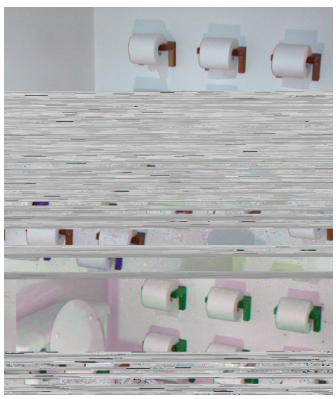
De inconsistentie is enkel aanwezig op die momenten dat een traditioneel systeem down zou zijn!

En het probleem wordt opgelost binnen tien seconden.

47

Wat als...

- § je 25% papier wil toevoegen?
- § een papierhouder afbreekt?
- § je het systeem moet uitleggen?



#9 – Kill Complexity before it kills you

Smals ICT for society

Korte Quiz

Hoe groot is het korte termijngeheugen van de mens?

- 1 bit?
- 3 bit?
- 3 byte?
- 3 kilobyte?

2.5 bit

Mensen kunnen slechts onderscheid maken tussen 7 ± 2 dingen tegelijk (Miller, 1956)

Bv. Aantal titels in een inhoudsopgave, #verschillende toonhoogtes, geluidsterktes, smaken, stereo geluidsposities, #elementen in een architectuur, punten op een lijn, …

Smals ICT for society

Je kunt niet controleren wat je niet begrijpt!

#1 – Vertrouw niets dat >7 items heeft (bv. 13…)

#2 – Kruip uit je hoek

- "Over de wall" design doet complexiteit exploderen
- Complexiteit ontstaat door interacties tussen blokken

#3 – Bekijk de volledige keten

- een object wordt gepersisteerd a.d.h.v. Hibernate, neemt geheugen in, heeft impact op de middleware en database transacties, suboptimale queries à besef de gevolgen!

#4 – Geen quick hacks of bypasses (ze gaan niet weg!)

#5 – DevOps kan helpen

#9~~K~~Kill Complexity before it kills you

Smals
ICT for society

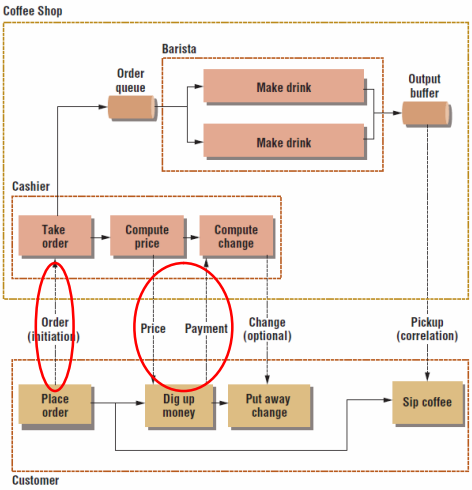
#10 – Forget what you have learned

Asynchronicity & replication

52

Forget what you've learnt (1/2)

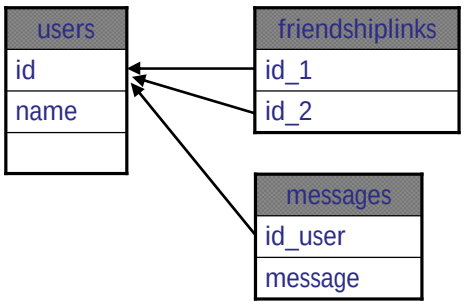
#1 – Asynchroon ipv. synchroon (vaak een mix)



- Slechts 2 synchrone acties; enkel om te bevestigen: "goed ontvangen"
- Geen onbeschikbaarheid als Barista een rookpauze houdt
- Throughput kan veel hoger
- Als klant geen geld heeft
  è kopje weggooien
- Als order ongeldig
  è retry…

Smals
ICT for society

Stel dat facebook database normalizatie gebruikte (1/3)



Smals
ICT for society

Stel dat facebook database normalizatie gebruikte (2/3)

- SQL statement to get the messages on my wall:

```
SELECT message FROM messages
  INNER JOIN friendshiplinks
  ON (((friendshiplinks.id_1 = messages.id_user) AND
       (friendshiplinks.id_2 = myid)
      ) OR (
       (friendshiplinks.id_2 = messages.id_user) AND
       (friendshiplinks.id_1 = myid)))
```

- Looks alright?

Smals
ICT for society

Stel dat facebook database normalizatie gebruikte (3/3)

- 700M users met gemiddeld 135 vrienden…

  Met 50% overlap, krijgen we een friendshiplink table met 47G rows.

- 30G messages per maand…

  Na één maand:
  JOINs over tabellen met 30G en 47G rijen.

Smals
ICT for society

Stel dat facebook geen replicas had, en synchrone calls naar de DB deed

- 350 miljoen gebruikers checken elke dag minstens éénmaal hun profiel. Dit bekent:
  4000 JOINS per seconde tussen 2 tabellen van 40G
- Oh ja. Deze tabel wordt 11500x per seconde geüpdatet.
- Of: 64 µs per transactie (inclusief netwerk).

Smals
ICT for society

Hoe dan (waarschijnlijk) wel?

- Partitionering
  - Niet alle data in één databank, maar opdelen op basis van user id (bv. 26k tabellen van 26k users)
  - Maar hoe de gegevens tussen users combineren?
- Denormalizatie & duplicatie
  - Dezelfde gegevens herhaald opslaan (vriendenlijst 2x bijhouden, messages bij elke user bijhouden, etc…)
  - Nog steeds JOINS… En hoe 26k tabellen orchestreren?
- En verder…
  - Extreme in-memory caching (40 TB), UDP ipv TCP, active replication, tuning IRQ handling & OS writes

Smals
ICT for society

#11 – Architecture is your friend

On scale-out, decoupling and toiletpaper

Smals ICT for society

Architecturale principes

#1 – Keep it simple, stupid

- Begrijp de interacties tussen de verschillende elementen van de stack en de verschillende componenten in de architectuur
- bv. connectie afgesloten in applicatie maar niet in application server

#2 – Redundantie

- Anticipeer op component-falen en diversifieer de oorzaken ervan
- Elimineer Single-Points-Of-Failure (SPOFs): GSM netwerk, mensen,...

#3 – Ontkoppeling

- Beperk de impact & verspreiding een faling

#4 – Geef de voorkeur aan horizontale architecturen

- Deze zijn robuuste & schaalbare architecturen, gemakkelijker los te koppelen

Smals ICT for society

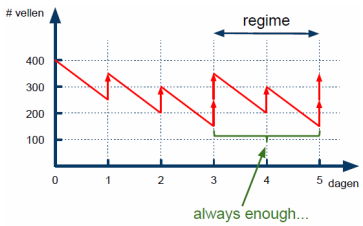
Scale out vs. Scale up

Scale up (vertical)

- Een "grotere" machine kopen
- Intrinsieke downtime
- Inflexibel & duur maar simpel

Scale out (horizontal)

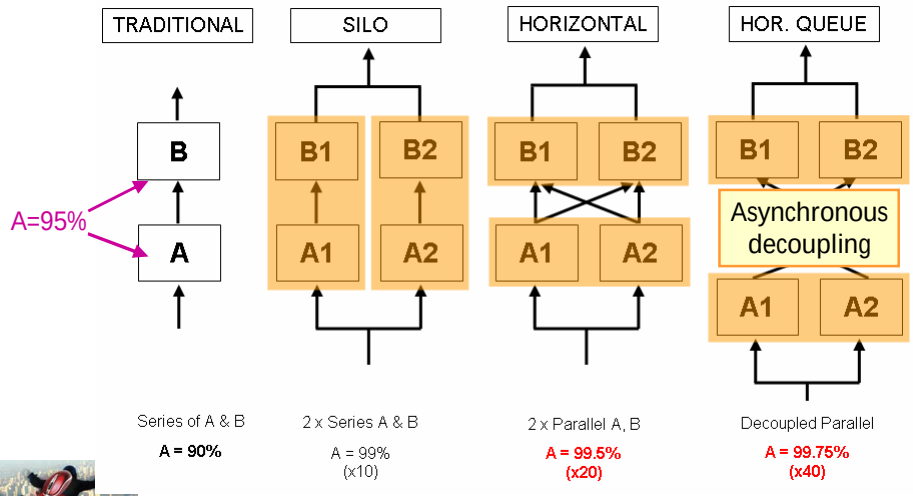
- Een "kleine" commodity server
- Geen downtime geassocieerd
- Flexibel; architectuur moet het wel ondersteunen



(b) 4x Roll with 100 papers

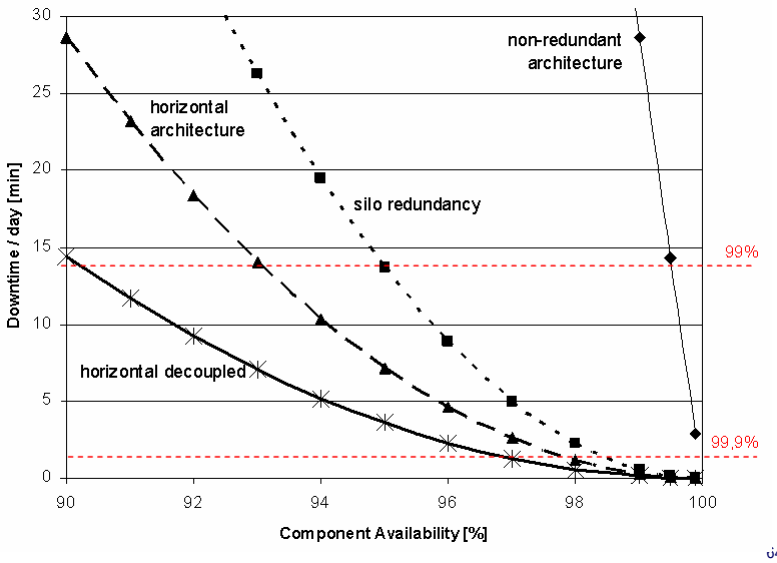
Smals
ICT for society

Vergelijking tussen 4 architecturen (1/2)



Smals
ICT for society

Vergelijking tussen 4 architecturen (2/2)



Smals
ICT for society

#12 – Don't implement failover

6 techniques to improve Availability

Smals
ICT for society

6 tips & tricks to improve availability

#1 – Zet je ego aan de kant
Om de root cause te vinden, moet iedereen meewerken.

#2 – Eén probleem, één oplossing
Doe één ding per keer, grondig.

#3 – Gebruik bewezen oplossingen
Probeer dit niet zelf: protocols, logging, databases, transaction managers,...

#4 – Begrijp je stack
Manage je connecties, requests, errors, ... doorheen de gehele stack

#5 – Hergebruik configuraties
Probeer configuraties te standardiseren na een grondige domeinanalyse

#6 – DevOps
Sloop de muren tussen Development & Operations

Smals
ICT for society

#13 – Technology can help

NoSQL, XTP & Languages

Smals
ICT for society

noSQL databases

Traditionele database systemen

- Relationeel model wordt afgedwongen
- Focus op consistentie; geörienteerd rond transacties

noSQL = not only SQL

- Simpel model: key-value, column store, doc store, graph
- Ongeëvenaarde performantie, availability & scalability!!!

Wie gebruikt het?

- High-end applications: Twitter, Facebook, LinkedIn, Sourceforge,...

What's the catch?

- Meer werk in programmatie / integratie
- Gebrek aan 3rd party tools (operations, monitoring, etc...)

XTP platforms

XTP = eXtreme Transaction Processing

- Opvolger van DTP application platforms
- Nieuwe manier voor de ontwikkeling van distributed transaction processing applications
- Evolueert richting cloud-enabled application platforms (CEAP)

Wat is het verschil met gewone Application Servers?

- Hoge performance, scalability, elasticity, availability
- Simpele, transparante en automatische fail-over
- Houdt de data & processing samen (!!), in-memory

Wie gebruikt het?

- Stock exchange, investment banking, telecommunication sector, social media, online gaming, ...

Programming languages

Vb. Erlang = high-level functionele programmeertaal

- fault-tolerant, soft-realtime, concurrent!
- Ondersteunt hot-swapping
- Werd oorspronkelijk gebruikt bij Ericsson
- Message-passing ipv shared variables

Alleen in missie-kritische kerntoepassingen!!

Wie gebruikt dit?

- Telecom & switching
- noSQL databases zoals CouchDB, SimpleDB (Amazon),...

Ericsson AXD301 ATM switch:

99.9999999% reliability (9 nines) (31 ms. per jaar!)

Smals
ICT for society

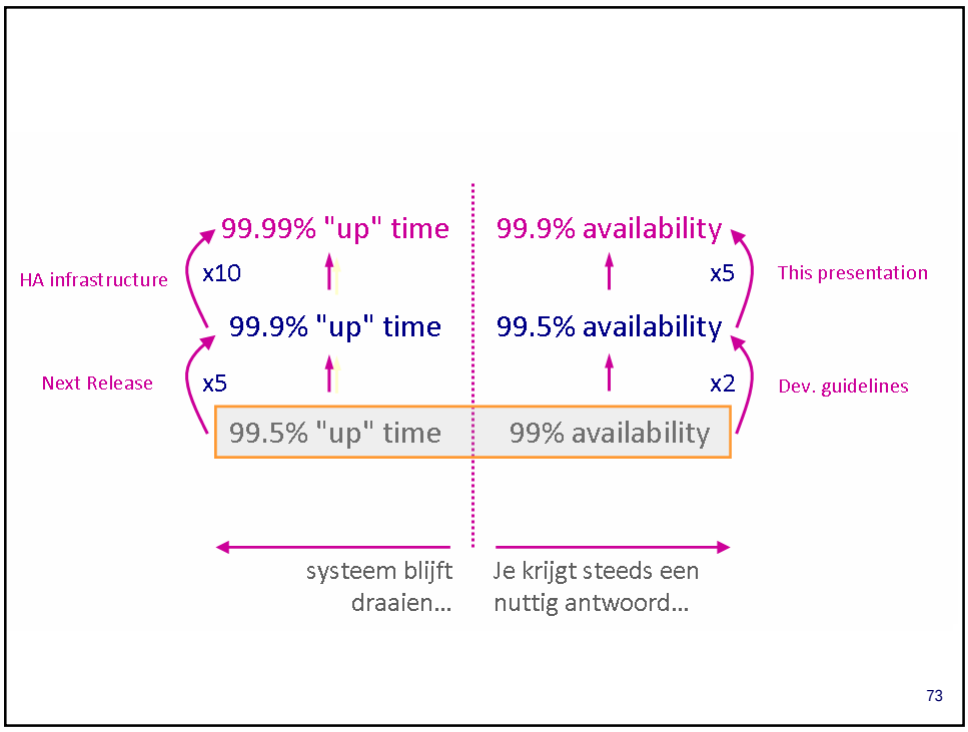
En hoe zit het bij Smals?

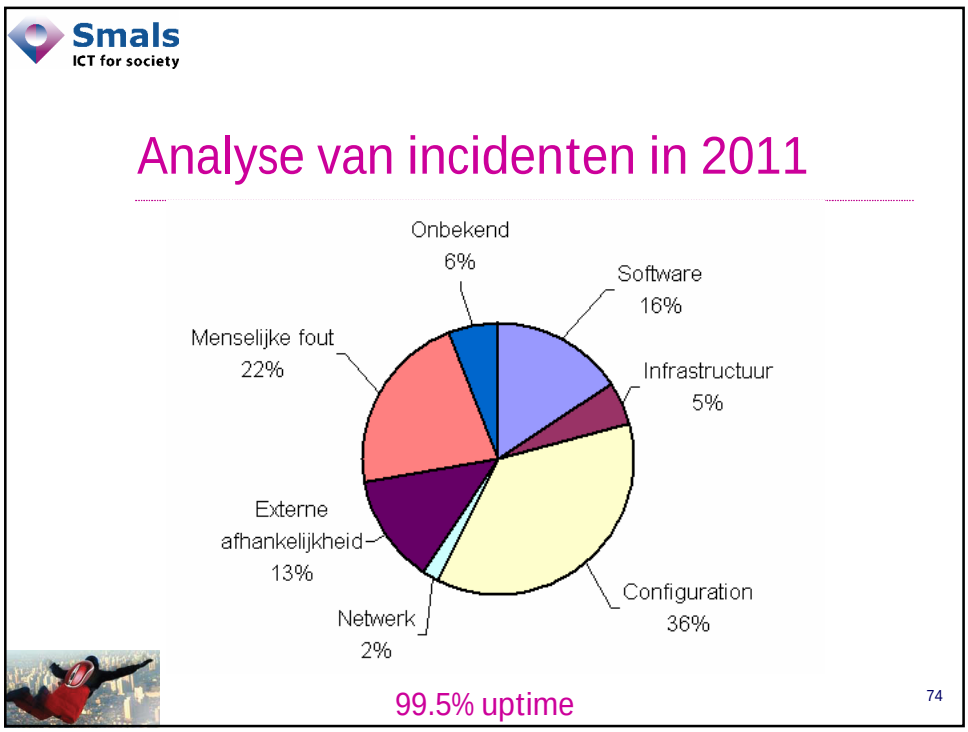
Initiatieven

Smals
ICT for society

Initiatieven binnen Smals

- Service Level Agreements (aflijnen met de klant)
- Redundante infrastructuur (2N)
- Next Release environment (no planned downtime)
- Development guidelines
- Security Review
- Quality Asssurance
- Change, Problem, Release, Capacity management


99.99% "up" time
99.9% availability
HA infrastructure
x10
x5
This presentation
99.9% "up" time
99.5% availability
Next Release
x5
x2
Dev. guidelines
99.5% "up" time
99% availability
systeem blijft draaien...
Je krijgt steeds een nuttig antwoord...
73


Smals
ICT for society
Analyse van incidenten in 2011
Onbekend 6%
Software 16%
Menselijke fout 22%
Infrastructuur 5%
Externe afhankelijkheid 13%
Configuration 36%
Netwerk 2%
99.5% uptime
74

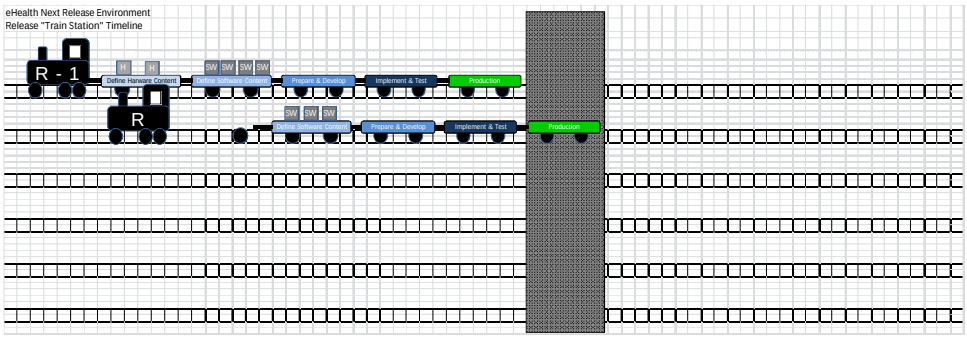
Smals
ICT for society

Next Release

- Next Release biedt 4 redundante omgevingen aan:
  è N-1, N, N+1 Releases and DRP-N
- Om de naadloze overgang van de ene naar de andere versie toe te laten
- Bevat ook release management: kalender wanneer features gekozen, geïmplementeerd en getest worden
- Een release omvat de gehele stack: HW, firmware,MW , data & applicatie

Smals
ICT for society

Next Release Lifecycle



Smals
ICT for society

Te onthouden

- High Availability is een business probleem
- Test alles, zeker ook je NFRs (niet-functionele requirements)
- 10x hogere beschikbaarheid è 10x hogere kost
- De hele keten & stack beschouwen
- Samenwerking tussen Dev en Ops vereist!
- Complexiteit steeds in het oog houden!!
- Governance cruciaal: transparantie, automatisatie,...
- Asynchrone communicatie kan veel opvangen
- Fundamenteel probleem: data bij berekening krijgen

Smals
ICT for society

Literatuur

- Links
  - Canned Platypus
    http://pl.atyp.us/
  - Facebook Engineering Tech Talks
    http://www.facebook.com/Engineering
  - Amazon downtime report
    http://aws.amazon.com/message/65648/
  - Netflix's ChaosMonkey
    http://www.codinghorror.com/blog/2011/04/working-with-the-chaos-monkey.html

- Boeken & documenten
  - High Availability Concepts
    Dienst Onderzoek, Johan Loeckx, in bijlage
  - Blueprints for High Availability
    Smals bibliotheek, Evan Marcus & Hal Stern
  - "Your coffee shop doesn't use two-phase commit"
    IEEE Software, Gregor Hohpe, March-April 2005, page 64-66

#1 – It's the economy, stupid! – High Service Availability
#2 – Stop thinking in "nines" – Metrieken van High Availability
#3 – Who cares about a server? – De oorzaken van onbeschikbaarheid
#4 – Hope for the best... but plan for the worst – Everything will happen
#5 – Learn from the best – De brandweer
#6 – Understand your data – Availability-in-depth
#7 – CAP is GOD – Consistentie & Availability zijn een trade-off
#8 – You have 234 friends – Leven met inconsistentie
#9 – Kill complexity before it kills you – Really, simple is good!
#10 – Forget what you have learned – Asynchronicity & replication
#11 – Architecture is your friend – On scale-out, decoupling and toiletpaper
#12 – Don't implement failover – 6 techniques to improve Availability
#13 – Technology can help – NoSQL, XTP & Languages

79

Smals
ICT for society

Vragen?

80