MICROSERVICES TENETS: AGILE APPROACH TO SERVICE DEVELOPMENT AND DEPLOYMENT (A.K.A. SOA VS. MICROSERVICES)

10th Symposium and Summer School
On Service-Oriented Computing

Prof. Dr. Olaf Zimmermann (ZIO)
Distinguished (Chief/Lead) IT Architect, The Open Group
Institute für Software, HSR FHO
Hersonissos, June 29, 2016

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

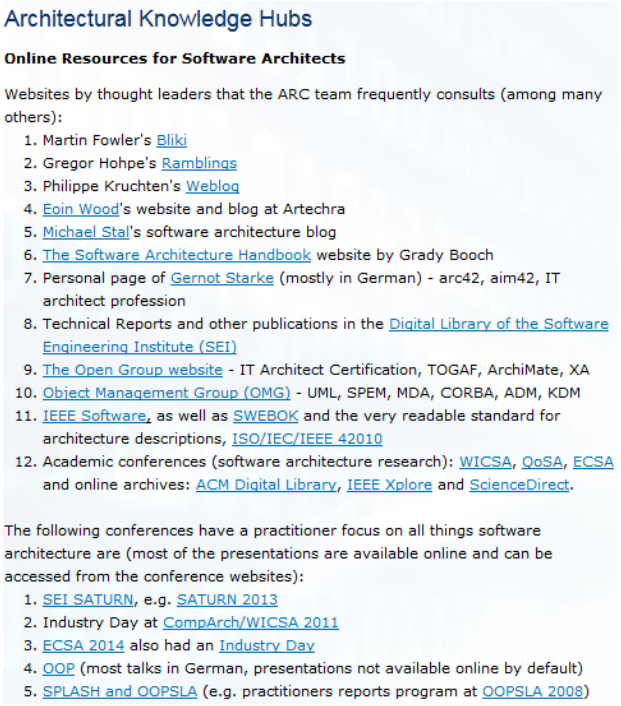
ZIO Past and Present

The Open Group Open Certified Distinguished Architect

IT Architecture Chief/Lead IT Architect

- **Research & development and professional services since 1994**
  - em. IBM Solution Architect & Research Staff Member
    - Systems & Network Management, J2EE, Enterprise Application Integration/SOA
  - em. ABB Senior Principal Scientist
    - Enterprise Architecture Management/Legacy System Modernization/Remoting
- **Selected industry projects and coachings**
  - Product Development and IT Consulting (Middleware, SOA, information systems, SE tools)
  - Tutorials: UNIX/RDBMS, OOP/C++/J2EE, MDSE/MDA, Web Services/XML
- **Focus @ HSR: design of distributed/service-oriented systems**
  - Cloud Computing, Web Application Development & Integration (runtime)
  - Model-driven development, architectural decisions (build time)
  - (Co-)Editor, Insights column, IEEE Software
  - PC member, e..g ECSA, WICSA, QoSA, SATURN, SummerSoC

- **What do architects do anyway? Analyze NFRs/QAs and satisfy them!**
  - Systematic end-to-end systems thinking; abstractions and generalizations; modelling
- **Why bother about architecture?**
  - Manage complexity, initiate and steer projects, coach developers
- **Where can I find more information?**



Architectural Knowledge Hubs

Online Resources for Software Architects

Websites by thought leaders that the ARC team frequently consults (among many others):

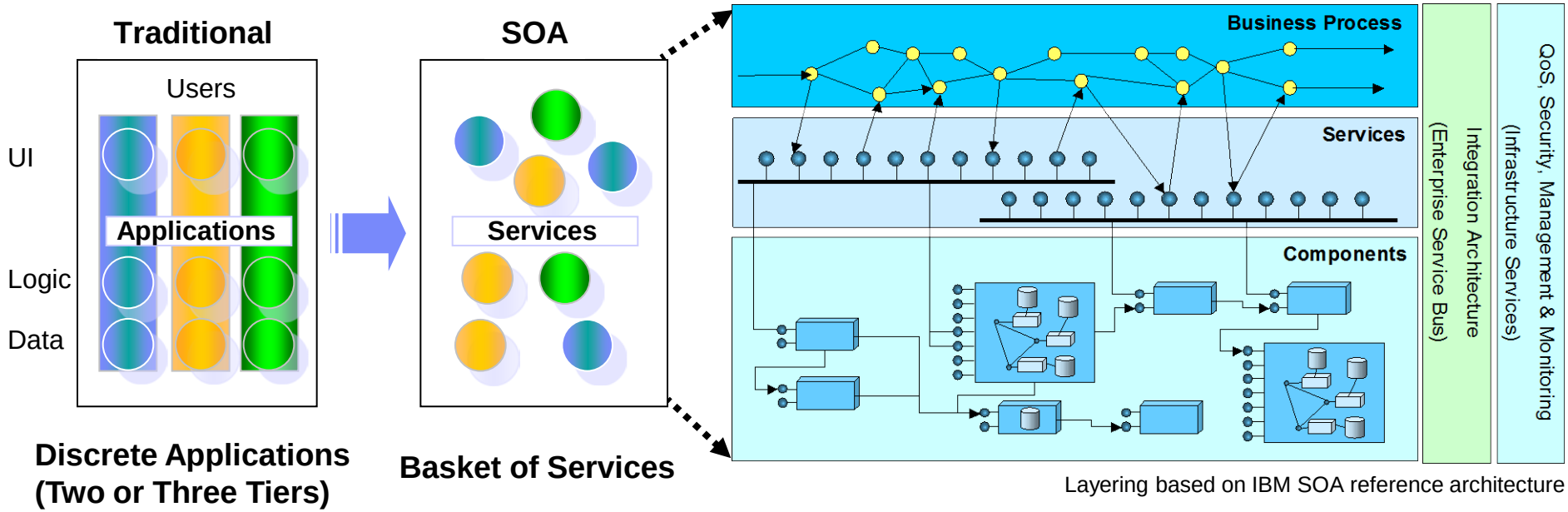
1. Martin Fowler's Bliki
2. Gregor Hohpe's Ramblings
3. Philippe Kruchten's Weblog
4. Eoin Wood's website and blog at Artechra
5. Michael Stal's software architecture blog
6. The Software Architecture Handbook website by Grady Booch
7. Personal page of Gernot Starke (mostly in German) - arc42, aim42, IT architect profession
8. Technical Reports and other publications in the Digital Library of the Software Engineering Institute (SEI)
9. The Open Group website - IT Architect Certification, TOGAF, ArchiMate, XA
10. Object Management Group (OMG) - UML, SPEM, MDA, CORBA, ADM, KDM
11. IEEE Software, as well as SWEBOK and the very readable standard for architecture descriptions, ISO/IEC/IEEE 42010
12. Academic conferences (software architecture research): WICSA, QoSA, ECSA and online archives: ACM Digital Library, IEEE Xplore and ScienceDirect.

The following conferences have a practitioner focus on all things software architecture are (most of the presentations are available online and can be accessed from the conference websites):

1. SEI SATURN, e.g. SATURN 2013
2. Industry Day at CompArch/WICSA 2011
3. ECSA 2014 also had an Industry Day
4. OOP (most talks in German, presentations not available online by default)
5. SPLASH and OOPSLA (e.g. practitioners reports program at OOPSLA 2008)

(screen caption clickable)

Partitioning into Components and Services (Example)



Example:
An insurance company uses three SAP R/3, MS Visual Basic, and COBOL applications to manage customer information, check for fraud, and calculate payments. The user interfaces (UIs) are the only access points.

A multi-step, multi-user business process for claim handling, executing in IBM WebSphere, is supposed to reuse the functions in the existing applications. How to integrate the new business process with the three legacy applications in a flexible, secure, and reliable way?

What is SOA? (Source: OOPSLA Tutorials 2004-2008)

No single definition – “SOA is different things to different people”

- A *set of services* that a business wants to expose to their customers and partners, or other portions of the organization.

Business Domain Analyst

- An architectural style which requires a *service provider*, a *service requestor* (consumer) and a *service contract* (a.k.a. client/server).
- A set of architectural patterns such as *enterprise service bus*, *service composition*, and *service registry*, promoting principles such as *modularity*, *layering*, and *loose coupling* to achieve design goals such as separation of concerns, reuse, and flexibility.

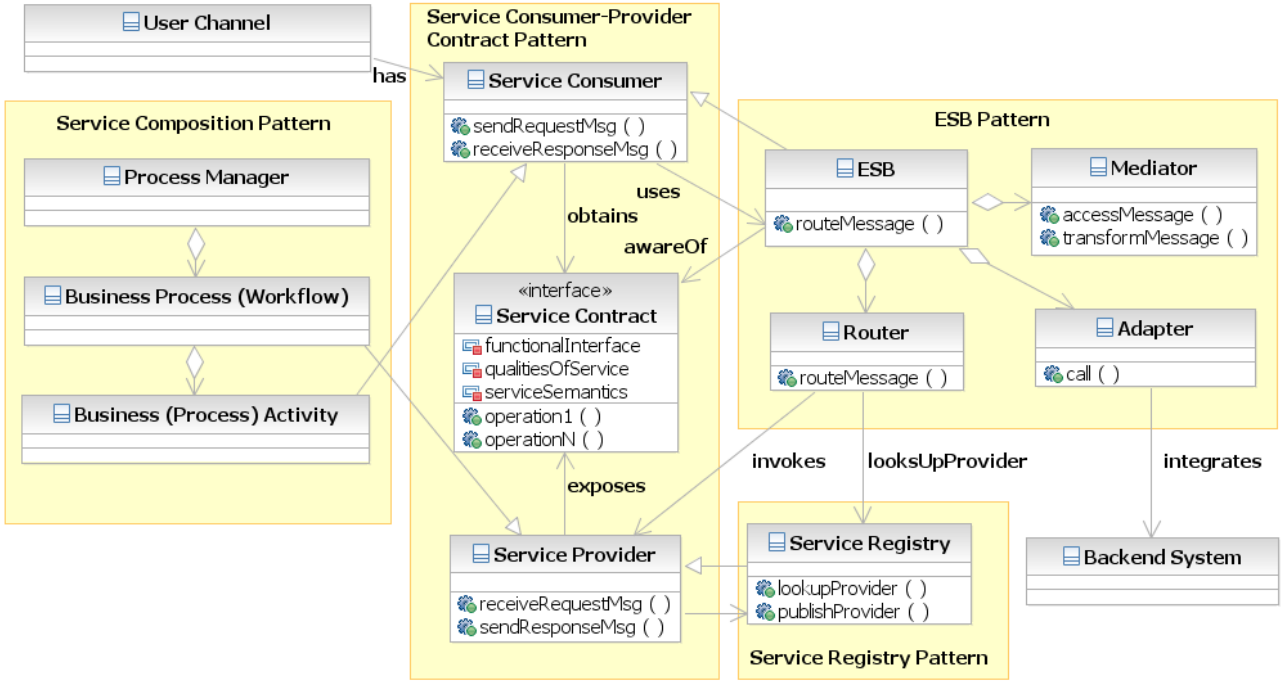
IT Architect

- A *programming and deployment model* realized by standards, tools and technologies such as Web services and Service Component Architecture (SCA).

Developer, Administrator

Adapted from: [IBM SSS]

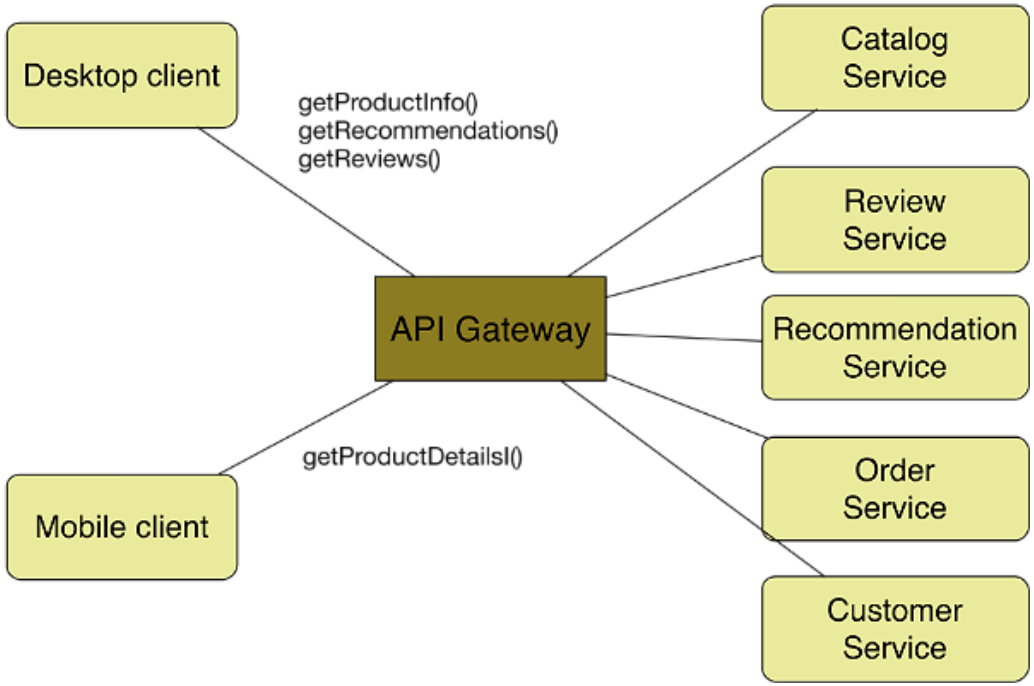
- **Service Contract, Enterprise Service Bus (ESB), Service Composition**



Reference: O. Zimmermann, An Architectural Decision Modeling Framework for Service-Oriented Architecture Design, dissertation.de, 2009 (ISBN: 3-540-00914-0).

Example of a Microservices Pattern: API Gateway

- **Intermediary process (in services LAN)**
- **Client-specific APIs wrapping services**
  - Coarse-grained
  - Fine-grained
- **Allows services and clients to evolve independently**



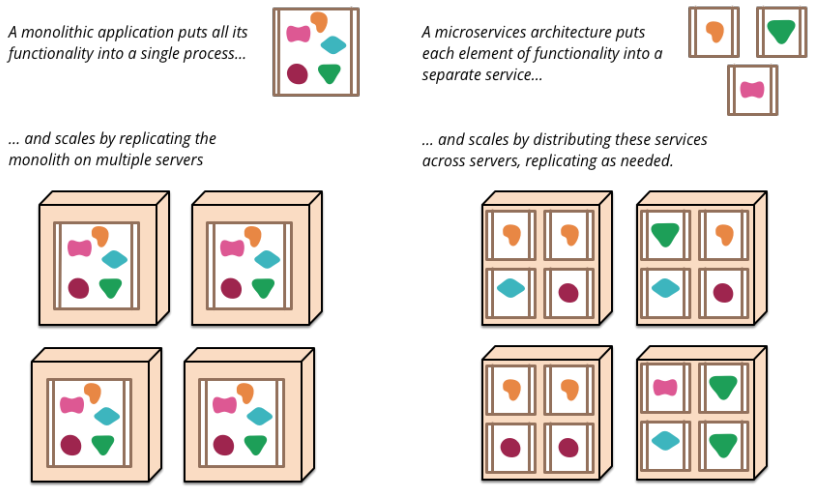
Which patterns (GoF, PoEAA and EIP) does this design resemble? What is the connection to Domain-Driven Design (DDD)?

Reference: http://www.infoq.com/articles/microservices-intro and http://microservices.io/patterns/apigateway.html

Microservices – An Early and Popular Definition (2014)

Reference: http://martinfowler.com/articles/microservices.html

- **J. Lewis and M. Fowler (L/F): "The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies."**



Characteristics from L/F Definition Analyzed and Compared

Characteristic	Viewpoint/Qualities/Benefit	SOA Pendant
Componentization via services	Logical Viewpoint (VP): separation of concerns improves modifiability	Service provider, consumer, contract (same concept)
Organized around business capabilities	Scenario VP: OOAD domain model and DDD ubiquitous language make code understandable and easy to maintain	Part of SOA definition in books and articles since 200x (e.g. Lublinsky/Rosen)
Products not projects	n/a (not technical but process-related)	(enterprise SOA programs)
Smart endpoints and dumb pipes	Process Viewpoint (VP): information hiding improves scalability and modifiability	Same best practice design rule exists for SOA/ESB (see e.g. here)
Decentralized governance	n/a (not technical but process-related)	SOA governance (might be more centralized, but does not have to; “it depends”)
Infrastructure automation	Development/Physical VP: speed	No direct pendant (not style-specific, recent advances)
Design for failure	All VPs: robustness	Key concern for distributed systems, SOA or other
Evolutionary design	n/a (not technical but process-related): improves replaceability, upgradeability	Service design methods, Backward compatible contracts

Principles of Microservices by S. Newman (N)

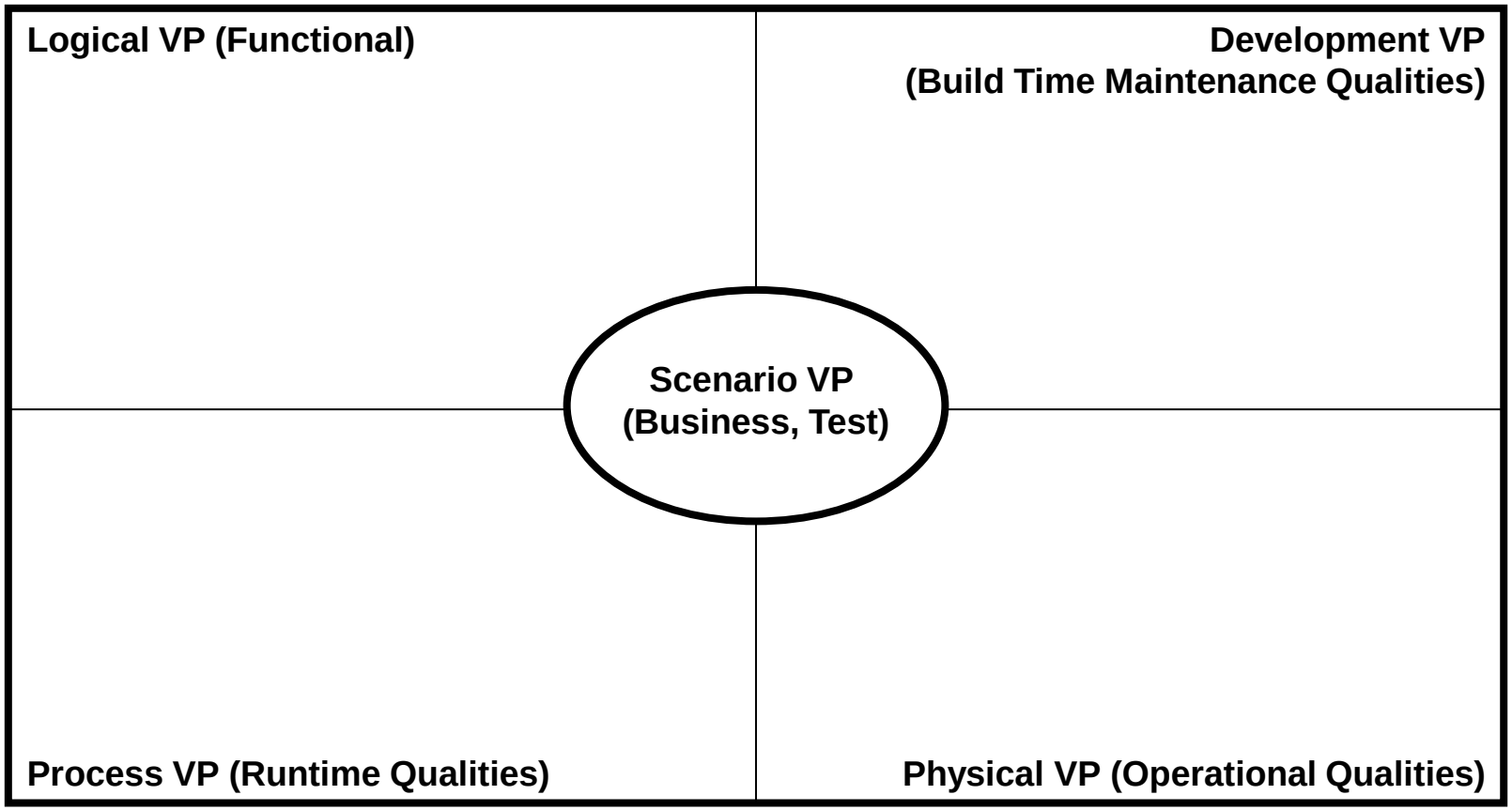
- **Model around Business Concepts**
  - Bounded contexts from DDD
- **Adopt a Culture of Automation**
  - Testing, deployment; continuous delivery
- **Hide Internal Implementation Details**
  - E.g. hide databases; define technology-agnostic APIs
- **Decentralize All the Things**
  - E.g. shared governance, choreography over orchestration, dumb middleware with smart endpoints
- **Independently Deployable**
  - Let versioned endpoints co-exist; one service per host
- **Isolate Failure**
  - E.g. introduce circuit breakers
- **Highly Observable (Semantic Monitoring with Aggregation)**

Newman Definition Analyzed and Compared with SOA

- **Classification of the seven Newman (N) principles of microservices by viewpoint/quality and SOA pendant:**

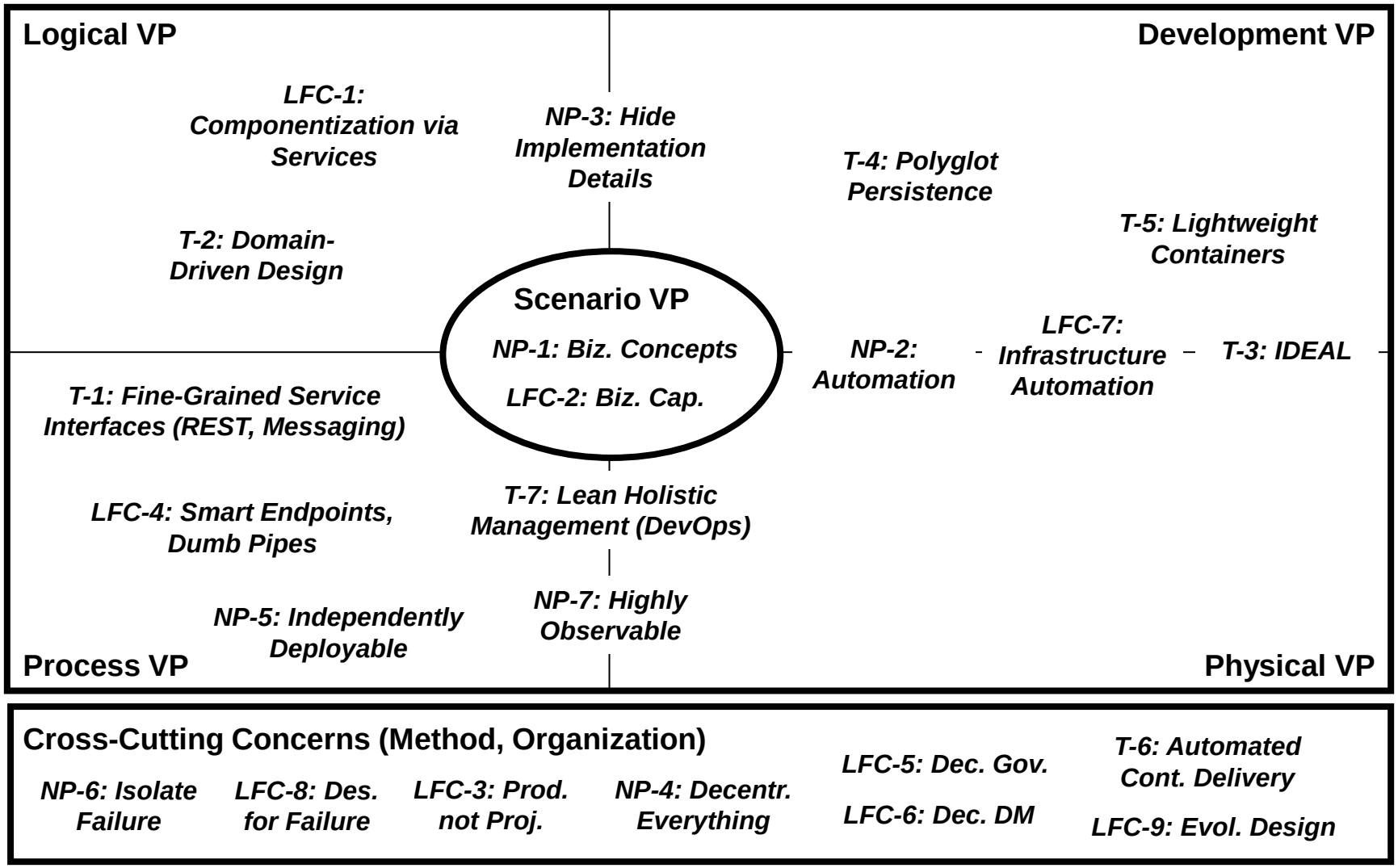
Principle	Viewpoint/Qualities/Benefit	SOA Pendant (Yes/No/Not Applicable)
Model around business concepts	Scenario Viewpoint (VP), intent	Key part of most SOA definitions since 2003
Adopt a culture of automation	Process VP, Physical VP, intent	No direct SOA pendant
Hide internal implementation details	Development VP: flexibility, portability, maintainability	Important architectural principle and development idiom (common sense) irrespective of style (but promoted by most styles)
Decentralize all the things	n/a (not technical but process-related)	SOA governance, might be more centralized, but does not have to
Independently deployable	Deployment/ Physical VP: speed, scalability	No direct pendant in style, but precursor attempts such as Service Component Architecture (SCA) , an OASIS specification with vendor and open source implementations
Isolate failure	All VPs, intent: robustness	Done in any distributed computing approach (hopefully)
Highly observable	Process VP/Physical VP: manageability, maintainability	Done in any distributed computing approach (hopefully)

Tenets/Principles/Characteristics by Viewpoint (4+1)



Reference: Kruchten, Philippe (1995, November).
Architectural Blueprints — The "4+1" View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.

Tenets/Principles/Characteristics by Viewpoint (4+1)



The Seven Tenets for Microservices Implementations of SOA

1. ***Fine-grained interfaces*** **to single-responsibility units that encapsulate data and processing logic are exposed remotely, typically via RESTful HTTP resources or asynchronous message queues.**
2. **Business-driven development practices and pattern languages such as *Domain-Driven Design (DDD)* are employed to identify and conceptualize services.**
3. **Cloud-native application design principles are followed, e.g., as summarized in Isolated State, Distribution, Elasticity, Automated Management and Loose Coupling *(IDEAL*).**
4. **Multiple storage paradigms are leveraged (SQL and NoSQL) in a *polyglot persistence* strategy.**
5. ***Lightweight containers*** **are used to deploy services.**
6. ***Decentralized continuous delivery*** **is practiced during service development.**
7. **Lean, but holistic and largely automated approaches to configuration and fault management are employed (a.k.a. *DevOps*).**

Practitioner Questions (Subset of those in SummerSoC Paper)

- **How did you find an adequate/a suited service cut (e.g., how small/fine is small/fine enough)?**
  - How can Domain-Driven Design (DDD) and/or other approaches to application scoping and functional partitioning) be applied to decompose monoliths into services?
    - Guidance required that is more concrete than “define a bounded context for each domain concept to be exposed as service”.
- **How did you overcome “distribution classics” design challenges?**
  - Service lifecycle management, data representation/schema mismatches, error handling, service versioning and evolution
    - e.g., change of interface in terms of syntax and/or semantics), and How to compose microservices into end user client applications?
  - How about application-level intermediaries, i.e., can microservices also be clients of other microservices?
  - If so, how to avoid microservice deployment dependency and dynamic invocation “spaghetti” (e.g., cycles, overly deep invocation chains)?

Summary of Positions

- **(Parts of) microservices community claims that microservices are a new architectural style that overcomes deficiencies of SOA**
  - RESTful HTTP, Polyglot Persistence, Continuous Delivery, Lightweight Containers, DevOps Approach to Configuration and Fault Management
- **SOA proponents argue that microservices are a state-of-the-art implementation approach to SOA ("SOA done right")**
- **A closer look unveils that most of the key tenets (characteristics) of microservices mostly pertain to development process and deployment viewpoint (and not the logical architectural patterns used)**
  - Many SOA patterns and nest practices can be found in the microservices literature too (under different names)
  - "Independently deployable" as key element (but also not new)
- **SOA design challenges still present – but new ones arise**
- **Not suited for each and every project (at early evolution stages)**

Microservices – Literature and Resources

- **“Building Microservices”, S. Newman (O’Reilly 2016)**
  - Sample chapters available online (free of charge)
- **“Microservices” (auf deutsch), E. Wolf, dpunkt 2016**
  - http://dpunkt.de/a2016_downl/Microservices.pdf
- **InfoQ Microservices zone**
  - http://www.infoq.com/microservices

Monolithic architecture
Microservices architecture
API gateway
Client-side discovery
Server-side discovery
Service registry
Self registration
3rd party registration
Multiple service instances per host
Single service instance per host
Service instance per VM
Service instance per Container
Database per Service

- **Microservices pattern languages (emerging):**
  - http://microservices.io/patterns/microservices.html
  - http://blog.arungupta.me/microservice-design-patterns/
  - http://samnewman.io/patterns/
- **SEI SATURN 2015 workshop**
  - https://github.com/michaelkeeling/SATURN2015-Microservices-Workshop