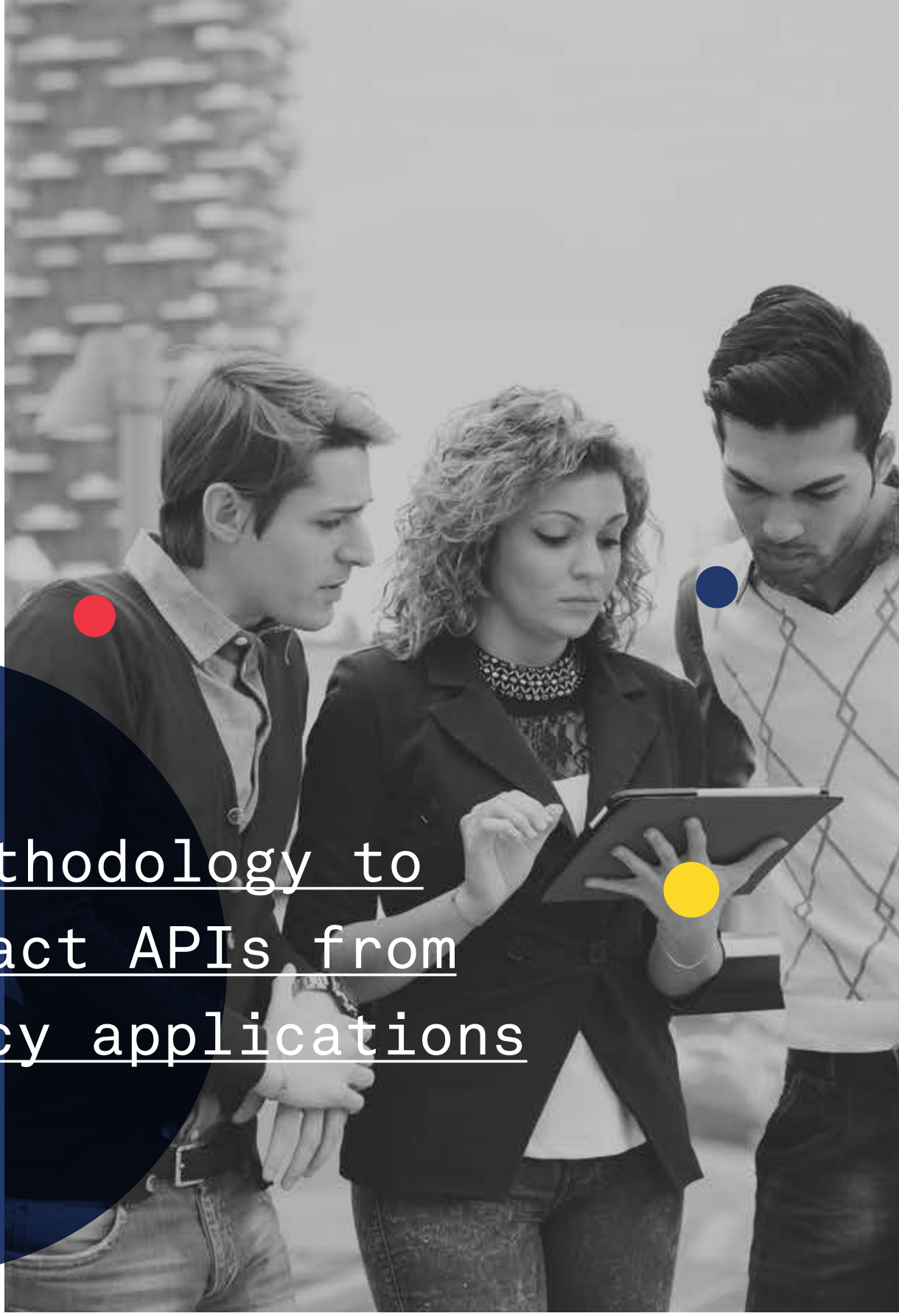

A methodology to extract APIs from legacy applications

wipro

API economy at present revolves around API hosting and management. The pre-requisite for API management, obviously, is to have APIs (Application Programming Interface) on the existing/new systems within the businesses. It is imperative to arrive at a methodology for designing, developing or extracting; and exposing such APIs around existing legacy systems.
Extraction of APIs (API-fication) can be direct: a completely new application or platform leverages the legacy application for supplying to the APIs; and indirect: the legacy application hosts the API within itself.

How to qualify legacy activities

A legacy application is functionality-oriented, it does not necessarily have an API system around it. The most challenging activity to get APIs out of it is qualification of the functions, starting with categorization of activities:

- **Routing:** The logic that routes messages from one subsystem to another based on rules (e.g.: workflows)
- **Decorative:** Attach additional metadata and data elements to the primary message (e.g.: authentication tokens)
- **Consumables:** Marshalling data elements in their entirety (e.g.: product catalog)
- **Helpers:** Protocol translations, format conversions, data migrations, context attachment services and token systems
- **Projectors:** Amplifying and delegating activities such as message multiplications and routing to different orchestrated receivers, hub and spoke activities
- **Validations:** Message validations, business validations against a database, etc
- **Interim IO:** CRUD (create, read, update, delete) operations on datastore

The categorization needs definitions of complexity levels that are useful in downstream activities of estimations, scheduling and implementation of APIs.

How to rank legacy activities

Upon identification of the functions (activities), the next logical step is to rank them using specific parameters. Since the API ecosystem is pivotal for developers, these rankings orient towards developer benefits:

Financial impact of the underlying activity for developers

Financial impact of the underlying activity for the business that hosts the APIs

Impact by creating direct or indirect APIs

Helper impact

Performance and scaling impact

Maintenance impact

Living but near-death status of the activity for developers to consider for future upgrade

Technology reusability impact of the implementation of activity

By design, this ranking mechanism has 'financial viability' as the key consideration, to address 'alternate revenue streams'. The manifestation of financial viability are API-fication scoring models (See figure 1, 2 & 3). The modeller is customizable.

Business		
Category	**Parameter**	**Description**
Revenue forecast	Developers reach	Count of developers across globe accessing APIs
Total Cost of Ownership (TCO)	Current TCO factor	Future TCO of the API system (on top of current TCO)
Revenue forecast	Logic/data factor	Factor of current revenue on to the logic or data (rather than on the services offered)
Business culture	Business complexity	Complexity of the business offered (evaluated on logic and data)
Business scale	Growth forecast	Growth forecast based on data/logic volume against revenue growth

Figure 1: API-fication scoring parameters for business

Developer ecosystem		
Category	**Parameter**	**Description**
Technology fungibility	Cross technology fitment	Ability to offer cross-technology options for exposing APIs
Modularity of functional components	Legacy architecture modularity	Modularity of system/component for exposing APIs
Hosting Ecosystem	Dependency	Dependencies on existing hosting technologies
Modularity of functional components	Compliance	Compliance requirements for logic/data
Pass-through complexity	Chargeback method	Dependency on usage based prices that go to partner/third parties
Documentability	Documentability regulations	In-built intellectual property exposed in API definitions

Figure 2: API-fication scoring parameters for developers

General		
Category	**Parameter**	**Description**
API Enlistment	Number of APIs (Slab)	Number of published APIs
Provider/Gateways	Number of hosting locations/providers/ gateways	Number of hosting locations/ providers/gateways
Metering	Metering conversion	Customizing default metering offered by hosting providers
Monetizing	Monetization scheme	Monetization scheme
Technical complexity	Complexity factor	Technical complexity of the application for APIs
API system type	API system type	Direct or indirect API system
Factor of upgrade	Factor of upgrade	Feasibility for API upgrades
Factor of decommission	Factor of decommission	Feasibility for decommission

Figure 3: General API-fication scoring parameters

The ranking of legacy activities orients towards developer benefits and has financial viability as the key consideration

Peeling the layers of legacy software

Peeling legacy applications based on architecture tiers offer potential methods for APIs.

- Architectural layers and tiers
- Data blocks irrespective of layer or tier
- Reusable code blocks for hosting separately
- Transactions
- Injectable and partial activities
- Based on refactoring and associated qualifications

The API-fication Scoring Model would contain parameters to evaluate these methods for optimum design and efforts.

Decorating APIs

When conceiving APIs, additional parameters of design and extraction (Decorators) qualify the APIs and deal with scalability and ability to document – apart from standards such as Swagger. The main decorators are:

- **Content:** Data and meta-data related to API elements
- **Security:** Identity and security context
- **Instrumentation:** Monitored behavior of APIs and monetization
- **Scale fabrics:** Cloud native and adapted APIs and configurations
- **Gateways:** Access pathways for published APIs
- **Documentation publishing:** In-built documentation protocols
- **Lifecycle management:** Managing genesis, creation, upgrade and decommission

Decorators enhance API consumption, enterprise management and usage, thereby improving financial viability.

Indirect API-fication

Indirect API extraction depends on potential endpoints in the legacy system and considers:

- **Scale:** The endpoints are within the legacy system, need to ascertain the scaling capability
- **Protocol and message formats:** Like other integration mechanisms, the protocol and message format impacts the API feasibility
- **Data locks and volume:** Parallel or asynchronous API calls should consider data locks and volume-related aspects
- **Security:** The security contexts of the legacy system and the API system should be seamlessly interoperable
- **Performance:** The existing performance levels of legacy application should get compared to the required level in the new API system
- **Legacy dependencies:** There would be additional dependencies on the legacy system such as database clusters, application delivery routing (e.g.- Citrix), domains and user profiles, identities, COTS products and proprietary formats/licensing
- **Synchronizing existing methods (process isolations):** Depending on the architecture of the legacy application there may be synchronizing challenges – especially on transaction systems where processes are interdependent and the right mix of isolations need to be evaluated
- **Fault isolation:** In Indirect APIs, the faults and their context, may require marshalling across legacy and API systems

The methodology

A methodology brings consistency in considering the actions, associated templates, algorithms and tools in performing the API roll-out. Figure 4 represents the methodology steps and details.

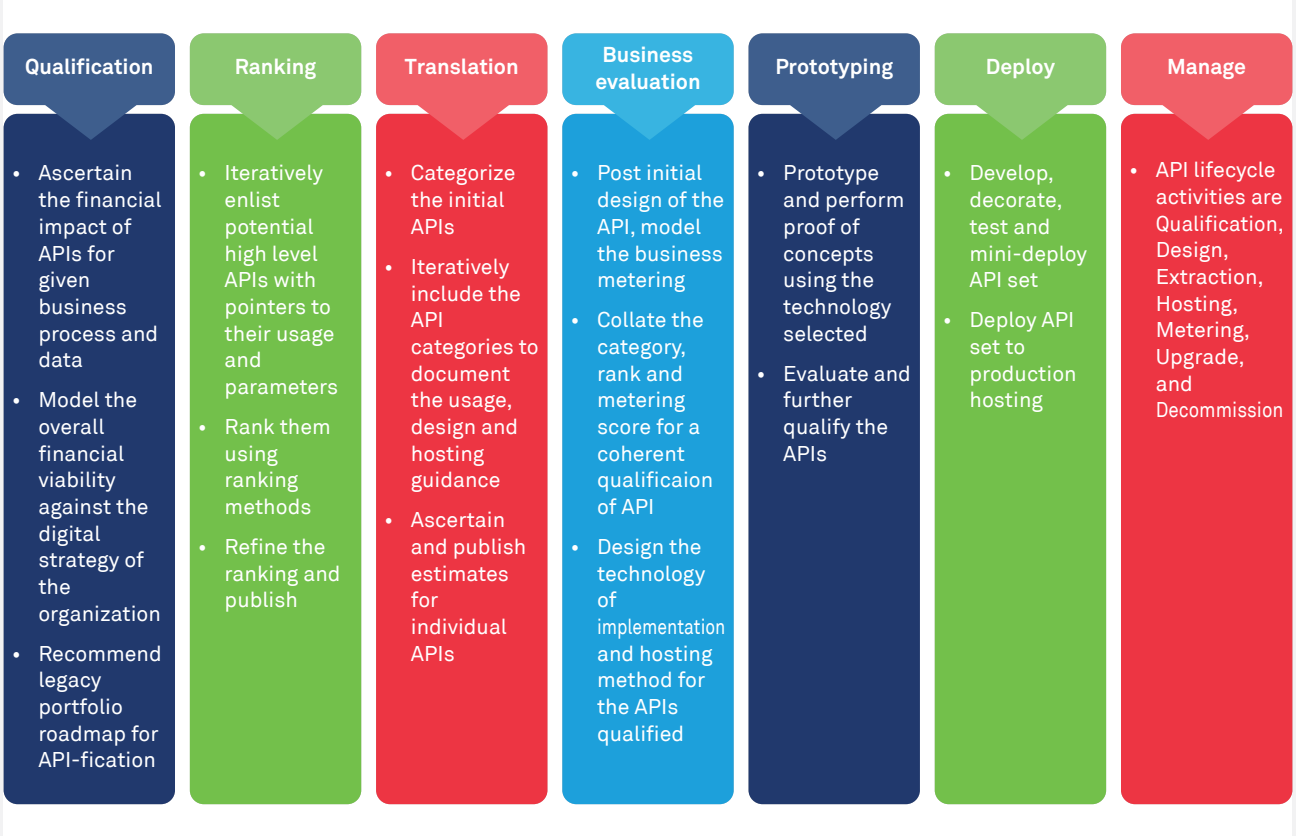

Figure 4: Methodology for API extraction

Towards transformation

A legacy application, with its dependencies and complexities, needs different evaluations to extract APIs out of it. There are direct or indirect ways of creating APIs and it is important to qualify (that includes rankings, financial viabilities and design qualifications) and a methodology to manage lifecycle activities of APIs. These aspects have to be optimally leveraged to transform a legacy application to the world of APIs.

About the authors

Praveen Kodikkambrath

Practice Director & Principal Architect - Microsoft Practice, Business Application Services, Wipro

Praveen has 19+ years of experience in development, consulting, delivery, sales and pre-sales, CxO and analyst interactions, innovation and marketing.

His area of expertise is Application Estate Transformation that includes Cloud and digital initiatives. He has built transformation services to scale, including management of services and strategy, customer accounts, solutions development, delivery and management of associated people structures.

He has pioneered a productized solution that addresses all transformation requirements across Microsoft technologies (Azure, Office365, BizTalk, Dynamic).

Wipro Limited
Doddakannelli, Sarjapur Road,
Bangalore-560 035,
India

Tel: +91 (80) 2844 0011
Fax: +91 (80) 2844 0256
wipro.com

Wipro Limited (NYSE: WIT, BSE: 507685, NSE: WIPRO) is a leading global information technology, consulting and business process services company. We harness the power of cognitive computing, hyper-automation, robotics, cloud, analytics and emerging technologies to help our clients adapt to the digital world and make them successful. A company recognized globally for its comprehensive portfolio of services, strong commitment to sustainability and good corporate citizenship, we have over 160,000 dedicated employees serving clients across six continents. Together, we discover ideas and connect the dots to build a better and a bold new future.

For more information, please write to us at **info@wipro.com**

IND/BRD/OCT2017-SEP2018