Product Development Challenges and Approach for IT Services Companies

Srikanth Inaganti

Context

IT services companies are delivering similar or same solutions in different vertical industry segments again and again at no cost and with no schedule differentiation over a period of time!! The hyper competition in the market place, coupled with reduced capability or skill set differentiation between the services companies, is not only making sustainable growth a challenge for services companies but also making it difficult for customers to choose the right services partner. Many services companies have initiated programs under different themes such as innovation, reuse, etc., that lead either to partial or no success. Given this scenario, this paper discusses the obstacles within a services company for creating a decisive differentiation that would help augment further growth.

Major Challenges

For the last few decades, big services companies have themselves either developed small reusable components and huge products, perhaps accidentally, or they have inherited products via acquisitions. Some of these products are highly successful, and some of them are kept in cold storage. Revenue growth from these successful products from the past is fast declining for various reasons, such as the architecture being old (client-server based, for example), not having support access over internet, being completely out-of-place for cloud model, etc. Some of the recent initiatives for presenting these services as products are met with partial success. Here is the list of challenges for product development within services companies in both building new products as well as in modernizing the existing portfolio of products.

(i) Most of the initiatives targeting differentiation focus on developing the technology components that cut across different domains. In spite of high reuse potential, these will not significantly reduce the cost and effort required to build a strong differentiation factor when it comes to the end-to-end business domain in question.

(ii) Lack of a long term (more than 3 years) strategy or roadmap for the product compounded by frequent (say within 3 years) changes in LOB leadership or practice owners is another challenge, as is not involving domain and architecture leads in direct discussions on strategy at senior management level. Even if a strategy is in place, treating execution as separate [1] leads to un-optimized results or a failure.

(iii) Lack of sustained support for research and development in domain and technology with a focus on building the product – over a long period – is another problem.

(iv) Lack of required upfront funding for development, thus linking the funding to revenues, leads to delays in getting the complete product to the market.

(v) Too much focus on tactical or short sighted goals – e.g., the next one or two quarter revenue targets – would force the LOB leaders to give low priority to initiatives that have high payback periods, which works against the product development culture. Too much rigor in cost optimization may lead to thinking negatively about transformational or product initiatives.

(vi) Lack of differentiated sales/pre-sales/licensing strategies for selling the product as it is and the product as a service.

(vii) Lack of either required support structures or synergy between different groups within the product development lifecycle. Lack of a single and coherent picture about the product initiative for legal, architecture, LOB, corporate functions.

(viii) To build or modernize products keeping in mind a few customers in order in the future to rework the product to make it generic may result in a delay in coming out with the required quality and coverage, ultimately leading to losing the opportunity to be a leader in the market place.

(ix) Too many parallel activities to compress the schedule may force the architects and designers to spend much less time on the problem at hand when actually it requires

right thinking upfront coupled with typical service industry productivity norms such as KLOC per day/week, resource utilization, cost vs. revenue, etc. that would successfully drive towards major reworks later, if not failures. Lack of proper controls for architects to make the architecture planning and execution right.

The following few sections try to discuss the activities and different groups that need to get involved in product development, and critical success factors, funding, revenue models, organization structure, etc., required to address some of these challenges.

Activities before Product Development Kickoff

In order to create a differentiation that can generate quantum benefits, here are the enterprise wide activities to be carried out. Typically, services companies are organized into verticals or LOBs, according to industry segments, servicing customers. Once the enterprise leadership decides to make a product out of the repeatable services across various industry segments, the following are the activities that it has to carry out.

1. Enterprise should be willing to invest and take calculated risks. This might be simple to state but is the biggest cultural change for companies which are either conservative or not used to making investments unless there is an opportunity at hand. Please refer to funding and revenue models section for more details.
2. Enterprise wide opportunity identification and validation:
    a. This requires enterprise architects and domain leads to be assigned to each vertical segment with the task of identifying the projects that are delivered to customers that can be modeled as products.
    b. Understand the type of solutions (architecture and design) delivered.
    c. Qualify the identified opportunities based on
        i. Repeatability
        ii. Expected revenue
        iii. Current opportunity pipeline
        iv. Verify contracts to establish IP, Copyright ownership.
        In case earlier contracts establish the IP, copyright ownership, beyond doubt then take the existing delivery and productize it. In case earlier contracts don't establish IP, copyright ownership – develop a new product altogether.
        v. Tentative break-even time, in case of a fresh build or modernization effort
3. Create a high level plan and tentative investment required.
4. Define the roadmap (Architecture vision, Milestones/Product releases).
5. In case of a decision to use open source software, evaluate its implications for product deployment or distribution strategy. If required, seek legal opinion and approval.
6. Register trademarks for the products being built.
7. Requirements capture and analysis
8. Develop the product.

As part of the development during the architecture elaboration, it is natural to think about using open source software in an effort to reduce the cost of development as well as tp consider the overall product license. Using open source IDEs for solution development should not be a major issue. However, while embedding open source components into runtime, consider the following aspects with respect to open source licensing and its impact on product sales.

(a) Support for open source product in the longer run

(b) IP, Copyrights protection of derivative works. This means understanding the type of license agreements such as GPL, LGPL, Apache, CDDL, etc., required.
(c) Impact of licensing on distribution (of product) vs. hosting

9. Release and Rollout
10. Seek feedback and continuous improvement

The following diagram captures the most important and often neglected activities that need to be carried out before product development kick-off.

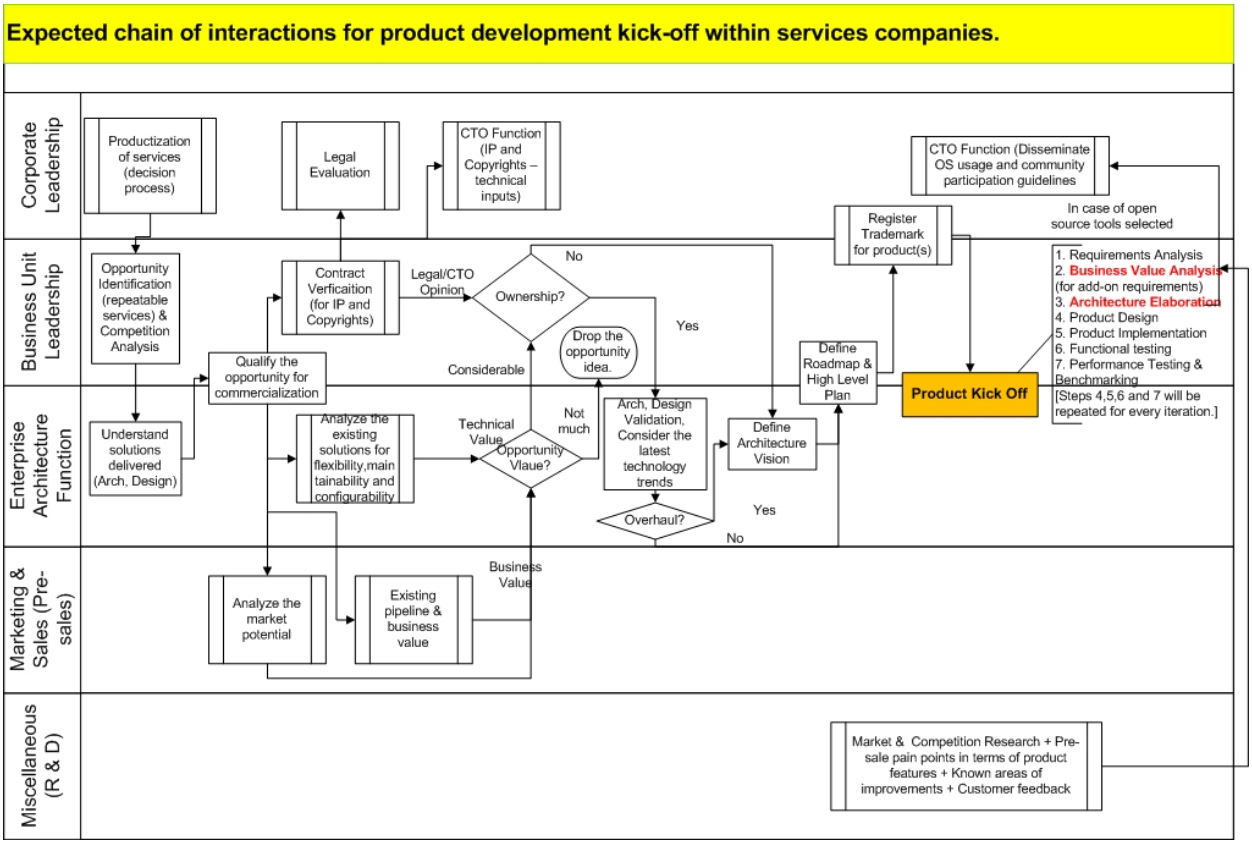

Figure 1. Expected chain of interactions for product development kick-off

Here are the critical success factors to create required differentiation:

(i) Have effective synergy between strategy, operations, and sales/pre-sales, corporate and legal – to effectively mitigate any risks associated with IP, copyrights, and usage of open source, etc. As far as IP and copyrights issues are concerned, it is strongly recommended to prepare a STOP DOING [2] list as guidelines, disseminate them to all stakeholders involved with product development lifecycle and supported with regular audits by CTO and legal function.

(ii) Deliverables from services offered in general become customer's property in most cases – if not all. In order to meet quarterly revenue targets (clinch the deals), services companies will have a tendency to sign-off on complete transfer of IP and copyrights at the end of the project. This will be suicidal if products are to be positioned as solution building blocks at a later point of time. This is the trap that companies running after short-term growth without long-term thinking will fall into. Hence, it is also recommended that all RFP responses that contain the products should be scrutinized by a council/task force responsible for ensuring that services companies will not give away their IP and copyrights, by mistake or without a strong reason approved by corporate leadership.

(iii) Focus on converting or building end-to-end business domain solutions as products not technology specific components or solutions. Leverage Xtreme programming and agile methodologies to incrementally develop the product.

(iv) Define long term strategy and invest accordingly. STOP DOING: Never build a product while keeping just current projects pipeline in mind. Project delivery timelines would force product development team to take some shortcuts that lead to compromise on the quality and domain coverage. Make sure that reworks are minimal in future.

(v) Need to hire seasoned product managers to bring in best practices in product lifecycle management or groom capable service line managers into product managers. These product managers should have balanced knowledge about market insights, business domain, and technology.

(vi) Define right organization structure within each LOB for products. Please refer to suggested organization structure section for more details.

(vii) Define different performance metrics for product business unit. Need to move away from utility and profit center driven performance models at least in the shorter term (perhaps for 1-2 years after the product hits the market) while building the product or until the product matures enough.

(viii) Define incentive or reward model for other support functions to work on product initiatives – to improve motivation and cooperation levels. Please refer to the following section for more details.

(ix) Involve domain and architecture leaders in strategy discussions at senior management level to bring up a partnership model approach. Please note that senior resources or seasoned consultants don't prefer to be order takers and not involving them leads to disorientation.

(x) While building the product, it is critical to bring specialists or solution architects, and technical leads to report into enterprise architect for better control on the product evolution. It is high time that services company leadership should elevate the enterprise architecture function as true advisory but not reporting function on the field.

(xi) Rather than attaching domain leads and enterprise architects to end-to-end project deliveries on a day-to-day basis after the architecture and design phase is over, they should be treated as change agents by separating them into product engineering group so that they get enough time to think forward to improve the product features and qualities further.

(xii) Define the traditional as well as cloud specific revenue models, if the product can be consumed via cloud. Please refer to following section for more details on cloud revenue models.

Funding and Revenue Models

There are two familiar models based on the source of funding. One is corporate funding and the other is LOB funding. Although there are notional differences in the sense that LOB seeks approval for budgeting the product development or modernization, from the corporate as part of the long-term or strategic planning, the impact of the source of funding and the way it is funded on product evolution is quite different. LOB funding helps make the business unit head and his/her team effective partners to incrementally develop the product, and immediately take the product pieces (individual modules or service offerings) to market to get feedback for further improvements in the next cycle of planning. The portions of the incremental revenues accrued from the interim product releases can be utilized to further improve the product features.

Here are the pros and cons of different funding models.

Corporate Funding	LOB Funding

Top-down strategy, tends to be complete/one-time funding and imposes constraint on product development schedule, Typically sought when strong business case exists but the cost of it cannot be borne by LOB, Typically comes when seasoned/excellent product manager or leadership initiates the product	Bottom-up strategy Incremental funding – for the lack of securing funds from corporate, corporate gives enough freedom to LOB leadership to invest and show results in a agreed time frame.
No maneuverability for the lack of incremental approach, Tends to be strategy – execution [1] approach.	More domain focused, Tends to be strategy as a choice cascade, more maneuverability in terms of changing the direction [choice cascade model, 1]
Once the corporate leadership involved, LOB leaders may not feel enough freedom to innovate or experiment.	Sense of partnership model between LOB and corporate leaders, LOB leaders are in control of product evolution decisions
Easy to get deviations or waivers from routine business targets. High attention/focus will be given by all support functions like all technology practices.	Bringing alignment across LOBs for required support is difficult in situations where every business unit has their own business targets. For example, support function loaning the resource at market value rather than at individual CTC would bloat up the cost and hence LOB leaders will be forced into tradeoff between cost and quality of resources especially when it comes to cost of architects or consultants. Also in support function leadership point of view, what is the motivation for them to loan resource at low price?
Relatively short (base) product development lifecycle, perhaps large break-even period. Too many releases for testing within the market, and improvements may not work out as corporate leadership might not prefer to wait for too long – especially in services sector.	Long cycle time to come out with complete product – as incremental development is dependent on profit and loss of LOB. Hence, it is suggested to leverage SaaS and cloud architectures for revenue generation right from the first iteration or phase of development. It is critical to select those modules/service offerings that have a strong business case to go for cloud based deployments. Please refer to section on expected generic product features for more details.
Expect delays in making major decisions especially to adjust the strategy as per the feedback.	Product evolution can be linked to different application deliveries as long as contracts are in favor in terms of IP and copyrights.
Marketing, Sales functions will have to wait for relatively long time to show case	Marketing, sales functions get the incremental finished product builds (modules or individual service offerings) to show case, raise the enthusiasm from potential customers and get timely feedback to readjust the decision made.
Some of the initiatives driven at the corporate level, like innovation, reuse, etc., can be leveraged for further enhancing the product to make up for any shortcomings in the funding. This might involve getting the buy-in from the innovation leadership and assessment of alignment with their objectives.	

Nowadays, hype around cloud architectures [3] opens up new revenue models in addition to typical user based licensing of products. Before that, domain leads and enterprise architects will have to assess how the product fits into the cloud model – that needs answer to the following important questions.

(i) Is the “pay as you go” model appropriate for the type of utility it is [5]?

(ii) Is load fluctuation high enough to go for dynamic provisioning of CPU and storage?
(iii) What type of cloud would be better? Public or private or hybrid?
(iv) Platform choice consideration - Should the cloud platform be chosen right at the beginning OR should we delay that till the deployment time? This depends on whether to leverage all 3 delivery models such as IaaS, PaaS and SaaS or only two such as IaaS and SaaS. If at all PaaS had to be chosen as one of the cloud delivery channels, business analysts and architects had to be extremely careful about the vendor lock-in risk. One of the key decision driver during the platform choice should be the ability to move from one vendor to other without major effort. For example, an application developed for Google App engine can be moved to Cloud Foundry without any changes [6]. Other associated considerations are whether cloud hosting forces intrusive integration via cloud platform API into the product or not? Whether or not chosen cloud platform necessitates tweaking the application or product with platform specific API?

Type of cloud hosting depends on the type of application [5], cost reduction goals in terms of CAPEX and OPEX, kind of customers expected, expectations on economy, security, and control. As a rule of thumb (which may not be true in all the cases), non-critical/support function related, high volume usage applications by individual consumers can be better candidates for public cloud hosting where as applications that are targeted at enterprises such as hospital management system for big hospitals or dispensaries are potential targets of private cloud. Please be noted that distinction between different types of clouds is expected to blur in coming few years time as governance models get standardized, matured to make the customers comfortable.

The following are typical revenue models that can be applied for products that are SaaS compliant.

(i) User Based Licensing + Customization Costs + Annual Maintenance Costs: Typical model where product is restricted by a mechanism that restricts its usage for either a particular maximum number of users or maximum number of concurrent users. Customization costs are charged as per actuals. Standard maintenance costs either on time and material basis or at flat rate. Another variation to this is enterprise licensing which offers unlimited usage in terms of number of users and transactions.

(ii) Private cloud + Transaction Based Charging (TBC): This comes with infrastructure CAPEX for provider and pay as you use model for consumers, be it individuals or small or big enterprises. In this number of transactions executed from each activity or business process can be captured using Billing and Metering module within the product.

(iii) Public cloud + Transaction Based Charging (TBC): This is similar to private cloud plus transaction based funding except that CAPEX is not borne by the actual provider of the product.

The options (ii) and (iii) are like post-paid connections from a mobile service operator or pay utility bills at the end of every month. Especially in the case of health care provider space, few enterprise customers are asking for user based licensing in spite of cloud hosting!! These customers want to take advantage of enterprise licensing as a flat rate rather than paying per use – which is typically high for a 50+ bedded corporate hospital. Cloud hosting with user based licensing is like pre-paid card from a mobile service operator.

(iv) Private cloud based + User Based Licensing + Annual Maintenance Costs: This model restricts the number of users per application as well which indirectly maps more or less onto the number of transactions in a time-bound usage within a day, 9:00 a.m. to 6:00 p.m. But if the regular usage goes beyond a certain number of users, provider would be at loss compared with CLOUD PLUS TBC. Hence, it is essential for the provider to understand the dynamics of usage and finalize on the rates. Please note this is in addition to processing storage charges at the infrastructure and platform level. Annual maintenance costs can be attributed to any additional customization requests, specific customer requirements that can be catered to via service versioning, etc.

(v) Public cloud + User Based License + Annual Maintenance Costs: This is similar to option (iv) above except that there is no CAPEX required for provider for infrastructure.

Suggested Organization Structure

Within each vertical segment or LOB, product business can be treated as sub-organization. Product domain and engineering teams should be measured differently than service industry norms such as utilization, cost vs. revenue, and other factors. Product engineering manager, product architect, domain lead, and product development head should be part of research team. Few of the specialists or technical leads would assist product architect in doing research and proof of concepts, as needed. Some specialists or solution architects and business analysts can get aligned ON and OFF with product customization works on demand.

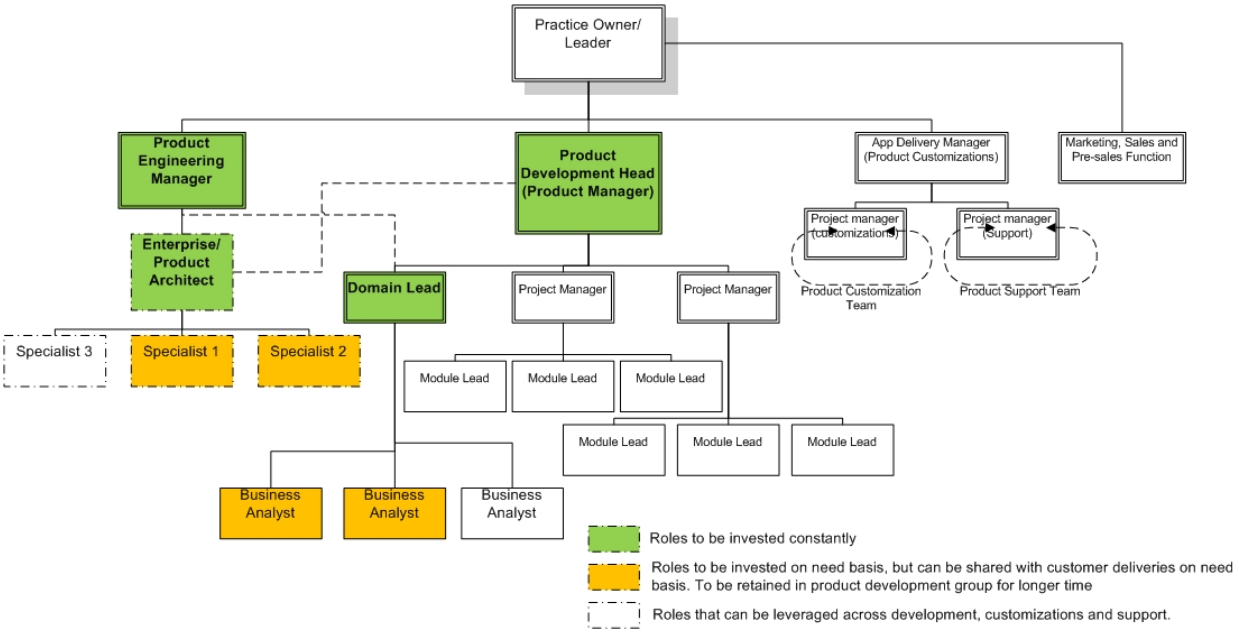

Figure 2. Suggested Organizational Structure

Typical Product Requirements

It is essential to keep a type and region specific segmentation of customers right from the beginning to develop a decent product. For example, in the healthcare provider space, typically there will be slight variations in the data captured for registration of patients between corporate, army, navy, and public/government owned hospitals. Practically, it is not possible to capture all the scenarios right at the beginning. However, making the product architecture comply with SOA or service design principles would make the system amenable for future changes. Hence, product

maturity improves over a period of time as more and more variations are designed into it. Note that product maturity in the world of SOA is closely associated with SOA maturity level [4]. The following diagram shows typical product requirements that architect has to be concerned with apart from non functional requirements.

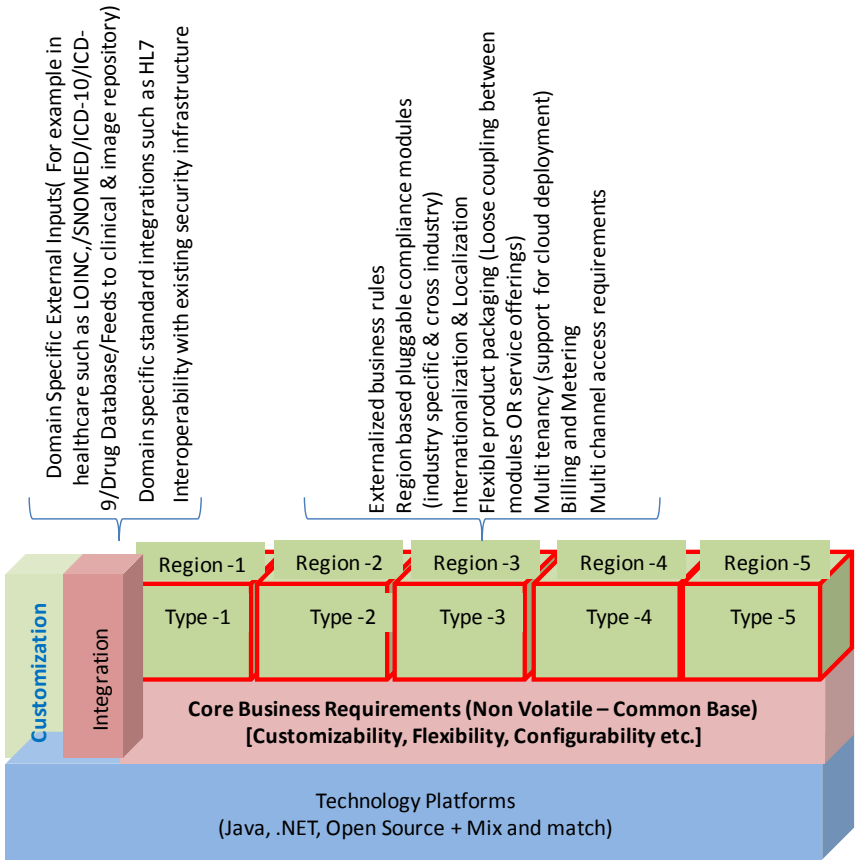

Figure 3. Typical Product Requirements

Conclusions

Typically, service companies are partners to the customer for a specified time frame throughout the delivery plus warranty period. In a lucky scenario, they will be partners for the maintenance period as well. Most of the time the work gets done in a reactive mode. In other words, services companies are used to develop systems for stated requirement and transfer the ownership during the warranty period before pushing it to maintenance phase. Product business is entirely different and services companies are expected to scale up to be true partners to customers over a long period of time, expect to be pro-active and continuously innovate/improve on the product features – via regular product releases and rollouts – for those who sign annual maintenance contracts. This can only be achieved with an appropriate business model defined for the product and performance models defined for different roles within product business unit, through domain and technology thought leadership and guidance supported by continuous funding to do market/domain analysis, technology research in a cost center model approach.

Author

Srikanth Inaganti is a Practice Partner, Enterprise Architecture Consulting Practice at Wipro Consulting Services. Currently he is working as business unit architect for HCIT, Wipro Infotech. He can be reached either at srikanth.inaganti@wipro.com or inaganti.srikanth@gmail.com.

Acknowledgements

I would like to thank Harbirsingh Sawhney, Sudhakar Akkala and Krishna Mohan Avutapalli for discussions that we had around building the product and challenges faced, which lead me to write this article. I would like to thank my colleague Chandrasekhar Ramaraju for sharing his thoughts and providing me with valuable feedback. I would also like to thank Dr Udaya Bhaskar Vemulapati, Senior Practice Partner, Wipro Consulting Services for required support in many ways.

References

1. The Execution Trap – HBR, July-August 2010
2. How do you do "Stop doing?"
3. Cloud Architecture's missing link – ZAPFLASH, Jason Bloomberg – July 12, 2010
4. SOA Maturity Model, by Srikanth Inaganti and Sriram Aravamudan, Published in BP Trends in Dec, 2007
5. How cloud stretches the SOA scope, The Architecture Journal, 21st Volume
6. Java standards help prevent PaaS vendor lock-in

Abbreviations

API – Application Provider Interface

CAPEX – Capital Expenditure

CDDL – Common Development Distributed Licensing

CPU – Central Processing Unit

CTC – Cost to the Company

CTO – Chief Technology Office

GPL—GNU Public Licensing

IDE – Integrated Development Environment

IP – Intellectual Property

LGPL – Limited GNU Public Licensing

LOB – Line of Business

RFP – Request for Proposal

SaaS – Software as a service

SOA – Service Oriented Architecture

TBC – Transaction Based Charging

UBL – User Based Licensing

