Feedback 1

**What we must do:**

1. We must develop an in- house team for Integration as it clearly seems to be the strength of most of our competitors. Planview has a team of 5 people globally for overall integrations, this we came to know by interviewing one candidate from there.

2.We must create *"Expertise Communities"* for various areas of expertise that we have i.e. front-end, Integrations, ECR, etc and each of these communities should meet once a month and share their experiences and thoughts they must also document their experiences so that we don't reinvent the wheel each time and some of their work can also be made a part of the product.

**What we must not do:**

1. We must not micromanage, there are various articles which emphasis on its affects on creativity and innovation and also on how it causes unnecessary stress.

**What I would like us to do:**

1. Have non-meeting hours or non-meeting days in a week (No meeting Wednesday's) it will help people to work on the things that need focused attention and individual time.

2. We must run Pilot projects of 3 months within the company, for newer ideas it could be in our own/new/related space...anyone with ideas can start that needless to say this must be done as self-initiative and out these 2 best ideas can be selected and could get funded by the company to make it commercially viable.

Feedback 2

*Having seen lots of products/demos/Youtube, in your mind, what features/Tech would you like to have in the product going forward which you feel would make the product successful in the space we currently operate?*

1. Capabilities Beyond Project Management Program/Portfolio/Demand/Idea Management to cater to the management needs.

1. Dashboard & Reports   
   This module showcases the ROI of our tool to customer’s management. So, this has to meet business needs 100%, no compromises! Be it simple data extraction, pivot tables, standard metrics or custom metrics with easy export and secure sharing. Reports cater to different needs than Dashboard and hence need to be addressed separately.
2. Guided In-App Tours & How-to Videos   
   Will reduce new user training & overall onboarding time. Textual online help is probably the last option users rely on to get help.

1. Follow Best Practices of Different Methodologies in 1 Product  
   Be it Scrum, SaFe, Scrum of Scrums, Kanban, Waterfall, or even simple Work Management, all options should be available.

1. Easy to Scale & Increase Complexity  
   Scale from a team in a project to a large account spawning multiple departments.

1. Integration with other Tools OOB integration with at least 2 tools in each area and then rest can be availed separately.

1. Use of AI Optimization & Predictions via AI/ML should be built-in and should not require setup by the user.

1. Try-and-Buy Approach Even if the tool requires hand-holding to scale up there should be a way to experience the tool for a small team so that they can get assurance it has all that the organization needs. No/Less involvement of Sales and more involvement from Support to help customers realize the value so that they themselves sell the assess and upsell the tool to their management.

1. Customer Stories & Testimonials  
   Actively seek & publish testimonials to help win quick confidence for prospects and also build champions in the customer organization that will help grow the tool usage.

*What frivolous features/Tech we currently have and is overly configured which may be dropped? Consider the same space*

1. There are many small features that don’t get used often or at all. But none that are overly configured or need to be dropped as they help score important points in RFP evaluation. Probably the amount of ECR configuration needs to be curbed so that it’s easier to manage.
2. On the Tech front, we should NOT rely on the old stack/codebase to get the latest UI/UX and greatest performance.

*In your opinion if you were given a free hand what features/technology you would like to see in the product to make it successful.*

1. Brand Recognition  
   Our product should have a distinct character/be unique. In terms of look-n-feel, application response, product usability, support services, a user should easily remember our product/service after using it once. These are the softer things that are harder to nail but are also the ones that leave a lasting impression in the user's mind.

1. Product Decisions by Data: Best path to improvement is to measure it first!

* We have several capabilities that cover the needs horizontally. We should build vertically on top of the features ONLY based on customer feedback. We should NOT build on top of existing features without knowing existing usage and what are the open issues/concerns/asks.
* Similarly, any new feature must be validated by some of the top customers. Top customers here are not by size or revenue but in terms of the quality of input they can provide and how they plan to use the feature to get value out of them.
* Measure usage of a feature or technical infrastructure that needs continuous investment of time/effort/cost.
* Have a structure in place so that inputs to the feature can be easily assessed across departments. Each feature should be a living entity that has continuous data coming from all ends. So, if someone has raised an issue regarding a feature not only fellow Support members know about it but PM, Development & Documentation should also be aware of it. That way, that input can help improve the quality, usability, and online help for a given feature.

1. Continuous Deprecation While continuously building new capabilities is important to help the users, continuous removal of unused features will reduce the overall maintenance. A clear end-of-life and eventually deprecation strategy should be in place for ALL features.

1. Make it Collaborative Post-Covid, it is all the more important to have any extremely connected workplace. Collaboration should be the underlying framework for all the features. It should be easy to share, be notified, discuss, collaboration via email to keep teams connected.

1. Continuous Feedback Loop

* Quantitative & Qualitative surveys followed by free-flowing discussion on current usage, upcoming initiatives so that the customer feels connected to our organization holistically as an work enabler/growth partner rather than just a SaaS vendor.
* In-App surveys, feedback loops and ratings for new and existing features should be present so that we are able to get a better understanding of the usage and where customer needs help the most

1. Involve Implementation/Support Team/Engg Leads

* Considering these teams are closest to the customers, we should hear their views on what should be the areas they would like us to focus on when we are building a new product or a new feature.
* We should also ask PS Team them to share what went right and what went wrong in the implementations. That way, we know the exact reasons for our success in respective customers. It also helps other teams understand how customers derive value from our tool. The formula may not be some but the ingredients to success of a new product can be!

Feedback 3

For SK

I believe we need to actually simplify rather than make features more complex.

In particular,

1. We have multiple (3) ways of calculating percent complete on a card.  In addition, when we have cards in a hierarchy we have two different ways in which the percent complete rolls up.  I understand that at-least one customer has asked for each of these combinations.  However, given all the combinations it is very difficult to even recognize which combo is being used when a customer reports an issue (most of the time we need to change the option setting).

2.  Over use of card metaphor:

In numerous places we use a set of cards metaphor for showing lists.  In some cases, this representation can become problematic given the amount of space available on the screen.  For example, sprint planning -- here seeing the backlog of hundreds of cards and trying to plan, say, three sprints is a challenge in this paradigm.  Should use lists and tree lists etc when they are more appropriate even if on a casual look that looks boring!

In SwiftEnterprise, we know that Timesheet has way too many combinations and as a result every time we have a new customer that uses a different combination of options we get a new batch of defects.  -- Over 10 years we have not deprecated any feature.

Features I would like:

In SwiftEnterrpise:

The Start module provides rudimentary scheduling -- but does not have a proper scheduling algorithm under it.  Seeing all the competitors, I believe we need to put in the algorithm.

There are two ways of developing this: 1. we build the algorithm in Java and incorporate it in current Start module, or (2) we build a new stand alone microservice that provides scheduling capability, or (3) build a desktop client (node/electron) that provides capability like MSP client and integrate it with the server (or both SK and SE)

I see scheduling as being necessary component of scaled agile as well, when people really start using it.  Also, if we are doing portfolio planning then the underlying scheduling algorithm can again be used to time-phase the projects (or test alternate arrangements).

In both the applications:

I would like to develop a document repository to complement our applications so that we can off-load our attachments (and versioning thereoff) to an external database and communicate with that database through https.  That way attachments will become a link in our database.

This opens up additional possibilities of integrating with google drive, dropbox etc.  and finally reduces the size of our application database significantly.  In some cases like (HCL in old days and Forecia now -- it is 80% or more of the database is attachments!).

Tech Stuff:

1. Update old third party libraries, especially the open-source libraries.  We have some libraries (e.g. image compression in SK) that is so old that it no longer exists and the follow up has been around for over 5 years.  The reason is not only to take advantage of new capabilities, but also at some point because of changes to Java these may stop working -- at that point it becomes very difficult to update them under pressure.

2.  Externalize application cache and web session cache: -- use Infinispan -- This is a reasonably straight forward change to architecture that has large impact on Non functional attributes of our deployment and performance. (Fahad and I had done a POC for this 2 years ago)

1.  Once the session cache is stored in a central place, and loaded each time a request is made, the server becomes "stateless".  This enables us to use load balancing based on CPU load, do fail-over as well as auto-scale the app-servers as load change.

2.  As we now have a shared cache, each cached item is stored only once -- for example, a kanban board will be cached only once -- regardless of the number of app-servers and users are using the board.  Although external cache is slower than in-process cache, the cache can be much bigger and hold significantly more data.  This should result in significantly higher performance.

3. Experiment with Apache Ignite -- this is a layer that sits between SQL databases and app-server that is distributed and thus allowing us to give it lots of memory -- thus making sure that significant parts of the database is in memory when we use it and to reduce the disk IOPS.

Again, if successful, will improve the response time.

Finally,

4.  I would like to explore (I have not done this yet) framework to build plugins that can be configured using OSGI (standard) manifest on the server

and similar widgets for the UI.  This not only allows us (or our customers) to develop new components, but also provides a standard way in which we can integrate microservices in our existing UI.

Final thought:  Having gone through the presentations, I feel that we have lots of features and so on in the PPM area.  We need to add a few more things to round up the story.  But the key weakness comes from lack of a coherent position or story for the solution and our ability to monetize it.

Also, we need to pay more attention to NFR's, such as, response time, usability (that is, logical layout of menu's and features), scalability and Ops (managing the cloud deployment proactively).

Feedback 4

Should:

1. TIGHTLY and SEAMLESSLY Integrate SwiftKanban and SwiftEnterprise -Items from SK can be opened in a SK window from within SEnt and vice versa
2. SwiftEAse to be converted as a SK plugin
3. Enhance SwiftKanban for broader agile adoption - Consider Board as just one of the views -Add list view, timeline view, List view with sub lists of user chosen categories and advanced filtering, Network view (using AI to reflect similarity) and others
4. Modern PPM with AI for Swift Enterprise that appeals to both US & India
5. Resource Management for both US and India
6. Spend resources creating good pre-configured library of templates (we can call it industry Apps) with good documentation - ex: hiring, New Drug Discovery, manufacturing, hospital administration etc -this will give high vertical value and can be done by hiring temporary domain expertise
7. Integrate seamlessly into AzureDevops

Should not:

1. Spend resources and time merging SK and Enterprise into a single monolithic  product
2. Create a stand alone integration product

Like

1. Create atleast 3 teams (juniors with a mentor on demand) - 2 or 3 person motivated teams to pursue ideas like Excel Apps that potentially can give us products or related ideas and expertise
2. https://t.yesware.com/t/41758ffdf06ff8a8ac4d237902251eb719431ce3/2bdb77edc6b4d3f0d17dccd4459d1816/spacer.gifThe item view (item 3 in should) component with multiple view of work items should have a robust API and sold as a "Universal Item View" component that can be used by anyone for  their UI development - will be very valuable
3. Power APPs based new product only after we identify clearly - market value niche, why it will be bought in a crowded market, differentiators, Try & buy strategy etc.

Feedback 5

Instead of like to have features, I will say like to have functionality -

1. PPM/ NPD/ Innovation Management/ Idea Management with AI "Tadka". In my mind they are all have similar base set of functionality that can be finetuned for different purposes - including Finance, Demand, Prioritizing, What-if, etc
2. I think we should take the preset template building as a Prod Management activity instead of leaving it to PS when they are free - this in my mind will allow us to show differentiator and also have different flavors of solutions.
3. Ability to flip from WBS/ Gantt to timeline to board views to Calendar views seamlessly will be (eg Asana)
4. OKRs/ Goals tracking
5. In context collaboration
6. Do some work on making our test management module useful and robust
7. DevOps linkages to GIT/ Jenkins/ Docker should be OOB
8. Power BI type integration for DIY reports and dashboards, include some better OOB dashboards and reports
9. Embed Rishi and other AI capabilities into the mainstream

Things that can be dropped

1. Can't readily think of anything but I know deep down some redundant capabilities are there and they can be rationalized

I would take the PPM, DevOps, Better reporting to the current capabilities as the MVP for the new product

Feedback 6

As you can imagine and understand, I have many misgivings about building more products with more features with the kind of engineering and UI capabilities we have - and even the depth of domain expertise in product management.  Should we model ourselves after single product companies like Leankit and Trello and even Rally or VersionOne in their pre-acquisition days - for successful and faster acquisitions OR do we try and build a large product portfolio with the risk of not being able to do them all and do them with the level of quality and UI that we need in the market? (Of course, I may be off-tangent here thinking of this as a big-bang approach, while I know you will be more pragmatic of what we can take on and achieve.)

Anyway, I also recognize the need to beef up our current offerings in the best way possible to preserve revenues while positioning successfully for the future.  So, here are my thoughts in the attached PPT of what I believe are the best options for us to move forward.

Feedback 7

Apart from the fact that a PPM tool will require the basic set of capabilities (within itself, offered out of the box) like proper resource planning and demand management capabilities, among other things; and without going into UI/UX (function over form is what a user of such products would prefer?) and Integrations, which I think are table stakes, here are my thoughts -

* SwiftEnterprise -
  + What to remove - We've always heard that customers mostly use only these modules - Defect management, Risk management, Time sheets, task tracking and dashboards. We should maybe remove the rest, or when we're building the new product, leave these as a core set of features/ capabilities and not carry over the rest.
  + Existing modules/ features to improve - I think our STaRT module needs some looking into. If someone wants to quickly get started, there's no need for a behemoth like MS Projects. We should make STaRT more robust, fix it so that we can handle more tasks and make it the scheduler we recommend (we can always integrate MSP)
  + And maybe remove ECR -   
    This might sound unreasonable, but I've heard that we use the ECR framework for anything and everything. Sort of like a crutch. It's a very powerful framework indeed (which makes us twist it in many different ways), and it definitely helps us win orders and build customizations, but if we're going to be SaaS only, we need a more elegant solution - or better still leave it to the market - expose APIs, build some integrations, rest is out of the box. The more ECR we use, the more fragmented our solutions will be.
* SwiftKanban -
  + We should stop building features. Remove nothing for now. Add nothing. Just fix reported defects. The only thing we should try and do (after we've sorted out the performance issues we keep hearing about) is make the SwiftKanban experience more uniform. The save button should work the same on all screens. I should be able to find standard menu icons in standard places...

We need to look at our product's workspaces (and licensing) in terms of personas (clearly not everyone needs to be exposed to all parts of every product). Of course, like a libero in football, we could have a user who could access every part of the tool.

But I think we need to realise that different people in different teams expect different things from a tool and use it differently. Choice might be great, but all things being said, the tool is ultimately just a means of tracking some work. It's not actual work. The tool should get away from a user's mindspace as quickly as possible.

And that's what I wish in terms of what I'd like to see. It could be AI that takes care of it. It could be as simple as reducing the number of steps it takes to perform an action. It could be just simple decision support - for eg, we have the capability to predict lead time on SwiftKanban to some degree of accuracy - it shouldn't be difficult to suggest due dates for a card on creation - irrespective of whether the customer has it configured or has bought a module or not. We have the adjacency matrix, but what would help is a notification to the team after a month that says 'Hey, you're skipping stages/ columns frequently - might as well remove that stage from your workflow'.   
  
...

I remember discussing with Raghu around four years ago that we had begun moving SwiftEnterprise towards SwiftKanban and moving SwiftKanban towards SwiftEnterprise in terms of features that were being added and future scope. It seems like the time has come.   
  
One last point - I was telling Ashwin some time back that we've been much better at building for managers than for developers (and that doesn't change overnight). Anyway, we also might face fewer problems in terms of scalability and performance with a product that's used by a lesser number of users. And within our domain, both point to PPM (which in many ways is more decision-support than tracking). The target set of users are also a more focused lot, who'd expect and give (in terms of feedback) more signal than noise; and in some sense we'd be moving up the chain, at least in terms of money and decisions.

Feedback 8

**Having seen lots of products/demos/Youtube, in your mind, what features/Tech would you like to have in the product going forward which you feel would make the product successful in the space we currently operate?**  
Single platform for Agile methodologies rather than different products SwiftEnterprise, SwiftKanban & EASe will be more successful  
Need clear approach if we are going only for cloud or for on premise or both.  
AI/ML/BOT which will auto fill the data to the end user without thinking too much.  
Integrations with multiple systems will help   
Collaboration with multiple platforms will help   
If we want to sell across the globe we need to have localization support.  
  
**What frivolous features/Tech we currently have and is overly configured which may be dropped? Consider the same space**  
  
JSPs in SE needs to be ported if we plan to increase life of this product   
YUI needs to be ported in SK if we plan to increase life of this product   
  
**In your opinion if you were given a free hand what features/technology you would like to see in the product to make it successful.**  
  
Should not be a Monolith product, as more the Monolith it is more is the cost of maintenance ,training, and understanding usability.  
It should be very easy to use Ex. whatsapp lightweight microservices   
Observe the usage data & accordingly modify/enhance it as per the user inputs  
It should have good integration & collaboration capabilities along with AI/ML/BOT

Feedback 9

Following are the basis of my suggestions after going through various competitions.

* None of the competitions which are purely in the project management domain alone is doing well.
* The ones who are doing exceptionally well are havening a complete ecosystem of tools with No or just a loose coupling and a tight integration with each other.
* Then others are doing reasonably well are spending significantly on marketing.
* But everyone in the ecosystem has benefited from the right amount of integrations and various marketplace presence.
* The project management verticle, well, in fact, any verticle for that matter, is continuously evolving, so one strategy getting executed for 12 to 18 months may not be enough.
* We don’t have deep expertise in every vertical; hence a platform would help us more in experimenting in various verticals by engaging with domain experts, doing a short-lived POCs, Rapid prototypes and pushing it to cloud-first for immediate feedback.

With the above point in mind, what we need is these (Red ones are **What we need to have**, Orange once are if **we were given a free hand**):

* **Extensibility:** Instead of having one focused solution, we need to invest in building a platform capable of introducing Standardised extensibility in various layers.
  + Open API - (not just for customers, our frontend should consume the same APIs)
  + OAuth with Open ID make authentication secure at the same time flexible and easy to operate.
  + Micro frontends - (not just for our internal consumption, our customer should be able to use it to extend)
  + SDKs and CLI for easy integration with DevOps tooling.
* **Integrations / Marketplace listing:**
  + Presence on Azure and Amazon marketplaces.
  + Double our integrations with the collaboration tools, First with Slack / MS Teams / Google Chat as the standard ones, and then some other upcoming like Miro.
  + Microsoft power Automate, Power Bi and logic app connector
  + IFTTT type of Integration platforms like Zappier and Microsoft Flow
  + Integrations with VUI (Alexa, Cortana, etc.)
  + Azure DevOps, Git, Developer plugins (Jenkins / k8s) for enhancing our DevOps story
* **SaaS First approach:** Cloud-first is the most important differentiator in Elite Organisation vs rest of us.
  + We need to have SaaS first approach, and this will help us build and experiment with features rapidly.
  + Build SaaS only feature first and extend them to on-prem later on depending on demand. This will help us reduce time to market drastically.
  + Transparent billing or pay as you use will go a long way. Reducing TCO for the customer. But then it will be a volume business for us. For Example
    - invoice based on # of the project or # of items
    - Size of data and Access frequencies.
* **Observability:**
  + The first area is to understand what's going on under the hood from application performance and security perspective will give us speed and stability.
  + The second area is to understand the usage pattern and making informed decisions on what features we should build or enhance. This will help us make data-driven decisions. Not rely less on gut-feel.
  + We need to understand the usage patterns, what features are most used vs least used, how much time it takes to accomplish a specific task, etc.
  + Based on the usage patterns, make the changes in UX and functionality, like showing relevant information at the right place instead of 4 clicks away. AI for UI is making a lot of progress in this area.
* **Localisation:**
  + We have to do tie-ups with external sources for making our application multi-lingual and have a tie-up with the locals for the partnership for sales.
* **AI/ML / MLP:**
  + What we need is an excellent Conversational UI with Domain intelligence. Like Personal Assistant, which Param proposed some time back. I believe it is the main differentiator in this or any space. This will require significant efforts and skillsets from NLP expertise.
  + Smart speaker - It will only make sense if we have a strong personal assistance capability. Voice extension is a simple problem to crack.

**What we need to stop doing:**

* Stop being obsessed with one methodology, instead build what works for the user based on observation of usage pattern.
* We do have mix panel, but no one looks at it, not sure if we ever build any feature on to reduce the pain of user by looking at data.
* Longer implementation cycles.
* We should not rely only on Employees, Instead of increasing headcount, explore more in the space of Crowdsource like Upwork. Primarily because of its instant availability of skills and the cut-throat competition. Very useful for building or updating smaller integrations and satellite features.

Feedback 10

My opinion is that we should spread out to other domains rather than ho deeper into IT services, software domains. Hence, ALM/devops etc does not become important for me.

I also am a proponent we build a good platform and over that build vertical solutions rather than building hardcoded products that work in one way. This is because I feel we don't understand any domain very well (including IT) . So we have to work with people who know domain and jointly put together  products/solutions rapidly. That is only possible with a platform.

**Things we should have if we have to do well and expand in our space:**

1)  Hybrid project portfolio management spanning from strategy to execution.

2) good scheduling engine - much better and faster than STaRT. If u want to do serious ppm with whatifs etc and spread out to other domains.

3) modern ways of resource, demand and finance management. Sensibly AI enabled use cases for predictive, assistive, automation.

4) our own self-service reporting engine. ECR++. Using this we should develop many OOB reports/dashboards also.

5) new rules, forms configuration module that is much easy for business users. ECR++.

6) Robust oob integration with SAP, Oracle, salesforce, jira, PowerBI, document management systems, Mural, icertis.

7) only SaaS/common cloud. No on-prem.

**Things we should drop**:

1) old eform framework - replace with new

2) old ecr renderer - replace with new

3) old Start - replace with new

4) features that are not used much. Using 80-20 rule.

**Things I would aspire to have**:

1) conversational bot that is truly intelligent. Had circulated a sample use case years back.

2) building industry specific templates for ppm.

3) excellent mobile interface for all the key features.

Feedback 11

**Features/Tech To be there in the product**1. UI Level Integrations with important Tools which are always part of ecosystem  
(e.g. JIRA, Azure DevOps (TFS) , cloud Document Management systems, microsoft teams , slack , GIT and many more)  
  
2. Extensibility of the Product w.r.t. Developer plugins so that the product can be customized and extended based on   
the business needs by the customer.  
  
3. UI Driven Rules Automation within product as well as across Integrated systems to achieve seamless workflows  
between integrated systems.  
  
4. It should be a Single Product offering,  Configurable and Customizable Execution PPM Platform to support  
TRADITIONAL PROJECTS AS WELL AS DIGITAL PRODUCT PARADIGM  
(Resource , Demand , Financials w.r.t. Portfolio , Program and respective Projects/Initiatives/Application/Product)   
as well as   
Configurable and customizable Execution Layer (Waterfall, Scrum , Kanban , SaFe and Other Methods)  
  
5. Both Execution PPM and Actual Execution Layers should be designed separately and then should be well integrated  
OOB. This way customers can use Execution PPM and Actual Execution Layer from OOB  
OR  
Customer Can use Execution PPM from our product and can integrate Actual Execution layer which is not part of our product  
OR  
Customers can use Execution PPM from other tools and can integrate the Actual Execution layer which is modelled in our product.  
  
6. Make current features more collaborative (e.g. RTF, emoji in comments)  
  
7. Make Product easily pluggable from other systems  
  
8. Product should be able to open widgets of other systems seamlessly.  
  
9. New UI/UX to be focussed more on putting placeholders for showing relevant information at the right place.  
  
10. Better and faster way of making work hierarchy breakdown (as we have seen finally it is all about work and it's eventual breakdown to granular level and  
reporting the status at each and every intermediate levels).  
  
11. Make features of Timeline , Portfolio Project Comparision/What-if, Better Resource Allocation/Loading Visualization, Budget and Actual Data acpture and  
visualization.  
  
12. Important Navigation , User Actions and Insights for a user to be supported via ChatBot.  
  
13. AI Use case for Execution PPM to suggest which projects to be considered in portfolio based on the other parameters given for each project in terms of  
Business Need , Resources required , Resource cost , Business ROI, Project Timeline , Time to Market Impact etc.  
  
14. Integration with Reporting frameworks like Tableau and Power BI  
  
  
  
**What Can be Dropped or not Required**1. Bulky Timesheet Module.  
2. All the Old Reports which are part of SwiftALM/SwiftKanban ... they need to be revamped to better visualization  
3. Get Eform/ECR out of JSP so that it becomes more pluggable and integrable.  
4. Workflow feature of ALM needs to be simplified functionally as well as visually.  
5. Rethink ECR strategy. Larger configurability and customizability is better managed by either end user Citizen UI or   
Pure pluggable developer frameworks to achieve ripple effect in the market.  
6. Strategic PPM (Mission , Goals, Initiative Planning and respective what-ifs)  
seems a total separate layer and can wait till we come out with all other aspects of new product  
  
  
  
**Given a free hand ....  features/technology to be developed**1. Make product more pluggable and integrable  (Developer can extend it by writing required UI/Server side codes in pluggin  )  
2. UI Level Integrations for important tools generally present in customer ecosystem  
(e.g. JIRA, Azure DevOps (TFS) , cloud Document Management systems, microsoft teams , slack , GIT and many more)  
3. Make Work Hierarchy creation simpler.  
4. Create Execution PPM Layer which is Integrable to current work execution layer  
5. Put more features for collaboration of work.  
6. Rules Automation within the project as well as across the system.  
7. New UI with important information in context.

Feedback 12

1. What should we do?

A) Revamp our Marketing. We need 2-3 people from successful B2B SAAS companies. It is my absolute conviction that without this changing, no matter what we build or invest, we are burning money. Do not get me wrong! It is not that Mahesh is not doing something that I would have done. In SK, I think we have done whatever I could have also thought off - engage with the community, do webinars, be in the mindshare of the thought leader, etc. We are in the radar of the community. Yet, others in the Kanban community have made 2-3 times more revenue than ourselves. So, the market is small logic does not convince me.

Plus, our Marketing team, except Mahesh, is not a Marketing personality. I am sure even Mahesh knows this.

I do believe that none of us understand this well.

I only partly agree with your thought process that the product should speak for itself. All products would have some things they do well and some they don't. That can't be the reason for our 4/20 year hit ratio of very modest revenue goals. While I know that our Kanban product has limitations, once someone understands its navigation, it is the most value accretive product out there.

Comparison of revenue per employee cost is misleading because of complete lack of revenue growth; not because Engg is sitting idle.

B) We should also rebuild our Sales team. Even if we believe that our SE Sales is Enterprise, our Sales relationships are very peripheral. When I was in the Sales business, we were in the relationship business, even in sub-$1M accounts. We would dine with the CIO once every 6 months, take them to India once every 3 years. This was standard protocol. Mahesh, Ram, Raghu... have all done this for years in IT services. Yet, even our SE relationships are shallow and always prone to getting killed at the slightest disturbance.

Forget CTS or Wipro CEO now who are "firangs", why haven't we been able to meet and connect with the CEO of Zensar or Hexaware, who are our generation. Why can we not simply go and meet DK at Birlasoft who was our colleague at HCLT.

c) Build a product and design team. There is no product of any scale and size without this.

d) Only 4th in my list:

Two different markets: software development + outside software development

For software development: my theme for the product is: for the development team and by the development team. So, what makes them better. Here is my thought on this:

--- Core area of improvement: anything that make development teams do development better:

------ Collaboration: Integrate with the MIRO/Murals of the world, asap, when its feasible

------ Voice Integration

------ OKRs and Retrospectives

------ I have some other ideas like social interface for Development teams, that is engaging and collaborative, with gamification.

------ Build capabilites to support a POD like team structure where PODs are staffed (not projects are staffed)

--- Satellite area of functionality: PPM, Capacity and Resource Planning

------ PPM: We need a checklist level PPM functionality but I am clear in my mind - today, decisions are made by development teams; not by the CIO office. I don't think anyone will buy SE because we have a great PPM functionality.

------ Capacity and Resource Planning: We know enough to do this ourselves in 2-3 quarters. No big deal, at all. This is important, more important than PPM, in my opinion. However, I am calling this as satellite because this is not for the development teams and

For outside software development: exactly opposite priority. These are top down companies with top down management style even now.

--- Core area of improvement: PPM, Capacity and Resource Planning

------ However, we need some consultant from 1-2 non-software industries to guide us through this.

There are 2 different markets, 2 different product philosophies and need to be treated differently. Mixing up the two will make both spaghetti.

2. What we should not do

A) Focus on all teams other than Engg. We are obsessed with Engg just because they are more than half of the employee base.

Well, they shouldn't be half of the employee base. Engineering is last in the supply chain. Engineering take 2 weeks to build a Story. It will take 2-4 Sprints to turnaround an EPIC. To ideate, imagine, design and write its Stores, takes 2-4 months! Upstream takes 3 times more time. Today, we have slowed down our own productivity by not focussing on where the constraint is in our own supply chain.

Every Technical Lead cannot become a BA. They don't want to do it. They don't enjoy doing it. Of all the Leads we have, only Rahul enjoys it. 5 others don't. They want to grow in the technical track.

AVS, we all use libraries today. We built 9 widgets for Agile Program Management in 1 month with a new team on new stack. We rolled it out with another Sprint of Integration. I can assure you it is faster development than Kairon with far more broader functionality and testing complexity. There is no comparison. I don't like to compare but when you keep comparing as if they are doing something that the product lines aren't, that is incorrect. I have repeatedly asked you to join the Sprints for that reason so that you can see and experience it yourself.

I am reading everyday, every moment of how to make software development better. I can assure you I am reading and attending conferences 3-4 times more than anyone else in Digite.

B) Cannot deprecate any feature because for every feature, there is a Leadership stakeholder who has suggested and got it done. It cannot be deprecated. There are many examples of this. There are 4-5 different help options in SK. No one uses them. The only Help option people use is the BOT today and it is used liberally. Yet, I cannot deprecate them or remove them. They cannot be maintained; they are getting obsolete.

Plus, one user will come in the ecosystem and say, we want this. So, we have to build it for them, even if it were a new component or a new product.

3. What I would like to do:

A) Don't run after a product that can grow the company. Run after an idea that we feel we can make the ecosystem better. I know this is very philosphical but IMHO, if we have to succeed like a product company, we have to run after an idea. We are not PE tot turnaround and sell. That is not our mindset and I believe we cannot today adopt that mindset.

Feedback 12

1. We must make our system Scalable and Performant.

2. Personal Assitant available on Smartphone/speaker to carry out mundane tasks.

3. Cloud-First development approach.

4. AI for UI to find the usage pattern and show the relevant screen to the users.

5. Smart Speaker integration for real-time navigation on UI.

6. Making ECR open source, it is a closed source makes it less lucrative for other Services Professionals.

7. Data Governance Module to validate the quality of data as we have faced a lot of issues regarding data quality.

8. Self Explanatory Analytics.

9. Devops Monitoring and control via smart speaker / phone.

10. Integrating Google Meet / chimes data and extracting minutes of meeting.

11. Sentiment Analysis on Customer Queries raised which will help to find how haapy is the customer.

12. Feature Usage Dashboard to help the Organization in license procurement.

Feedback 13

1. After going through all the competition , we have **still not been able to figure out what differentiates companies making good revenue vs not making good revenue** in our space. They all seem to be doing well enough ( except a few )   . The only thing that most of us keep coming to is

                                                          a) The ever elusive slick UI

                                                          b) Some feature which we don't have  ( Itegrations / PPM / Usability  etc . )

                                                          c) The fact that these products dont love AGILE /Kanban/waterfall ( i.e. they can be used for anything )

                                                          d) Should go to other domains than IT / International Support

 My suggestions for what to do and what not to do stem from these conclusions

                                                         a) I think team is anyway concentrating on this - If this still fails then something is really wrong with current initiatives

                                                         b)  No number of  features will be enough because we really cant say companies are making money because of this feature -

**Find a feature set that is common among top 10 compeitiors** - see that we have that and then drop the rest

                                                         c)  We seem to be stuck up on the love for a certain process - But atleast from  implementations

                                                               most are ECR ( flexibility being the key )   or  Kanban ( people using it they way they want to )

                                                              I feel most industry uses the Agile etc. for the jargon and ofcourse we must leverage that - but in name only

**We should create a singluar process creation mechanism that enables all methodologies by leveraging - Time variant markov chains**

 d)   Other Domains / International support -   We must look at tying up with software partners in Mechanical  / Energy / Oil / Travel and Logistctics domains

                                                                in an attempt to sell our software - we can give them large margins just to get some information on these.

**International Support is low hanging fruit**- just outsource UI / Documentations to free lancers who can get inital alternate UI versions out -

**We need international sales in EU APAC region  - we need to push for this**

                                                               Sales team needs to be strengthened IMO having just 2-4 top ppl in sales is very risky in my limited experience

                                                         e) Long term PPM : We continue with the AI initiatives because that is something others dont have at this juncture - but we need to move fast - **having a 1 year                                                                      timeline is too long** - Get initial versions out - fleshing out will follow

Feedback 14

While going through my note, I missed one point. Just want to mention.

1.       Export of Status reports / dashboards to PPT format