Sensible Defaults

PM Australia Home

Structure of this page

The **first section** outlines our preferred approach to running ***software delivery projects.*** This is a high-level overview with key activities and outcomes we expect from teams. If you are on or starting a project which is not following the approach here, please have a chat with Delivery Support or your Market Tech Principal. We would like to understand when and why we're doing things differently.

The **second section** is specific detail for our practices and covers the entire lifecycle of a project. We have split this into 3 main time stages: 1) Starting a new project 2) During Delivery 3) Ending a gig

If you're unsure of what we mean by something in the first section, it should be covered in detail in here.

Our sensible defaults for software delivery

Ways of Working

- We run a project kick off & ensure everyone on the account is familiar with the delivery commitments in the SOW
- We work in Iterations that provide a cadence for our work, support team rituals & communication
- We use the following rituals & tools to support our work; Daily Stand-ups, Retrospectives, Showcases, Iteration planning
- User stories are used to articulate scope & are:
  - Relatively estimated (we don't start by estimating scope with card counting - don't estimate epics - when they are broken in to stories they will expand some times exponentially)
  - Force-ranked in a backlog so the team know the most important items to work on
- Just enough stories are elaborated to ensure no wait time for the devs
- Tech debt is made visible and actively managed - it is treated as a "first class citizen" alongside business stories
- We have a card wall that is always visible & reflects the flow of the work

Tech

- Code is kept in a version control system
- We do Test Driven Development.
- We do Pair Programming and rotate pairs regularly.
- We practice Continuous Integration: have a Continuous Integration server & practice Continuous Delivery with changes being pushed frequently
- Our infrastructure is automated, created as code, and updated through our CI server
- We do Trunk Based Development with Feature Toggles
- Our application is observable in production & the team knows the path to production
- We don't trade off on Quality, we trade off on Fidelity. Quality includes Cross Functional requirements.

Managing Delivery

- We Plan and Track our work
  - We use a burn-up chart to show estimated scope, changes to scope (including the rate of change) & to track our progress towards our goal
  - Velocity and cycle time are used to drive continuous improvement
  - We create an iteration report every iteration for our stakeholders (including TW) covering burn-up and risks, issues, blockers, key events to drive conversation around getting the planned and actual delivery dates for the release (not the iteration)
- We track, visualise and actively manage RAIDs
- We take and share notes from key meetings.

All documents are stored in the Team Google Drive: Community::Project and Program Management > Project Management > project-management-communities > pm-community-australia > PM Toolkit

NB: The sensible defaults are a sub-set of the larger PMOz Toolkit

The deep dive...

1) Starting a new project

Here are some standard team shapes

Team Shapes

WHY: We know that some team shapes work better than others depending on the type of work and the phase of the work

WHAT's THE DEFAULT?

- Unless its absolutely necessary we don't staff solo roles.
- General ratios on a software delivery project team are:
  - 1 PO : 1 BA/UX : 1 PM/IM : 3-6 devs (1 leading the dev): 1 QA
  - #s of roles depend on the problem to be solved and the phase of the project (e.g may have more design at the beginning and bring the QA on board during iteration 0)

Links to:

- Example **Engagement Kick Off** deck created by Sameer Deans Engagement Lead [2018.02]
- A Self Canvas by @Katie Todeschini-Martin and @Fotina Koutropoulos is a great short activity for getting to know your colleagues.
- Hofstede Insights: Compare Countries a simple tool for understanding how people from different cultures may have different norms and expectations from your own.

Engagement Kick Off

WHY: Bring TWs team together before they start on client site to get to know each other, get to know more about the client and get to know how and why we are going to help the client achieve their outcomes.

WHAT's THE DEFAULT?

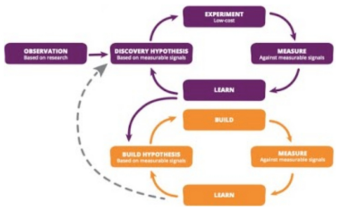
An Engagement kick off meeting is held that covers the following agenda items:

- Purpose of Engagement (what is in the SOW - background from the CP including account strategy if available)
- Outcomes for the first week and end of engagement (as we know it today)
- What happens > day 1 - week 1
- Known Conditions - Blockers on client
- Team shape - Roles and Responsibilities within the team
- Stakeholder map
- Team Building / Personal Learning Goals (and capture any open questions)
- Logistics: location and travel to the client, leave, comms channels, security, start time, finish times, regular meetings times etc.
- Share the team drive and add people to group chats
- Share the onboarding deck (if there is one)
- Share the Sensible Defaults

WHO: Client Principal, Delivery Support, MTP/Head of Engineering/delegate, TWers in the engagement team [Optional: Staffing, Office Principal]. It's usually facilitated by the Delivery Lead and has strong support and presence from the CP to provide context.

HOW LONG: 2 hours - It's a short sharp session to ground everyone in the work, prior to starting on client site.

IT'S NOT: a replacement for an inception or for reading the Statement of Work



Content Pending

Dual Track Product Definition and Delivery

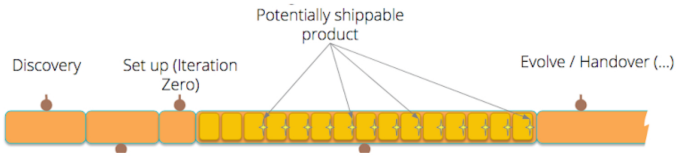
Variations for product innovation, Intelligent empowerment and Responsive org

Project Lifecycle

WHY: ThoughtWorkers and client know roughly the phases that a project will move through

WHAT's THE DEFAULT?

- There is a shared understanding of the project lifecycle (team and client)
- Most engagements will move through the following stages / phases . *Discovery - Inception - Develop & Deploy - Evolve / Handover*
- Some initiatives will have the definition of the product happening as concurrent tracks of work.



WHO: Client, CP, and Team (TL, DP)

IT'S NOT: Set in stone or linear

Discovery

TW Extended methods deck includes a toolbag of techniques that cover both Discovery and Inception methods.

Overview of some example activities and timeframes for a Discovery.

Discovery Output Examples:

- Genie / Product Innovation
- Acumen / Product Innovation
- ASC / Responsive Org (and original proposal)
- QTAC / product Innovation

WHY: To discover and understand the client's business, customer and technical context. This enables you to then identify opportunities, define and prioritise them in order, provide recommendations for next steps and then further evaluate them before committing to more delivery investment.

WHAT'S THE DEFAULT?

- Client business goals to determine direction and how success will be measured
- User/Customer research to understand current needs, goals, and pains
- Expert review of existing client solution to understand how well it supports the business and user/customer
- Tech assessment of architecture and client team delivery practices to uncover impediments for future solutions
- Competitor/substitute research to assess how well served the user/customer is by alternate solutions in market and what features and benefits are used to differentiate them.
- Product vision and roadmap development to focus the team and align client stakeholders on the future challenges to solve in order to get there
- Opportunity identification, definition and high level prioritisation for triaging potential solution options (i.e. Bets) for new or enhancements to existing products, services or features.
- Recommendations and workshop/showcase to communicate and gain buy-in for next steps, to further evaluate prioritised potential solution options (i.e. Bets) and suggested team structure

The following could be part of an extended Discovery (prior to an Inception) in order to gain greater confidence for a specific Bet:

- Experimentation (MVPs not a release) to evaluate riskiest assumptions (e.g. customer demand, channel, value proposition, pricing) of prioritised Bets
- Proof Of Concepts (PoCs) to evaluate suitability of potential solution technologies that Bets are dependent on
- Financial modelling to determine potential revenue generation, savings, margin and costs of Bets

WHO:

- **Product Innovation and software delivery or Intelligent Empowerment data offering?** Differentiated value for our clients is uncovered through a cross functional team of Principal/Lead and Senior Product/BA, Tech and XD consultants to identify and evaluate the highest value desirable/viable/feasible, potential solution.
- **Client**: With the above, TW team are pairing with client counterparts to harness their domain expertise and have a client exec sponsor to engage and provide support and advocacy within the rest of the client organisation.

HOW LONG: 1-2 weeks minimum, scaling depending on scope of ambition, but up to 12 weeks if inclusive of multiple rounds of further evaluation activities.

IT'S NOT: Unstructured/unfocused research and analysis with no specific objective or "deckware" handoff

---

Inception

This TW branded Inception deck gives a **SUPERB** explanation of the different activities that can be performed and during an inception workshop to get to the outcomes

This picture gives a great indication of the phases of an inception

Here is a playbook of how to run an inception, including some tips for facilitators

Using these cards or using this Inception Planning board will help with planning and running an inception

Here are some CFR resources that may be helpful

WHY: To build a shared understanding of what your project aims to achieve, build a delivery plan and tech vision as guides to achieve this and identify and assess risks that may impact you.

WHATs THE DEFAULT? That Projects start with a series of workshops that will deliver;

- Agreed vision for the project
- Understanding of the target customers
- Cross-functional requirements identified and prioritised
- Technical Vision - (for more detail on this see the ***Technical vision*** section below)
- Prioritised & estimated story backlog (not epics, or features)
- Realistic plan for delivery (see planning below) including a staffing plan that's agreed and communicated with the client
- Agreement on governance/cadence/communication plan and tools
- Initial RAID list
- Plan for establishing a path to production

WHO:

- Client - Those who have appropriate understanding of the business, technology, domain and of the challenge that the project will address & at least one person with decision making authority and someone who can share the clients expectations for the project
- Team - As many of the team who will be delivering the work as is feasible - and for a software delivery project *definitely* folk who are able to: facilitate the workshop sessions (one of these people should have been in at least one inception previously and be comfortable leading the facilitation effort), establish the customer, technical vision and solution, establish and estimate a backlog of work and establish a plan

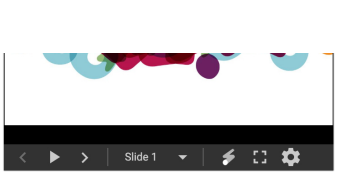
HOW LONG: usually 1 or 2 weeks + time required to prep before the inception.

IT'S NOT: Upfront Analysis Phase

Onboarding

WHY: Account Onboarding:

- Improves Account **cohesion**
- Improves **Confidence** and **quicker ramp-up** of new joiners
- Reduces **miscommunication/guesswork**



Link to Onboarding deck [2018.04] template

Here is an e.g of an email may be sent to new joiners

WHAT's THE DEFAULT : A conversation usually supported by a deck or email that addresses

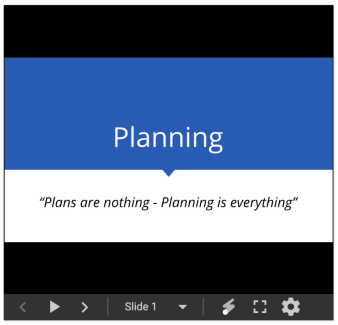
- The FAQ's that a new joiners have when they arrive on an account
- The Statement of Work (SoW) and team profile
- The Security Onboarding roll on check list and incident response plan from a security perspective
- The account strategy and past and current work

WHO: Delivery Principal and anyone new to the account

WHEN: ASAP when a new person is assigned to a project

HOW LONG: Up to 1 hour

IT'S NOT: A replacement for reading the SOW outputs of inception and any release charters that have been completed.



The above deck shares more about the defaults and Links to resources that share how to plan for an iteration & groom/refine your backlog and plan a release

This link Delivery Plan - Release Plan -- Template' is a sheet that helps you calculate a Delivery/Release plan . It also has a burn up template for ongoing tracking of your project.

Here is the link to a Project / Release Charter

- A release charter documents your plan & supports alignment and expectation management with the team

Planning

WHY: We need to be able to make informed decisions about projects. Plans are are a key input for those decisions

WHAT'S THE DEFAULT?

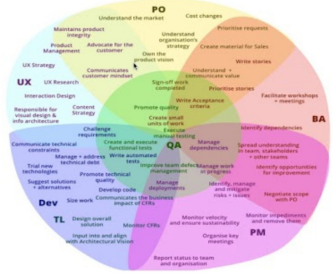
- Individuals in the team plan their day to day work and communicate this at stand up
- Teams plan the work for the coming iteration by grooming/ refining the backlog and planning the work that will be tackled during the iteration.
- There is a plan for the next next major bucket of work to be taken on by the team. This plan should show a roadmap of what needs to be done when, key dates, and deliverables.
- ▪ This is established using estimates (in points) of the scope and the velocity of the team.
- Plans are not set in stone, they are easy to change and are updated regularly in response to change

WHO: The team & key stakeholders

WHEN: *Daily* - individuals. *Once an iteration* - the team. *Before* beginning the next major bucket of work to be taken on by the team. *Continuously*- in response to change.

IT'S NOT: Set in stone, done once, or followed no matter what changes

---



Here is a guide to the roles within the CST

Roles & Responsibilities

- CST
  - Delivery principal & Technical principal & Exec sponsor &Lead BA or PM
- Tech Lead (see following section)
- PM / IM Responsible for
  - Managing plan - Project reporting - Manage deployment - Stakeholder management - Organise key meetings - Remove blockers - Manage RAIDs - Plan & manage non-SW delivery tasks
- Dev Responsible for:
  - Developing code - Writing automated tests - Promoting technical quality - Sizing work - Addressing technical debt - Assessing & implementing new tools and technologies - Communicating tech constraints
- QA Responsible for
  - Promoting quality - Creating & execution of functional tests - Planning & managing test strategy - working with the BA on acceptance criteria - Supporting UX in identification of usability bugs and debt -Management of the defect resolution process
- XD Responsible for
  - Discovery - UX research - UX strategy - Advocate for customer - Content strategy - Interaction design - Information architecture - Visual design - Product management
- BA Responsible for:
  - Analysis - Story elaboration - Manages change to the backlog and stories - Roles and personas - Facilitates workshops - Identifies dependencies - Negotiates scope - Managers / Refines backlog.

Tech Leadership

Whilst we call these "leadership" activities, it's the responsibility of the whole team that they are done. Your Tech Lead should ensure they're happening, but doesn't have to be the one to do it.

- Target Architecture diagrams clearly visible and accessible by the team
  - You'll probably need multiple of varying degrees of depth. Simon Brown's C4 Model is a good model to follow for structuring the different levels of diagrams
  - Make sure everyone can update it. e.g. Don't use tools you have to pay for unless everyone has a license
- Document your decisions. Any important decision should be recorded.

    - Even if we're all friends, at some point someone is going to ask why a decision was made. Having evidence will make everyone's life easier.
    - Commit them to your source control if possible. Nat Pryce's adr-tools can help.
- Organise regular Tech Huddles. Get your techies together and talk to them. Do it in a place where you can talk openly and draw big diagrams. These huddles are a good opportunity to reinforce the technical culture in your team.
    - It doesn't have to be only devs, but it should be a safe space, so manage attendance and don't make it an open invitation.
    - Discuss any decisions which have been recently made
    - Try periodically redrawing the architecture. Get someone different to talk through it!
    - Do collective code-review sessions for interesting or significant features.
- Conduct and record Post Mortems of incidents. Security, service outages, failures of process, etc... They're great learning tools.
- Manage your technical debt
    - Track it. Record your technical debt somewhere e.g. card wall or JIRA. . Get your team to add things whenever they're feeling pain and bring it up for discussion during iteration planning..
    - Fix it. Encourage your devs to refactor as they go. If something feels too big, talk about it in a Tech Huddle and dedicate some time to fixing it properly. It is acceptable to spend some percentage of the time developing a story addressing technical debt. Always leave code in a better state than you found it.
- Curate a team Tech Radar
    - Try Build your Own Radar or even just a spreadsheet
    - Focus on your own team or project's needs, not the industry.
    - Help your team understand what technologies are preferred, and get them to update the radar periodically

---

Technical Vison

WHY?

A technical vision

- Builds client confidence that we are working towards a plan
- Captures and communicates technical assumptions
- Gives us a shared starting point to evolve and validate as we deliver software

WHAT's THE DEFAULT?

The team develop a technical vision that is communicated to and agreed to by the client.

WHO (and WHEN)

Client and delivery team, this should be considered the key technical output of any kick off or inception activity that we take part in

HOW LONG

Depending on the technical complexity of the delivery expect to reserve between 25% to 50% of your inception for technical activities

IT'S NOT

Upfront design or making technical decisions without the team or client's input

INPUTS

Business goals, cross-functional requirements (CFRs), industry best practice, client technical constraints, case studies

OUTPUTS

Candidate Architecture

A diagram showing how the parts of the solution fit together and communicate. It should:

- Show the services, data stores, queues, etc. that make up the solution
- Show the underlying providers for each service (e.g. AWS, in-house)
- Show any crucial authentication or access control mechanisms
- Show the key data, user, information flows which generate business value
- Be shared and reproducible by the whole team

Prioritised and evaluated CFRs

e.g. not just 'response time is important' but 'a 50ms end-to-end response time is acceptable'

Tech Stack

What languages, tools, services and libraries will we use? Why are we using them? Decided in consultation with the team and client.

Path to Production

A concrete plan for establishing CI/CD as quickly as feasible

---

2) During Delivery

Iteration 0

Refer here list of typical Iteration 0 Activities

Iteration 0

WHY? Allows the team the time to get set up to delivery

WHAT'S THE DEFAULT? Team from the inception (UX/ PM/ BA/ TL + QA) +*Dev complete the tasks needed to start work.

* the number of Devs will depend on the nature of the work

Ways of Working

A Good place to find the whys for the Ways of working is the Agile Samurai written by a former TWer its a great place to go to understand why we do what we do.

Here is a link to a Trello board that shares ways to run a retrospective.

Here is a link to how you may plan an iteration

Here is a link to how you might prioritise - map & split stories

Iterations :

WHY: Iterations establish establish a cadence for our work. They provide a framework around which we schedule - routines - rituals/meetings - governance - planning sessions - showcases and progress reporting that establish, drive and report on the work in the time box.

WHAT's THE DEFAULT

- We plan and work in iterations
- Iteration length is agreed by the team and the client at the start of the engagement. For software delivery they are typically 2 weeks long - except on short engagements < 8 weeks, where 1 week iterations are generally more suitable
- We schedule the rituals and meetings that support the work around this time box.
- We provide an iteration report, every iteration, and meet with key stakeholders to discuss it (see reporting default below for *why* etc)

Rituals :

WHAT's THE DEFAULT

- Daily stand-up meetings (see *content pending* for why)
- Retrospective meetings to drive improvement usually at the end of an iteration (see *here* for why)
- Iteration planning meeting before the start of the iteration
- Showcase meetings at the end of an iteration where we present our work to our team and stakeholders regularly usually at the end of an iteration (see *here* for why)

Practices

GENERAL

WHAT's THE DEFAULT

- We deliver working software as often as possible.

USER STORIES

WHAT's THE DEFAULT?

- We break our work into user stories
- We estimate the user stories - we don't count cards
- We track/manage these stories in a backlog which has a priority order that is influenced by the team but determined by the Product Owner. The backlog also includes tech tasks, spikes (time boxed investigations) and bugs
- We elaborate 'just enough' stories to ensure there's no wait time for the team ahead of the iteration starting. This usually means that the team have a enough stories ready to be worked for the next iteration and a small buffer of additional user stories
- Elaboration & prioritisation can take place in backlog refinement (grooming/ management) sessions.
  - Regular prioritisation of scope (new stories and stories created as a result of backlog refinement)
  - Regular discussion of the impact to the project (time lines, cost) as a result of prioritisation.
  - The goal is to elaborate just enough - just in time - Here is some information on why and how you could do this.
  - Additional information on iteration planning

TRACKING PROGRESS

WHAT's THE DEFAULT?

- We track the progress work on *stories*, tech stories and bugs during an iteration using the team wall (see below).
- We have a mutually agreed Definition of Done for a story. It must include deployment to a prod like environment.
- We track velocity (the number of story points completed or "done" by the team during an iteration) and the cycle time see tracking below for more detail.

REPORTING - See details below

WHAT's THE DEFAULT?

- We report on the work completed
- Progress towards goals

NOTES - See details below

WHAT's THE DEFAULT?

- We document the outputs (attendees, decisions and actions) of meetings and share them.
- See below for meeting notes and agendas

RAIDS - See details below

WHAT's THE DEFAULT?

- We Proactively identify and actively manage & track RAIDS

Specific technical defaults:

- Code is kept in a version control system
  - Changes are pushed frequently, at least once a day but more frequently is preferable
- We do Test Driven Development (and developers write the tests)
- We pair on stories and rotate pairs regularly. Weekly is too long! and don't wait for stories to end.
- We have a Continuous Integration (CI) server
  - Tests run every time code changes are pushed
  - We have visible ways of notifying people that builds are failing
  - Failing builds are addressed quickly and developers do not push changes to a broken build
  - A bug fix is the same as any other deployment
  - We have quality controls in our builds (which check for security vulnerabilities, code quality, etc...)
- We practice Continuous Delivery
  - We do Trunk Based Development with Feature Toggles
  - We don't have long-lived feature branches (half-day at most)
  - Any green build is a production candidate
  - We have a clear path to production and we deploy to production frequently
- Our infrastructure is automated, created as code, and updated through our CI server
- Our application is observable in production
  - We have aggregated logs somewhere accessible by the team
  - We understand how our system behaves in production, and know when it's unhealthy

Specific Quality defaults:

- Trade-offs are made on the Fidelity of a solution, never on Quality
  - We don't compromise on Test Driven Development or Pair Programming
- Cross-Functional requirements are part of the acceptance criteria of every story rather than breaking their implementation out into separate stories
- There is an effective mechanism for quality and change control in place (preferably through Peer Reviews - not just as part of Pair Programming)
- "Spikes" that get deployed to Production or are not spikes (they should be short-lived and throwaway)

Specific Analysis defaults:

- Understand the problem we are here to solve
  - Map out the current and to-be process and identify gaps
  - Map out customer journey to understand the customer experience
- Establish the Backlog
  - Identify and capture business and cross-functional requirements
  - Stories are broken down to the smallest possible size while still delivering clear value to end users -
  - Elaborated stories should include the following:
    - Context/description
    - Narrative
    - Assumptions and questions
    - Acceptance criteria
    - Technical notes (optional - dependent on the work)
    - Prototypes (optional - dependent on the work)
- Keep the backlog up to date and prioritised (see above stories)
  - Collaborate with the team to make sure stories have been analysed, elaborated, and estimated by the team prior to iteration planning
  - Socialise information to the team and beyond through information radiators (i.e. story wall, trade off sliders, customer journey, story map, etc.)
- Support the delivery of the backlog
  - Kick off stories for development (3 amigo's)
  - Support development with answers to questions during dev
  - Check delivered stories against acceptance criteria
  - Support the Product Owner through the story acceptance process.

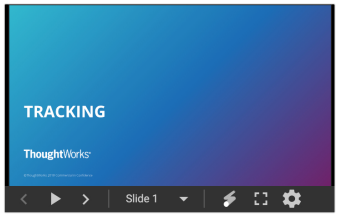
Specific Product defaults:

- Content Pending

Specific Experience Design defaults:

- **Talk to your customer**
  - Validate your assumptions and design solutions with the end users of your product (through user research and user testing)
  - Don't spend time perfecting your ideas. User testing should be as lightweight as possible. Early in the process, you can test with paper prototypes, and as you gain confidence in the solution, your tests will be higher fidelity, using the code that your delivery team is creating.
  - Involve your whole product team in user research and testing - bring them with you on the journey of understanding your end users.
  - Gather feedback from users throughout and use the most appropriate methods for the stage of delivery.
- **Focus on little and big picture at the same time**
  - Design in parallel with delivery. Your analysis and design work will be an iteration or two ahead of delivery.
- **Design collaboratively**
  - Pair with developers and testers to solve design problems as a team - design is a team sport.
- Understand accessibility requirements, design for all users and test for accessibility as required.
- Use existing design patterns and styles that your users are familiar with

---



This 'Delivery Plan - Release Plan -- Template -has a burn up template for ongoing tracking of your project.

Here is a link to a commercial tracker that you could use to track planned vs actual spend.

Tracking

WHY? To - Understand progress towards the project goal - Support continuous improvement & decision making.

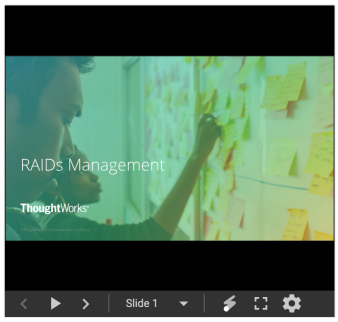
WHAT's THE DEFAULT?

- Plan (see above) using an estimate of scope based on
  - stories (not epics or features) &
  - story points
- Track progress vs this plan;
  - Establish a burn up chart that shows:

    - the goal established in the plan as estimated scope & the rate of scope growth

    - path to meeting the goal as estimated velocity
  - Track progress towards the delivery of the estimated scope as actual velocity
  - Make this progress visible to the client and the team (share it @ showcase, have it on your wall include it in any status reporting)
- Track the cycle time of the stories
- Establish an expected budget and track actual spend against it

WHO: The Delivery Lead is responsible for production of this tracking. The Delivery Lead and TL pair on sharing/discussing the contents with the client. The team help pull the info together.



Here is a link to a RAID tracking sheet

Here is a link to a RAID Playbook that sets out how you may proactively identify and manage RAIDS.

Here is an example on how to manage tech debt.

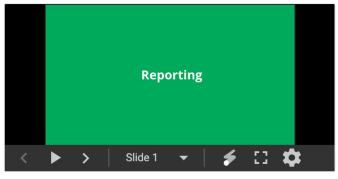
RAIDs Management

WHY? To - Properly mitigate risks reduces or eliminates angst amongst the team members and program leadership - there are not surprises for the executives. Solve issues permanently, avoids the waste and frustration of re-occurrence.

WHAT's THE DEFAULT?

- Record all RAIDs
  - All risks and issues must be recorded
  - All dependencies must have an associated risk
  - All assumptions must have an associated risk
- Proactively manage RAIDs and track progress towards its resolution or management
  - Regularly spend time with the team and stakeholders identifying and managing RAIDs
    - i.e. once a week or fortnight meeting review of RAIDs
  - Maintain a permanent record with any changes to the RAIDs
    - All risks and issues should have actions, owners and due by dates to address them
  - Make RAIDS visible to the client and the team (have a visual system and include prioritised RAIDs or any changes in them in any status reporting)

WHO: The Delivery Lead is responsible for ensuring all project RAIDs are captured and documented and incorporating the teams' inputs. The Delivery Lead and TL pair on sharing/discussing the contents with the client. The team help pull the info together. Subsequent actions taken from the capture of RAIDs should be determined by the project lead with support from subject matter experts and select team members.



Reporting

WHY: To support conversations around the status of your project.

WHATS THE DEFAULT? Track (see above) and Report your progress Regularly with your client & with ThoughtWorks.

WHO: The ThoughtWorker responsible for ensuring this is done and for having the conversation with the client

The above is a deck all about reporting: why, how. It includes 3 **sensible templates** to use, dependent upon your type of delivery.

Here is a link to the 'Delivery Plan - Release Plan -- Template' s-sheet that helps you calculate a Delivery/Release plan (based on an inception), it incorporates lessons learned with some pre-sets.

It also has a burn up template for ongoing tracking of your project.

about the contents. It should also go to: Client, The Team, TW's: CP, Delivery Support, Office Tech Principle

WHEN: Weekly or fortnightly (as per your iteration length)

IT'S NOT: A replacement for having the conversation with your client.

- The Visual Management toolkit contains presentations about visual systems, examples and templates (walls, cards, badges, etc.) for building a Visual Management System

Physical wall & digital tool

Physical Wall (Visual management)

WHY: Create a shared understanding of process, work in process and priorities

WHATS THE DEFAULT? A physical card wall with Backlog, Prioritised, Analysis & Design, Ready for Dev, Dev, Ready for testing, Testing, Ready for sign-off, Done

WHO: The Delivery Lead is responsible for setting up the visual system. The team are responsible for maintaining the information displayed i.e. moving and creating cards

NOTE: Whilst a physical wall is a recommended as a sensible default for seeing the flow of work and for the team to share. We also recommend it is used in conjunction with a **digital tool** (e.g. JIRA, Excel, Trello) for capturing the detailed acceptance criteria, attaching wire frames etc for each story card. Electronic systems should compliment the physical wall rather than replicate it 1 to 1, this reduces the overhead of using two systems whilst providing an audit trail and

---

Security incident report template

Security

This is everyone's responsibility, not just the Tech Lead or Developers in your team. Do read the checklist it won't take long and it's IMPORTANT!!

We always follow the general principle of never disclosing identifiable client information. Below are some practices we should follow:

- Read the **delivery project security checklist**. In particular:
  - make sure your team has onboarding/offboarding checklists and records, and that these include access revocation and deletion of client intellectual property when offboarding (example).
  - make sure your team has an incident response plan in place and you know how to follow it (example).
  - follow the security low bar for all TWers, including enabling FileVault and using a strong password on your laptop user account.
  - NEVER EVER put plaintext secrets e.g. passwords, API keys or account credentials in a source code repo. Irrespective of whether clients do this it does not excuse us from knowingly participating in this practice. There are lots of ways around this. See the attached secrets management cheatsheet, and if you have questions, talk to Robin Doherty.
- Don't use your personal GitHub ID for ThoughtWorks or client related work work. Maintain a separate Github account using your ThoughtWorks email address so there is no confusion between work and personal code/projects.
- If you're giving a public presentation that even hints at some client work you've done, get their consent.
- Before you have to use it, familiarise yourself with the process for removing information from Google's caches.

SAMPLE AGENDA

- Intros
- <The purpose of the meeting>
- Any other business
- If you're feeling fancy - note who the attendees will be and what areas they're coming to represent.
- Recap actions and decisions

Agendas and Meeting Notes

We have learnt the painful lesson that it is **worth the effort, to send follow up emails to all stakeholders after any meeting / decision making conversation:**

WHAT:

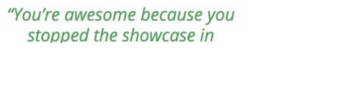
- When sending the meeting calendar invite add the purpose and a basic agenda
- After the meeting, send a simple email that confirms our understanding of key outcomes, key decisions, actions and next steps, who was present and who was absent (based on the invite).

WHY: It's a useful way of ensuring we have a shared understanding AND a useful reminder for everyone involved should there be queries later.

It is also a great way of managing those stakeholders who don't attend the session but then want to change the outcome afterwards. We recommend an email with "*Thanks to everyone who could make it. For those who couldn't, here is a summary of outcomes, decisions and actions ...*"

HOW LONG: 15 mins

WHO: Nominate someone before or at the start of the meeting. Do share the load. If all else fails then the Delivery Lead is responsible for making sure this happens.



Feedback routine

order to accommodate all the stakeholders."

- A great intro to feedback by Nick Thorpe
- Another great presentation on giving effective feedback to other ThoughtWorkers

Feedback routine

WHY: ThoughtWorkers need regular, specific and actionable feedback in order to improve their effectiveness. Due to the nature of our flat and flexible structure which does not have consistent line management this responsibility needs to be shared by all ThoughtWorkers.

WHAT'S THE DEFAULT?: Speedback and 1-on-1 catch-ups are the most common forms of feedback *however* what works for one person or team will not necessarily work for another.

WHAT:

- Constructive feedback to increase effectiveness "Try doing X instead of Y when you ...."
- Reinforcing "The way you facilitated that meeting was excellent given the number of different points of view". Do more of this!
- Formal. Feedback which if not addressed has the potential to be or is damaging your career. Do not wait for a routine catchup in these situations.

WHEN: Routinely e.g. once per iteration AND when required or deserved

HOW LONG: 10-60 mins

WHO: Everyone, not necessarily all at once.

- Content Pending

Account meetings

WHY: Ensure alignment across the whole team and make sure people focused on the detail have an understanding of the overall context so they can be more effective and make better decisions. Make sure CPs are aware of potential opportunities.

WHAT'S THE DEFAULT? A regular meeting once per month on engagements of 6 months or more, more frequently on smaller engagements assuming the team is of a reasonable size. On very small teams e.g. 3 people doing an assessment there are other mechanisms to maintain alignment.

WHAT:

- Agenda
- Actions from last meeting
- Update from CP about overall relationship and high level strategy and key risks
- Staffing changes / rotations
- Client feedback, this is what they're saying, what are you hearing?
- Account hygiene / admin (review performance metrics CGM, leakage, Outstanding invoices, leverage, team capacity etc..)
- Updates for key team members e.g. DP, TP
- Potential opportunities
- Open Q&A
- Team drinks ;-)

Make sure document and share minutes from the meeting for people unable to attend or who join the account later and need context.

An alternative formate is a stand-up style "What do il know that you don't know?"

HOW LONG: 30 mins

WHO: Make sure the whole team including exec sponsor is invited.

3) Ending a gig

Handover

HANDOVER TO NEW TEAM MEMBER

Handovers take place in different ways. Sometimes you have in-person handovers and sometimes people will come after you have left and have to pick up things cold. Remember it could be you on the receiving end of a handover on your next gig!

WHAT TO DO

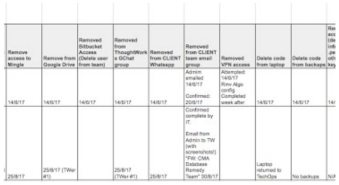
- Create a handover checklist that's been validated with your team to make sure you've got all your bases covered. It can include:
  - Orientation information about the client
  - On-boarding information
  - Work that the new starter will be taking over from you
  - Context of the history of the work to date
  - Any outstanding risks or issues
  - Any upcoming deliverables
  - Introductions to key stakeholders and other team members
  - Team norms and social activities
- If in-person, walk the new starter through the checklist and pair for the duration of the handover to help on-boarding
- If not in-person, more detailed documentation (which can include videos for example) about what to expect can be really useful as well as finding an on-boarding buddy for the new starter.

HANDOVER TO CLIENT - END OF ENGAGEMENT

Final handover to the client also happens in a few different ways. Some are more planned than others, but it is *essential* that certain events occur regardless. Payment of invoices by the client is contingent on ThoughtWorks fulfilling its obligations in the relevant Statement of Work (SoW).

WHAT TO DO (CLIENT PRINCIPAL/DELIVERY PRINCIPAL)

- Understand the deliverables from the SoW and make sure they have been completed by the team
- Ensure that the client has accepted the deliverables and agrees they have been met
- Ensure that any artefacts are packaged and delivered to the client in an agreed format & location
- Ensure ThoughtWorks has a central copy of artefacts that can be recovered (within contractual limits)
- Ensure close-down/rolloff procedures have been followed for all team members
- Take the team out to celebrate the end of the engagement!
- Client Principal should check in with the client to ensure handoff has been successful after all hand-off events have occurred



Close down / Roll-off

When a team members rolls-off:

WHATS THE DEFAULT?

- Update your onboarding/offboarding checklist, these should include access revocation and deletion of client intellectual property when offboarding (see Security section above for link).
- Delete client intellectual property (such as code) from your laptop when you roll off and ensure that your access to client systems, including code repositories, is revoked. This is usually required by our contracts with clients. There are no acceptable reasons for keeping code in breach of our contracts.
- Ensure that access to client systems is revoked – your client will likely have their own process – return access passes etc. Confirm that this has been complied with.

When a TW's rolls-off:

WHATS THE DEFAULT?

- The steps above Plus
- Archive/save relevant documentation for the full project - taking into account the security guidelines
- Final invoicing
- End of project Feedback
- End of project Retro

Go Live Email

WHY: Celebrate success with your colleagues, share learnings and insights. Also very helpful if you are trying to understand what we've delivered across our offerings and across different industry domains and very helpful for sales.

WHAT's THE DEFAULT? An email to go-live@thoughtworks.com. and if you want to send the announcement to TW Globally do-not-reply-to-all@thoughtworks.com and to australia@thoughtworks.com if you want to send it to AUS with the title [Go Live] <Project name> - <Client> -<confidential or public>, a description of your project, the team, their successes and learnings.

WHEN: At the end of an engagement or major release

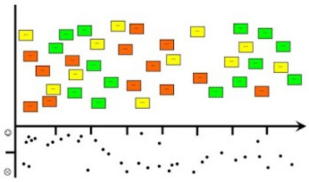
Here is a template for the email, in your project details it's a good idea to include:

- The project goal
- Timeline
- Lessons learning / insights
- Client feedback

Make sure you state how sensitive the information in the email is up front. You should also include photos and images, with at least one of the team.

When listing team members don't forget people who've rolled off or played supporting roles from the TW office or elsewhere!

Go live email HOWTO



End of Project Retro

WHY: The Project Retrospective dedicates time to reviewing a completed project and learning from both the successes and the failures so the team and ThoughtWorks can improve how they work going forward.

WHAT'S THE DEFAULT?

A project retrospective must be run at the conclusion of every project and findings shared with the Delivery Support team.

Preparing for the Project Retro

- **Identify your audience**.

The team participating in the retro include current team, client principal and other key players that may have rolled off the account or been involved in critical decisions or interventions throughout the project.

- **Select and prepare a retrospective activity to facilitate the session**

In the majority of cases, Project Retrospectives will be a reflection of many months/years and/or phases. A

In the majority of cases, Project Retrospectives will be a reflection of many months/years and of phases. A timeline retrospective is a good choice when a team would benefit from reflecting back over a period of time longer than a single sprint.

- **Assemble a project report and timeline**

Reflect on the project objectives and prepare a timeline with includes major events, phases and milestones. This will form the retro activity. Also, review the original project definition, success criteria and any metrics you have regarding the project's outcome.

- **Refine the agenda**.

See details below about the session

- **Schedule the meeting in advance and invite the team**

Ask them to come prepared with their key insights, observations, and ideas for improvement.

- **Get supplies**.

Several retrospective techniques require additional supplies, such as sticky notes or online voting systems. In-person meetings benefit from snacks!

Running the session

1. Welcome

First, welcome people. Confirm for everyone what the meeting end result will look like, and the process you'll use to get there.

Then, if you have people who don't know each other well, run a round of personal introductions.

Finally, set the tone by sharing the Retrospective Prime Directive.

THE PRIME DIRECTIVE SAYS:

'Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand. At the end of a project everyone knows so much more. Naturally we will discover decisions and actions we wish we could do over. This is wisdom to be celebrated, not judgement used to embarrass.' *Norm Kerth, Project Retrospectives: A Handbook for Team Review*

2. Project Review

Next, make sure everyone has a shared view on the project. You can do this one of three ways.

Option 1: Ask the group to talk about it.

For shorter projects or for mid-project retrospectives, you can ask the group to discuss the facts. Questions to ask: What was supposed to happen? What actually happened? What did you set out to achieve? What was your plan to achieve this? How did this change as you progressed?

Option 2: Share a report

The project leader presents a project report, and the team comments along the way.

Option 3: Use the prepared timeline or create a shared timeline

For example, see the Peaks and Valleys exercise

This is one way to create a shared timeline. It takes longer, but it makes for a better conversation and a stronger shared experience. And it's fun!

3. Timeline Retrospective activity

Set the Stage:

There are two areas of emphasis for the timeline retrospective:

- Seeing how events are distributed across a period of time via a very visual representation. You may also define some topics to easily group categories. eg. Account Health, Technical Delivery, Business Delivery, People & Culture
- Getting a read on how team members felt about where the team was at particular points in time along the timeline

Ensure that you clearly state what period of time the retrospective is intended to cover.

Gather Data:

If using physical media, ensure that all team members have post its in three different colours. Each colour should be used for a particular type of event that occurred during the specified timeline & category, where:

- Colour A (green in the attached image) represents good events
- Colour B (yellow in the attached image) represents significant or memorable events
- Colour C (orange in the attached image) represents problematic events

Have each team member individually events of each type on the appropriate colour of post it and place it in the corresponding period and category.

Generate Insights:

Have each team members place their stickiest at the appropriate point in time along the timeline. Give team members an opportunity to talk about particular events. Encourage open conversation.

Generate Insights

Just below the horizontal axis, draw a smiley face and a sad face, one above the other. For each time point on the horizontal axis, ask each team member to draw a dot indicating how they felt about things at that particular point in time, where they felt:

- Happy
- Sad
- Somewhere in between

The events that various team members have posted typically provide clues about how they were feeling about things at any point in time. When all team members are done, there will be clusters of dots -- essentially a "happiness scattergram:"

Next, approximate the midpoint between the clusters of dots, and draw a line that connects each of those points.

3. Discuss

Priorities: What matters most?

Changes to Make: Recommendations for future projects

3. Consolidate

Present findings to Delivery Services team.

WHO: The Project Retrospectives should include current team, client principal and any other key players that may have rolled off or were involved in critical decisions or events.

trello.com/c/zHeiCpz8/49-timeline-retrospective

Want to find Content?

Try Cloud Search

Main Hub

P3 Hub

Project Hub

Communities

Regions

Functions

Policies and Compliance

Contact Us