Getting Started with Rational Team Concert
or
RTC in 16 Steps

Kai-Uwe Maetzel
IBM Rational Software
kai-uwe_maetzel@us.ibm.com

Rational. software

SDP 20

This Presentation is Good for You if

- You know what Rational Team Concert is
- You have seen demos
- You have downloaded RTC and played around with it
- You looked at the JUnit example
- You understand the basic ideas and concepts of RTC

- You would like to use it for real
- You are wondering what a reasonable approach is
- Your questions are still somewhat general but are getting more specific

User FAQ (Examples)

- How do I integrate with existing systems?
- When do I use connectors? When bridges?
- Can I adopt Build without adopting SCM, for example can I use build without moving off Subversion?
- What is the RTC security model?
- What is the project timeline?
- When do I use secondary timelines?
- When do I use iteration types?
- What are team areas good for?
- How do I structure my team areas?
- How do I categorize my work items?
- How do I express who is responsible for which category of work items?

Admin FAQ (Examples)

- What kind of licenses do I need, what are the options?
- How do I setup an installation for large number of projects and teams?
- How do I setup connectors?
- How do I setup an integration with BuildForge?
- What are the deployment options?
- Backup and restore

General FAQ (Examples)

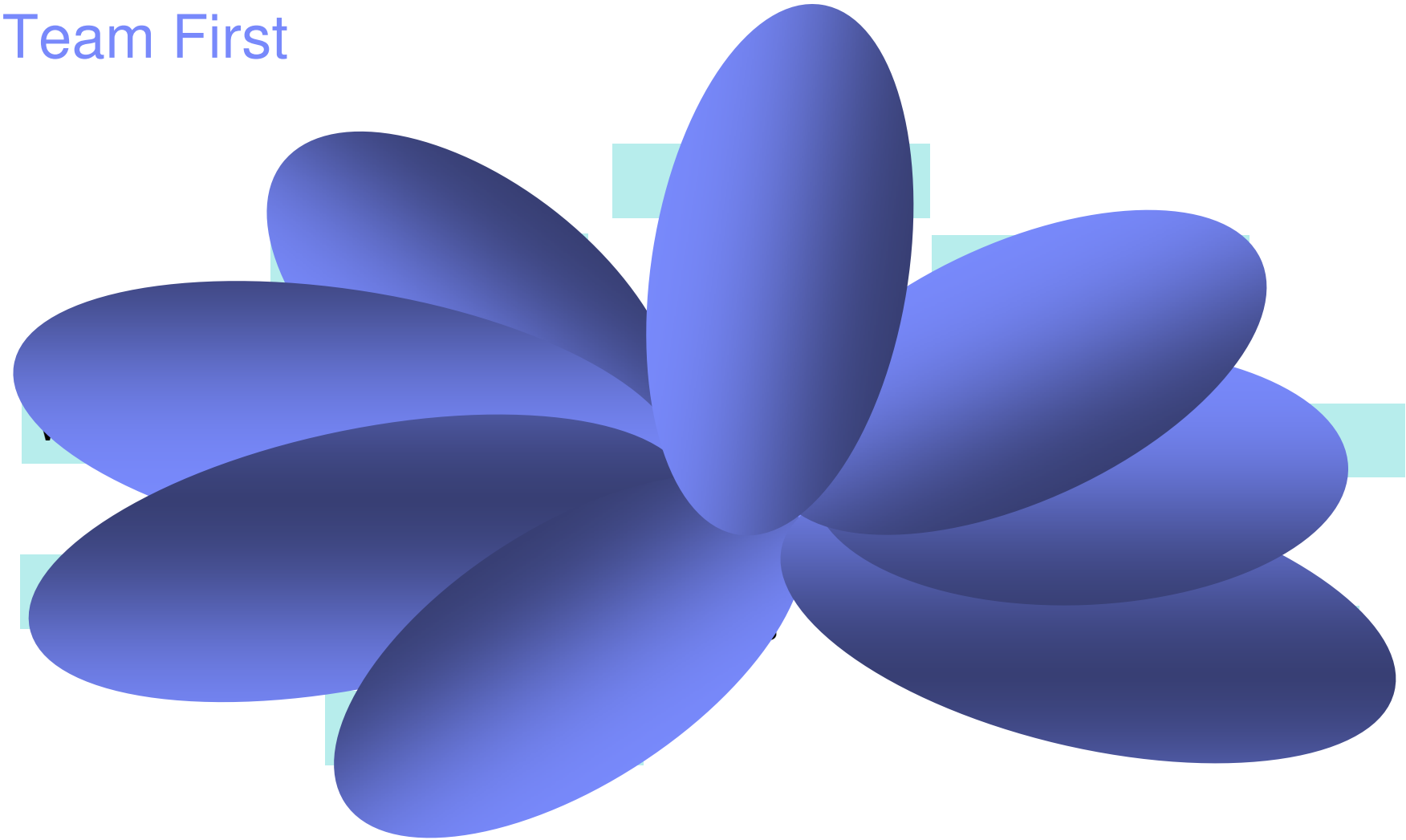
- RTC doesn't cover all the aspects of ALM. How do I integrate with other product which fill the gaps?
- What does the roadmap look like?

In This Presentation

- We will give some answers, but mostly **help you asking the "right" questions**
- We will not take the perspective of a particular process such as Scrum

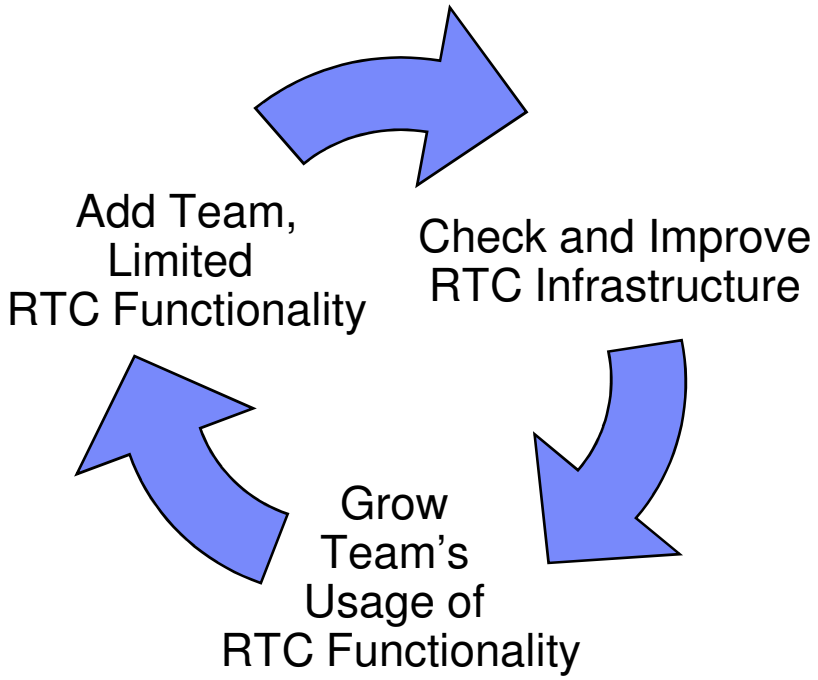
- For most topics you will find a **in-depth presentation at the conference**
  - C/ALM roadmap
  - Enterprise deployment
  - Connectors
  - SCM usage
  - Work item customization
  - Planning
  - …

Team First



- Teams are functional, not necessarily organizational teams

Classical Rollout: Start Small and Grow

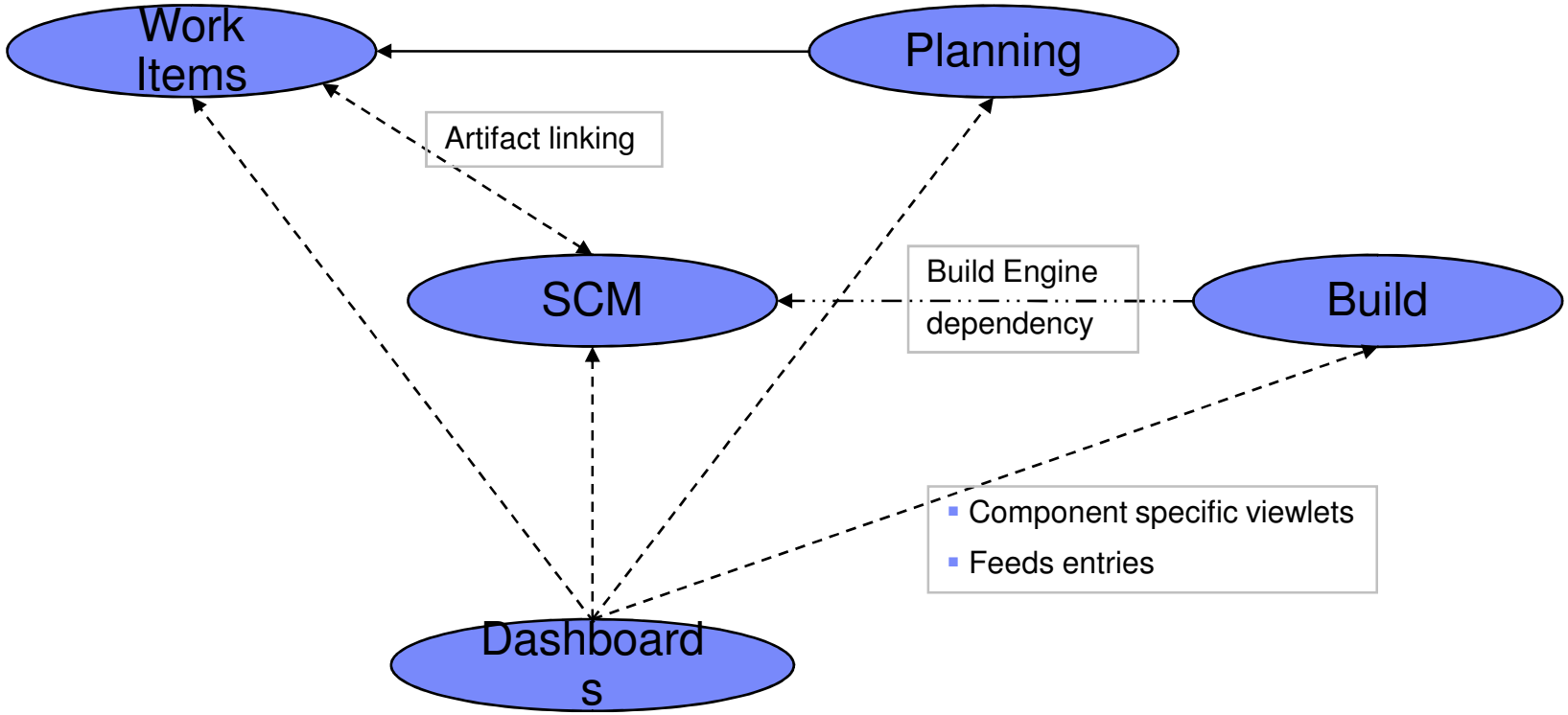


- Build up expertise in teams and leverage it for the on-boarding of new teams
- Don't overstretch teams with too much adaption of new technology at once

Adoption Paths of RTC Functionality

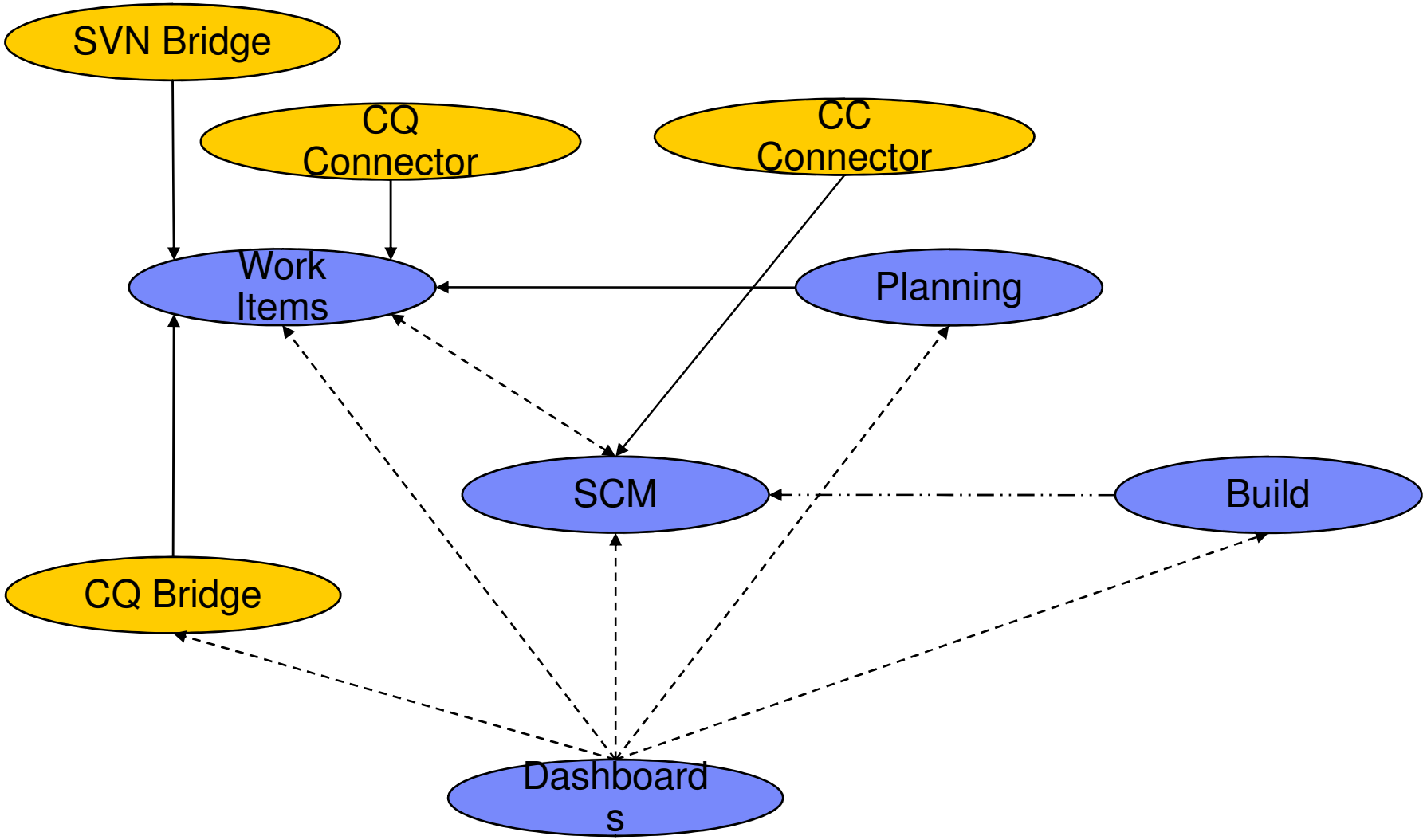
- Adoption paths depend on
  - Relationship between the RTC components
  - Your situation
  - Your priorities

Relationships Between Core Components



Possible Adoption Paths

Relationships of Integration Components



Your Situation

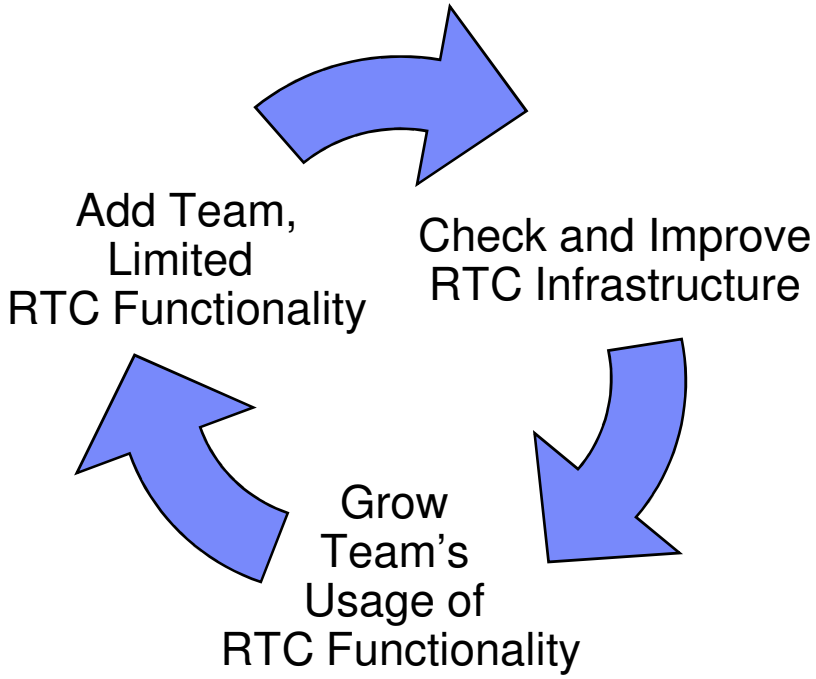
- RTC teams can operate independently (Island of Joy)
- RTC teams need to integrate with the existing enterprise architecture
  - Change requests
  - Build processes
  - SCM
- RTC projects are blank slate projects without cross dependencies
- An existing project moves over to RTC
- …
- …

Your Priorities (Examples)

- You have to move off your existing SCM solution because of …
- You don't have a useful reporting solution for your work items
- …

Scenario

We will walk through the cycle using the example of the adoption sequence given below.



Step 0: Install RTC Production Server

- Performed by IT Operations
- See installation guidelines

- Important points
  - Define backup and recovery strategy and test it on a test server
  - Use the right database for your anticipated growth, but you can also migrate later to a different database
  - Use the server's admin web UI to
    - Upload CALs to the server
    - Create user for the Repository Administrator
      - **Assign Developer CAL**
      - Assign to JazzAdmins repository group
    - **Disable ADMIN account otherwise all your security attempts are rendered useless**

Step 1: User Management

- Performed by the Repository Administrator; Web UI or rich client
- Define user management procedure
  - Establish rules for repository group membership
    - **Members of JazzAdmins have override rights and unlimited read access!**
    - **Every user with write access needs to be member of JazzUsers**
  - Connection to LDAP server or server-local user registry
    - Corporate LDAP servers are well suited for larger installation. When just starting you want to have more flexibility to experiment with repository groups.
- Create or import users (ongoing activity – can be automated)

Step 2: Create Project Area

- Performed by the Repository Administrator; Web UI or rich client

- Deploy out-of-the-box process templates
- Select the process template that is a good starting point for the project, no customizations at this point
- Add a project leader to the project area members list
- Assign appropriate process role to the project leader
- **Add the project leader as Project Area Administrator**
  - Project area administrators can break rules and thus rescue locked-down project areas
- Use RTC's "Invite to Join Team" to send invitation email to the project leader
  - Hand the project area over to the project leader

Step 3: Configure Project Area

- Performed by Project Leader; Web UI or rich client

- Configure the rules for **read access**; a project area is initially public
- Review the follow-up actions for ‘generate team invitation’
  - **What work items need to be created for joining team members?**
- Add team members to the project area and assign process roles
  - **Order the roles** so that the most relevant one is first
    - effects which preconditions and follow-up actions are effective
  - Send **team invitations**
- Configure initial Project Timeline
  - Optional start and end date
  - Optional break down into nested iterations with optional start end end date
  - The **Current Iteration** of the Project Timeline **determines the effective process rules** for all team operation executed in the context of the project area.

Step 4: Work Item Customization

- Performed by Project Leader or authorized team members

- Define **initial hierarchy of work item categories**
- Review and adapt work item types, their workflows, and their editor presentations
- Define the shared work item queries that should be available to the team
  - Share the queries with the project area
  - Add queries to the process customization if you think they should be available out-of-the-box the next time you create a project area.

- Review and adapt process rules
  - **Preconditions** for saving work items such as ‘required attributes’
  - **Permission** for all work item operations

Step 5: Populate Work Items

- Setup a **Connector** to your existing work item system such as the CQ Connector
  - Connectors are bi-directional but be careful about concurrent modifications
  - Since connectors map work items of two different systems onto each other, you might have to **revisit** your **work item type definitions** if there are mapping issues
  - **Use a functional user ID with the connector CAL**
  - Connector operations are protected by **process permissions**, so they need to be reviewed and adapted.
- Import work items from your existing system
  - Import is a one-time, one-way population of your work item base in RTC

Step 6: Customize the Project Dashboard

- The project dashboard is the key tool for self-reflection and improvements.
- Encourage team members to define their personal dashboards.

- Viewlets show
  - Static information -> Provided by the user
  - Trend and historic data -> Work item reports based on data warehouse
  - Just-in-time data -> Work item queries
- Assign the **Data Warehouse Administrator** the membership to **JazzDWAdmins**
- Setup **report and dashboard process permissions**
- If you can not show on the dashboard what you need to see you might have to revisit your work item type definitions, your work item queries, and you might have to define your own work item reports.
- **Continuously update and improve your dashboards**

Assessing the Adoption Work Items

- Remove the amount of conventions – everything that is not explicit does not exist
- If looking at the dashboard is not your first reaction when being asked a question, your dashboard is not well configured
- Avoid linear workflows: allow from all state to go to any resolution. This increases the willingness to use work items even for small tasks.
- Pay attention to the relationship between work items, these relationships reify a wealth of information
- Use cumulative work item types to show progress across child work items.
- Give users an explicit way to propose changes to the working environment:
  - Process rules and permissions, work item types, workflows, etc.
- Allow teams to customize as many aspects of their development as possible, only control the transition points.

Assessing the Adoption of any RTC Functionality

- What aspects of the adopted functionality are relevant for the project's success?
- Do you track those aspects on the project's dashboard?
- Have you configured role-specific rules and permissions?
- How many conventions do you use that are not explicitly enacted in RTC?
- How much do you require versus encourage?

Step 7: Adopt Agile Planning

- Use the adoption checklist…

- Start with plans for iterations
- Provide meta information about the goal of the iteration etc.
- Use the iteration plan to schedule the planned work
- Encourage team members to use the My Work view to schedule their work
- Adopt work estimation and thus risk assessment
- Adopt or define additional plan types

Step 8: Adopt SCM

- Use the adoption checklist…

- Start with one SCM stream owned by the project area
- **Structure your code into Components**
  - Start with at least two components one for the team's code, one for the prerequisites
  - Baselines are scoped to Components
    - Questions to be answered
      - Who is responsible for creating and testing the component baselines?
      - How does that fit with the geographical distribution of your team(s)?
    - “Architecture follows organizational structure”
      - One team owns usually multiple components, a component is owned by one team
- **Leverage linking of work items and change sets**
- Define the preconditions for SCM deliver

Step 9: Populate Your Source Code

- Setup a **Connector** to your existing SCM system such as the CC Connector
  - Connectors are bi-directional
  - **Use a functional user ID with the connector CAL**
  - Connector operations are protected by **process permissions**, so they need to be reviewed and adapted.
- Import code from your existing system
  - Import is a one-time, one-way population of code into RTC

Step 10: Adopt Build

- Use the adoption checklist…

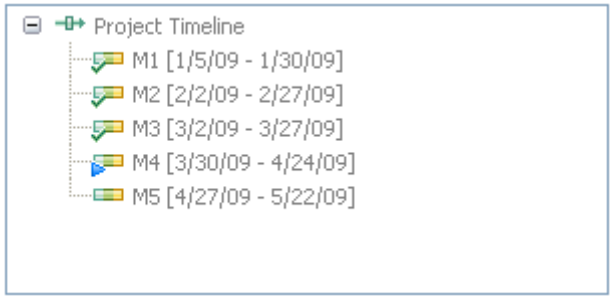
- **Use a functional user ID with the build CAL**
- Start with an periodic integration build that uses the content of the team's stream
- Leverage linking of work items and build results
- Leverage personal builds for team members
- Introduce a **distinct work item type** and queries **to track builds**
- **Track** the **quality** of builds **on the dashboard**
  - Predefined reports are available out-of-the-box; good starter set:
    - Build success history
    - Most frequent test failure
    - Build duration

Step 11: Grow the Timeline

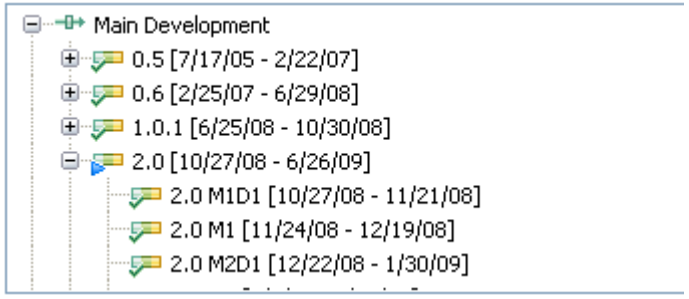
- As the project evolves the timeline needs to be grown
  - Add new iterations
  - Restructure iterations into nested iterations
  - Adjust the start and end dates
- Adjust the process rules and permission settings to fit the goals of the iteration
  - Use the appropriate scope for each rule or permission setting
    - This iteration; this iteration and all its siblings; all child iterations of a given iteration; all iterations in this timeline; all iterations in all timelines
  - Use iteration types for explicitly reusable process configurations
  - Use iteration types for all iterations, it makes targeted customization much simpler

Projects and Project Areas

- Project



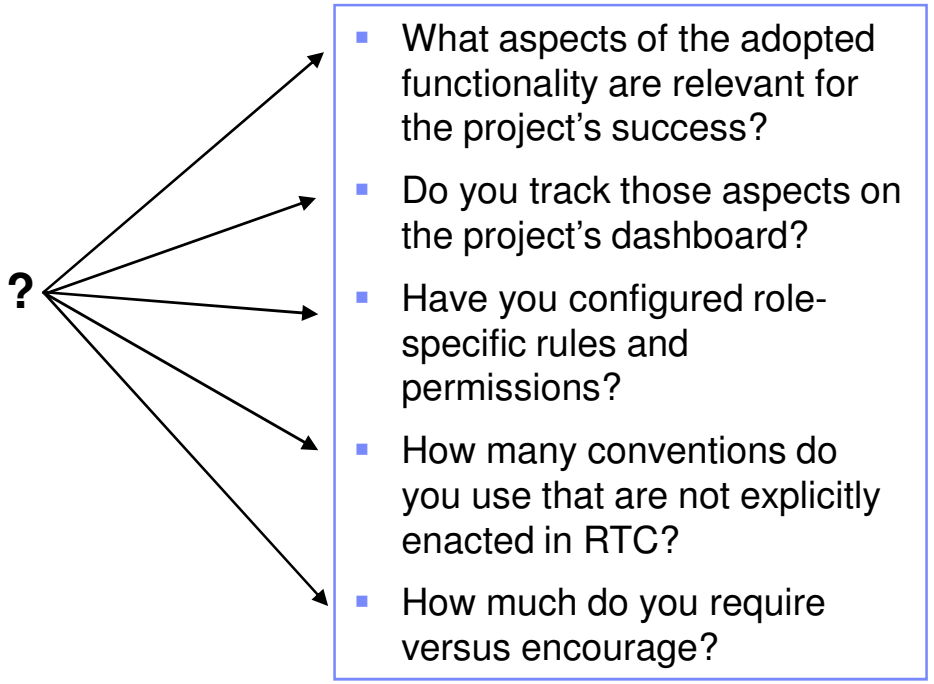
- Program



- Project areas can represent projects and programs
- Using project areas for programs allows you to reuse your setup
  - Team members, role assignments, team area hierarchy
  - Work item category to team area mappings
  - Dashboards, streams and builds
- In programs there is no real isolation between projects
- Create a **process template from** a **project area** to create a new independent project

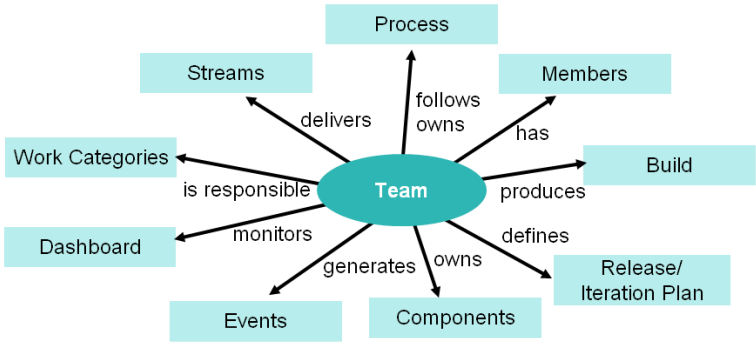
Step 12: Continuously Improve

- Dashboards & reports
- Work items and work item categories
- Component, streams, flows
- Build types, schedules, build tracking



Where Are We?

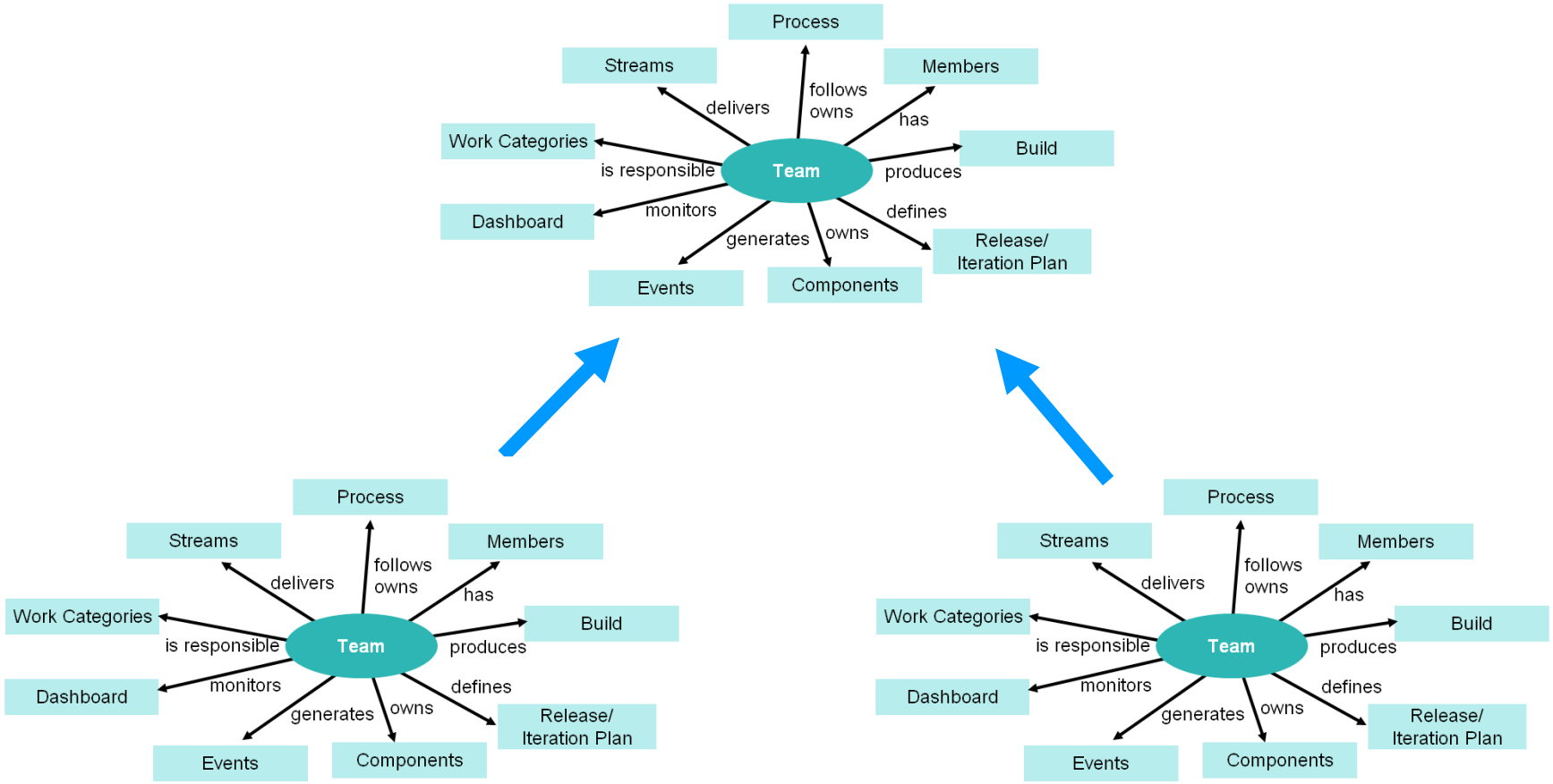
- We went through the full cycle
- One team adapted the full functionality
- Linear timeline



What Comes Next?

- Grow the “Island” by adding more teams
- Can interleave with adoption of new RTC functionality by previously added teams

Teams of Teams

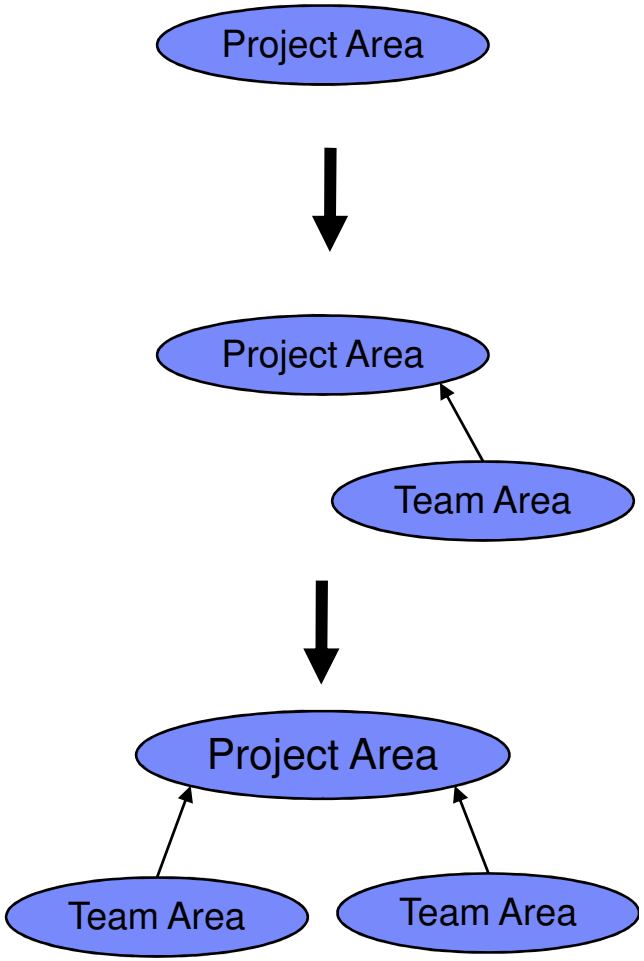


- Remember: We are talking about functional teams.

Step 13: Add the Second Team

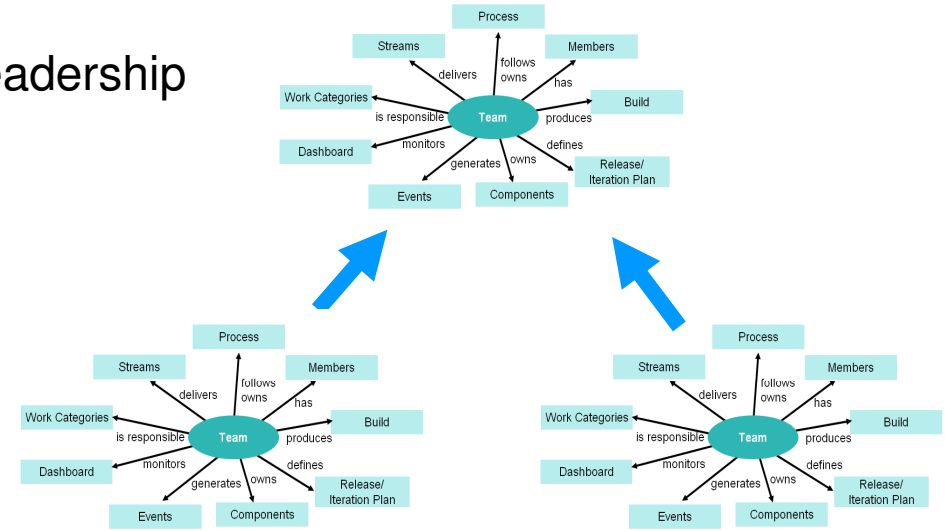
- Refactor organizational structure
  - Identify project leadership and the first team
  - Create a team area
  - Keep project leadership in the project area (including project release plans etc.)
  - Move team and its artifacts to the team area
    - Work item categories, team dashboards, streams, team builds
- Create a second team area
- Repeat steps 1 – 12 for the created team area

➢ Each team has its own sandbox



Step 14: Establish Collaboration Between the Teams

- Configure project area to reify project leadership and cross-team aspects
- Create a project dashboard
- Create stream(s) to integrate the code produced by the two teams
- Define the code integration process
  - Roles and responsibilities
- Establish automated integration build(s)
- Use project release plans, iteration plans, ...
- Create work item categories for cross-team aspects
  - Project operations such as process rules and permission changes
  - Project's Inbox
  - Integration build issues and release engineering ….



Step 15: Revisit Infrastructure

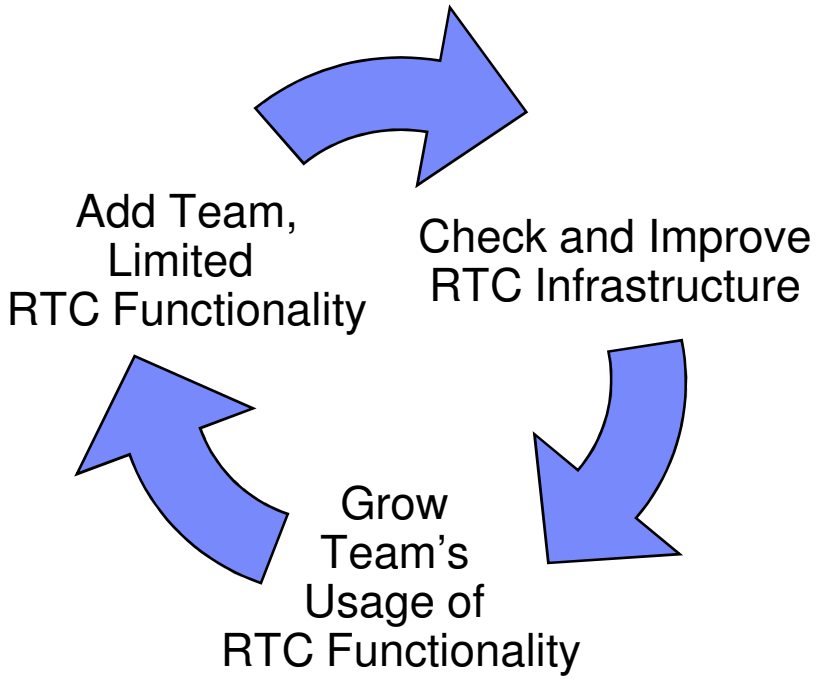
- Can you infrastructure handle additional growth?

- Change database or application server
- Use LDAP for user management

- Change the deployment
  - Separate DB from application server
  - Install proxy servers
  - High availability

- Backup and restore

Repeat the Cycle



Step 16: Parallel Development

- Example: Project Maintenance or Overlapping schedules
- Secondary timelines allow to run multiple parallel projects or programs within the same project area
  - Expresses strong relationship between these projects
  - Shared work item configuration and queries
  - Simplified work item triaging between the different projects (assign “Planned For”)
  - Shared iteration types
- For secondary timelines
  - create team areas for all functional teams working on the project
  - associate team areas with secondary timeline
  - Setup teams: step 1 - 12


Questions