Developer Collaboration Workflow Sequence

Based on my review of the user guide, here are the specific workflows for developer collaboration:

Developer Collaboration Workflow Sequence

Developer 1 (Initial Setup)

```
# 1. Create new project and set up repository
mkdir research-project
cd research-project

# 2. Initialize complete research compendium
zzrrtools --dotfiles ~/dotfiles

# 3. Set up version control and push to GitHub
git init
git add .
git commit -m "Initial zzrrtools setup"
git remote add origin https://github.com/[TEAM]/project.git
git push -u origin main

# 4. Start development work
make docker-rstudio              # → RStudio at http://localhost:8787

# 5. Add packages and do initial analysis
# (In RStudio container)
# install.packages("tidyverse")
# install.packages("lme4")
# renv::snapshot()

# 6. Quality assurance and commit
exit                             # Exit container
make docker-check-renv-fix       # Validate dependencies
make docker-test                 # Run package tests
make docker-render              # Test paper rendering

# 7. Commit changes with CI/CD trigger
git add .
git commit -m "Add initial analysis and dependencies"
git push                         # → Triggers GitHub Actions validation
```

Developer 2 (Joining Project)

```
# 1. Clone existing project
git clone https://github.com/[TEAM]/project.git
cd project

# 2. Set up environment (structure already exists)
make docker-build               # Build container with existing dependencies

# 3. Start development immediately
make docker-rstudio             # → Consistent environment with Dev 1

# 4. Sync with latest packages and add new work
# (In RStudio container)
# renv::restore()                 # Get Dev 1's packages
# install.packages("ggplot2")     # Add new package
# renv::snapshot()                # Update environment

# 5. Quality assurance workflow
exit                            # Exit container
make docker-check-renv-fix      # Update DESCRIPTION with new packages
make docker-test               # Ensure tests still pass

# 6. Commit with automated validation
git add .
git commit -m "Add visualization analysis with ggplot2"
git push                        # → GitHub Actions validates changes
```

Developer 1 (Continuing Work)

```
# 1. Sync with Developer 2's changes
git pull                        # Get latest code and renv.lock updates

# 2. Rebuild environment with new dependencies
make docker-build              # Rebuild container with Dev 2's packages

# 3. Validate environment consistency
make docker-check-renv-fix     # Ensure all dependencies are properly tracked

# 4. Continue development with updated environment
make docker-rstudio            # → Environment now includes Dev 2's packages

# 5. Add more analysis work
# (In RStudio container)
# renv::restore()                # Ensure all packages from Dev 2 are available
# Continue analysis with full package environment

# 6. Enhanced collaboration workflow
exit                           # Exit container

# 7. Use enhanced GitHub templates for pull request
git checkout -b feature/advanced-models
# Make changes...
git add .
git commit -m "Add multilevel models for nested data"
git push origin feature/advanced-models

# 8. Create pull request using enhanced template
# GitHub automatically provides:
# - Analysis impact assessment checklist
# - Reproducibility validation
# - Automated CI/CD checks
# - Paper rendering validation
```

Key Collaboration Features (rrtools_plus Integration)

Automated Quality Assurance on Every Push:

- **R Package Validation**: R CMD check with dependency validation
- **Paper Rendering**: Automated PDF generation and artifact upload
- **Multi-platform Testing**: Ensures compatibility across environments
- **Dependency Sync**: renv validation and DESCRIPTION file updates

Enhanced GitHub Templates:

- **Pull Request Template**: Analysis impact assessment, reproducibility checklist
- **Issue Templates**: Bug reports with environment details, feature requests with research use cases
- **Collaboration Guidelines**: Research-specific workflow standards

Seamless Environment Synchronization:

```
# Any developer can sync at any time:
git pull                       # Get latest changes
make docker-build             # Rebuild with updated dependencies
make docker-rstudio           # → Identical environment across team
```

Data Management Collaboration:

```
# Structured data workflow for teams:
data/
```

```
  raw_data/                 # Dev 1 adds original datasets
  derived_data/             # Dev 2 adds processed data
  metadata/                 # Both document data sources
  validation/               # Automated quality reports
```

This workflow ensures **perfect reproducibility** across team members while providing **automated quality assurance** and **professional collaboration tools** integrated from the rrtools_plus enhancement framework.