**Extreme Roles in XP — Teaching Notes & How-To**

**1) Why Roles in XP?**

XP thrives on **short feedback loops** and **working software**. Clear roles:

* Define **who** owns **which decisions and artifacts** (value, implementation, quality signals, delivery health).
* Create **regular touchpoints** (stand-ups, planning, CRC, acceptance) that trigger action.
* Surface problems **early** so they can be fixed **fast**.

**2) Role Overview (What they do • When they do it • What they produce)**

**A. Customer**

* **Does**:  
  Writes **User Stories**; specifies **Acceptance/Functional Tests**; **prioritizes**; explains stories; may observe CRC; **decides** story questions.
* **When**: Planning game, iteration planning, and acceptance; available for daily clarifications.
* **Produces**: Story cards, acceptance criteria/tests, priority & release decisions.
* **Notes**: Can be Planner/Analyst/Product Lead/PM/Designer; not necessarily an end-user, but **must have authority**.

**B. Programmer**

* **Does**:  
  **Estimates** stories; derives and estimates **Engineering Tasks**; **implements** stories; writes/maintains **Unit Tests**.
* **When**: Daily pairing/coding, continuous integration, ongoing sync with Tester/Customer.
* **Produces**: Tested code, unit tests, scripts/docs, merge records.

**C. Tester (Functional/Acceptance)**

* **Does**:  
  **Implements and runs** functional/acceptance tests; **graphs** results; **alerts** the team when quality trends down.
* **When**: In step with the iteration; after each integration/run.
* **Produces**: Automated acceptance tests, test reports/trend charts, defect/blocked signals.
* **Note**: **Programmers own unit tests**; Testers focus on **end-to-end acceptance**.

**D. Tracker**

* **Does**:  
  **Once or twice a week**, walks around, asks each programmer about progress/impediments/risks; **listens**; triggers corrective actions when off-track (e.g., schedule a **CRC** session, set a Customer sync, ask the Coach or another programmer to help).
* **When**: Weekly/bi-weekly, and around milestones.
* **Produces**: **Risk/variance notes**, follow-up actions, brief status for management/retro.
* **Style**: **Non-threatening**, open questions to encourage honest signals.

**E. Coach**

* **Does**:  
  **Observes the whole system**, keeps the team **“Stay Extreme”**; flags process smells and **nudges/corrects**; helps anywhere needed; applies the “**rolled-up newspaper**” when necessary (clear but kind reminders).
* **When**: All the time—stand-ups, planning, pairing, CI, reviews, retros.
* **Produces**: Process improvements, pairing/review suggestions, practice alignment.

**F. Manager**

* **Does**:  
  **Schedules** meetings (Iteration Plan, Commitment Schedule); ensures process is followed; **records** outcomes for reporting and **passes to the Tracker**; handles **external coordination & logistics** (info, resources, pizza, shelter, HR forms).
* **Does *not***: Decide **what** to build (Customer + plan do), dictate **when** it must be done (Commitment Schedule), or **check progress** (Tracker).
* **Produces**: Agendas/minutes, outward comms, resources/logistics.

**G. Doomsayer (anyone can play)**

* **Does**:  
  **Calls out big trouble**—current or upcoming—so the team faces risk early.
* **Practice**: Speaking fear is **encouraged**; it’s a form of **courage**. Tester and Doomsayer often combine well (quality + risk vantage point).
* **Not** an anti-role; it’s permission to **tell the truth**.

**3) How Roles Collaborate (key activities)**

| **Activity** | **Leads / Decides** | **Key Participants** | **Main Outputs** |
| --- | --- | --- | --- |
| Planning Game / Iteration Planning | **Customer** (value & priority) | Programmer (estimates), Tester (testability), Coach (process), Manager (facilitation) | Selected stories & goals, initial task breakdown |
| CRC Session | **Coach** facilitates; Customer clarifies | Programmers, Tester | Candidate classes/resp/collabs → inputs to tasks |
| Daily Stand-up | **Coach/Manager** keep cadence | Whole team; Tracker logs smells | Risk/impediment list & follow-ups |
| Dev & Integration | **Programmers** | Tester (acceptance runs), Coach (discipline), Tracker (variance) | Tested increments |
| Acceptance Demo | **Customer** (done/not-done) | Tester (visualizes results), Programmers (Q&A) | Acceptance decisions & adjustments |
| Retro / Improvement | **Coach** facilitates | Whole team | Concrete improvement actions |

**4) Role Combinations: When OK vs. When Risky**

**Reasonable combinations**

* **Manager + Tracker**: Often workable—**preserve the Tracker’s non-threatening posture**.
* **Tester + Doomsayer**: Natural synergy.
* **Anyone + Doomsayer**: Signaling risk is everyone’s right and duty.

**Generally avoid combining**

* **Programmer + Tester**: Self-testing at acceptance level creates **blind spots**.
* **Programmer + Tracker**: Self-reporting **hides reality**; bias in variance detection.
* **Customer + Programmer**: Value vs. implementation **conflates**; “HOW” crowds out “WHY/WHAT”.
* **Coach + Tracker**: Coach sometimes **applies pressure**, Tracker must be **safe to confide in**—motives can conflict.
* **Coach + Programmer** (experience-based): Coaching is **time-intensive** and global; owning features in parallel dilutes both.

These are **tendencies**, not laws. If you must combine, **explicitly list the conflicts and compensating controls** (e.g., non-manager runs Tracker interviews; Customer always makes acceptance calls).

**5) Very Small Teams (classroom/project adaptations)**

**4–5 people (recommended minimum)**

* **2× Programmers** (rotate pairs)
* **1× Customer** (product/TA proxy)
* **1× Tester** (can also act Doomsayer)
* **1× Manager** (can also act Tracker)
* **Coach** ideally independent; if unavailable, appoint a **process guardian** who does **not** own major coding.

**3 people (bare-bones, risky)**

* **2× Programmers** (strict pairing; cross-unit-test & reviews)
* **1× Customer** (also does basic testing—**high risk**; mitigate with an **acceptance test harness** and external check-ins)
* Split Manager/Tracker/Coach duties with **lightweight cadence** (weekly board/agenda/checklist) and invite an outside mentor for periodic audits.

**Risk note**: The smaller the team, the easier role boundaries blur. Keep **decision power** (Customer) and **verification power** (Tester/acceptance) as **separate** as possible. Make test results **visible** to all.

**6) Practical Checklists (to make roles “real”)**

* **Role cards on the team board**: for each role, list 3 responsibilities, 3 artifacts, weekly cadence, one decision right.
* **Tracker two passes/week**: three questions—What are you doing? What’s blocking you? Who do you need? Log actions immediately.
* **Coach “rolled-up newspaper” moments**: intervene on any of—missing unit tests, persistently red CI, no Customer acceptance, consecutive overtime weeks.
* **Tester visualization**: chart pass rate/trends in a visible spot; **announce red** in stand-ups.
* **Doomsayer microphone**: team working agreement explicitly allows **anyone** to raise yellow/red cards at any time.
* **Manager moat**: shield from outside churn, secure resources/logistics, record meeting outcomes and route to Tracker/Coach.

**7) Extreme Anti-Roles (watch-outs)**

If you encounter people who **undermine collaboration, hide problems, or sabotage practices**, the guidance is blunt: **remove them**. If you can’t, **protect yourself and expect pain**.