APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

How to use this appendix

The book refers to some ideas and concepts from Transactional Analysis that have been used in the book, and it would be helpful to provide an extended description in case you feel lost or confused.

Ego States

In plain words: Think of ego states as the "voices" or modes we all have. In TA, each of us can operate from a Parent, Adult, or Child state. The *Parent* voice mimics what we learned from authority figures (it can be critical or caring). The *Child* voice is our emotional, reactive side (playful, curious, or defensive). The *Adult* voice stays present and objective, focused on facts and solutions. We all shift between these states – sometimes even within the same meeting!

Why it matters in Agile teams:

- Adult-to-Adult communication = better collaboration & decisions.
- Under stress, people slip into Parent/Child modes that derail conversations.
- Recognising ego states can de-dramatize conflict and build empathy – you see a "harsh" comment as someone's Parent or Child mode, not a personal attack.
- Making *Adult* tone the norm supports respectful dialogue and clear problem-solving (fostering self-organisation).

How it shows up:

- A developer in a code review barks, "This is completely wrong – who wrote this?" (Critical Parent).
- Team members wait for a manager to direct them instead of taking initiative (Child-like dependence).
- Someone blurts "This isn't fair!" or throws up their hands in a retro (Child state).
- A team lead gives advice in a warm, "I've been there" way (Nurturing Parent).
- Two colleagues calmly examine a bug together, asking "What are our options?" (Adult-to-Adult problem-solving).

One-liner metaphor: *It's like having three radio channels – Parent, Adult, Child. Tune into the Adult station for the clearest team communication.*

Life Positions

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

In plain words: *Life positions* are basic attitudes about self and others. The healthiest is "I'm OK, You're OK," meaning you respect yourself and others as capable and worthy. Other positions include "I'm OK, You're not OK" (seeing others as the problem), "I'm not OK, You're OK" (seeing yourself as the problem), and "I'm not OK, You're not OK" (a hopeless view of everyone). These mindsets can colour how team members approach collaboration, especially under stress.

Why it matters in Agile teams:

- Trust & safety baseline: "I'm OK, You're OK" fosters mutual trust and psychological safety – crucial for collaboration and self-organization.
- Low-trust signals: The other positions signal trouble. Constant blame (*I'm OK, you're not OK*) poisons team cohesion; constant self-doubt (*I'm not OK*) silences voices.
- Confidence and voice: If someone feels not OK, they may hold back ideas or defer to others, and the team loses valuable input.

How it shows up:

- A team member often says, "This team never gets it right" or "I'm surrounded by idiots!" – a clear *I'm OK, you're not OK* stance.
- Another frequently downplays their own ideas: "This might be stupid, but…" – a sign of *I'm not OK, you're OK*.
- The team avoids bringing up issues because "it won't matter" – drifting into *I'm not OK, you're not OK* pessimism.
- In a pairing session, both people listen and build on each other's ideas – an *OK-OK* interaction.
- After a mistake, the team says "We're all human. Let's learn and fix it," instead of finger-pointing – *OK-OK* under pressure.

One-liner metaphor: *Life positions are like glasses – "I'm OK, You're OK" is the clear lens. Other positions are like distorted lenses that skew how we see each other.*

Scripts and Injunctions

In plain words: A *script* is like an invisible life plan or set of "rules" we unconsciously follow, often formed in childhood. *Injunctions* are the big "Don't" or "Do" messages that shape those scripts (e.g., "Don't make mistakes," "Always be perfect"). In team life, these show up as personal unwritten rules that can limit behaviour. Think of a script as an old story you're acting out even when it doesn't fit the current situation.

Why it matters in Agile teams:

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

- Unspoken team rules: Sometimes a whole team operates under a hidden mandate (e.g., “Don’t question the boss” or “We must appear busy at all times”). Identifying these can prevent groupthink and burnout.
- Hidden personal blockers: Individual scripts like “Don’t ask for help” or “I must be perfect” stop team members from collaborating, raising risks, or admitting issues.
- Adapting norms: Surfacing and questioning these assumptions (“Why do we always do it this way?”) lets the team update habits to fit reality.
- Understanding resistance: If someone resists a change, they might be following an old rule (“Don’t trust anyone,” “Don’t fail”). Knowing this, you can coach with empathy instead of frustration.

How it shows up:

- Despite an “open door” culture, one engineer never asks for help – likely their inner rule is “Don’t be needy.”
- The team consistently avoids conflict in retros – as if everyone shares “Don’t make waves.”
- A high performer refuses to delegate tasks, driven by “If you want it done right, do it yourself” (a perfectionist script).
- Team members downplay or ignore praise – perhaps they carry a “Don’t brag” or “Don’t stand out” rule.
- A new hire seeks approval for every step, echoing “Don’t do anything without permission.”

One-liner metaphor: *Scripts are like old software running in the background – sometimes you need to update the code to work better today.*

Symbiosis

In plain words: In TA, *symbiosis* means two people functioning as if they’re one. One person does the thinking or deciding (taking the “Parent” or “Adult” role for both), and the other person just goes along (sticking to the “Child” role). It can feel comfortable because each avoids what the other handles – but it stunts growth. In a team, symbiosis appears as over-helping or over-relying: one person takes on responsibilities that others should share.

Why it matters in Agile teams:

- Collective ownership: Agile thrives on shared responsibility. If one leader always decides and the team just follows, true self-organisation and learning can’t happen.
- Single point of failure: Over-reliance on one key member is risky – if they’re absent, everything stalls.

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

- Burnout & stagnation: The over-functioning person can burn out, while others under-function and don't develop their skills.
- Balanced team = growth: Breaking a symbiotic pattern creates a more balanced Adult–Adult dynamic. Dependent members gain confidence, and the overloaded leader can step back, making the team more resilient and creative.

How it shows up:

- The Product Owner writes every detail and calls every shot, while the team quietly implements – a PO/team symbiosis.
- A senior dev "saves" the team on every tough task, so others stop trying and just wait for rescue.
- In meetings, one voice (maybe a manager or tech lead) speaks for everyone, and the rest just nod along.
- The Scrum Master ends up tracking all tasks and solving all impediments for the team – effectively doing their thinking for them.

One-liner metaphor: *Symbiosis in a team is like a three-legged race – two people tied together can move, but not as freely as when everyone runs on their own two feet.*

Positive Strokes

In plain words: A "*stroke*" in TA means a unit of recognition – basically, any act of acknowledging someone. Positive strokes are the warm fuzzies: thank-yous, praise, a high-five, a thumbs-up. Negative strokes (criticism, scolding) count as recognition too (they're attention), but they hurt. Everyone needs strokes like plants need sunlight – we all crave to feel seen and appreciated. On an Agile team, giving genuine positive strokes fuels morale and connection.

Why it matters in Agile teams:

- Motivation & habits: When people feel valued, they're motivated to do their best and to repeat the helpful behaviours that earned praise.
- Trust and safety: A team that shares appreciation freely tends to have higher psychological safety. People know their efforts are noticed (not just their mistakes).
- Prevent "stroke hunger": If positive strokes are rare, team members might even seek negative attention (criticism) just to feel acknowledged. Keeping a healthy flow of praise prevents that and keeps focus on constructive behaviour.

How it shows up:

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

- Teammates regularly say “Nice job on X!” or give shout-outs in retrospectives – a culture rich in positive strokes.
- A team member who only hears from others when something’s wrong may start to disengage or get defensive (sign of stroke deprivation).
- Silence around good work (“no news is good news”) – an environment where recognition is scarce.
- One person gets all the kudos while others’ contributions are ignored – an imbalance in strokes that can breed resentment.
- The team uses a “kudos” Slack channel or does fun awards each sprint – signs that positive recognition is a norm.

One-liner metaphor: *Positive strokes are the team’s emotional fuel – keep the tank filled and the engine will run smoothly.*

Drama Triangle

In plain words: The Drama Triangle describes three roles people slip into during conflict: Victim, Rescuer, and Persecutor. In *Victim* mode, a person feels powerless (“Poor me”). The *Rescuer* jumps in to save others (whether or not anyone asked), and the *Persecutor* blames or attacks others. It’s like a bad drama where everyone has a role, and they often rotate roles without resolving the actual problem. The only way out is for someone to “step off” the triangle by moving to honest, Adult behaviour – for example, by taking responsibility (instead of Victim), setting boundaries or making a clear request (instead of playing Rescuer or Persecutor).

Why it matters in Agile teams:

- Wasted energy: When a team gets caught in Victim/Rescuer/Persecutor dynamics, they spend more time in personal drama than solving the issue at hand.
- Erodes trust: These roles create one-up/one-down power dynamics (Persecutor over Victim, Rescuer over Victim). That undermines the equal, trusting environment that agile teams need.
- Accountability lost: In the triangle, Victims deny their power, Persecutors deny their faults, Rescuers overstep boundaries. By recognising the pattern, a coach can help the team regain personal accountability and a solution focus.
- Faster conflict resolution: The quicker someone (coach or team member) sees the drama pattern and shifts to Adult-to-Adult communication, the quicker the team can defuse conflict and find a way forward.

How it shows up:

- Victim stance: “This isn’t my fault, there’s nothing I can do!” or a team member who frequently looks defeated and withdraws at signs of trouble.

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

- Rescuer behaviour: One person always tries to “fix” others’ problems or protect someone from consequences: “Don’t worry, I’ll do your tasks too.” They mean well but may foster dependency (and burn themselves out).
- Persecutor actions: Harsh blame or criticism: “Who screwed this up?!” or a cutting, sarcastic comment that makes someone feel attacked.
- Recurring patterns: Often the same drama plays out whenever there’s a crisis – and people might even switch roles midstream. (Yesterday’s Victim might become today’s finger-pointing Persecutor.) The telltale sign is a lot of blaming or martyrdom, and not much progress on the actual problem.

One-liner metaphor: *The Drama Triangle is like a game of hot potato – blame and helplessness get passed around until someone puts it down and addresses the real issue.*

Games

In plain words: In TA, a *game* is a recurring, negative interaction pattern with a hidden payoff. It’s not fun – it’s more like a trap. A classic example is the “Yes, but…” game: one person asks for ideas, others offer help, but the person deflects every solution (“Yes, but that won’t work because…”). The hidden payoff might be attention or avoiding change. Games often leave people feeling frustrated or guilty, and nothing really gets solved – yet the pattern repeats until someone breaks it.

Why it matters in Agile teams:

- Wasted time and frustration: Games suck up time and energy with no progress, leaving everyone cynical (“Here we go again…”). Morale drops when the same unproductive discussion repeats.
- Hidden problems stay hidden: Games persist because an underlying issue isn’t being addressed openly (e.g., the “Yes, but” person might actually fear the proposed changes). Until the pattern is broken, the real concern remains unresolved.
- Transparency and improvement: Agile teams value openness. Spotting a game and naming it (tactfully) brings the real issue to light, so the team can tackle it and improve their process.
- Better teamwork: Breaking out of a game replaces finger-pointing or wheel-spinning with honest communication and action – moving the team forward and restoring trust.

How it shows up:

- “Yes, but…” loop: In a retro, Pat says, “We need to improve X.” Team suggests ideas; Pat replies “Yes, but…” to everyone. The discussion circles until everyone gives up.

APPENDIX A – TA CHEETSHEET FOR AGILE COACHES

- Blame ping-pong: Two teams or colleagues keep passing blame: “We’re late because *they* didn’t clarify requirements” – “Well, we’re late because *they* developed too slowly.” The argument repeats every project with no change in process.
- One-upmanship: Team meetings turn into a contest of who has the bigger problem: “You think that’s bad? Wait till you hear this…” It bonds people in commiseration, but no solution comes – a game of sharing woes instead of solving them.
- Déjà vu conversations: The team groans, “We’re stuck in this conversation again.” Everyone senses they’ve been through this exact debate or conflict multiple times (and they have).

One-liner metaphor: *Games are like a team Groundhog Day – the same bad scene keeps replaying until someone says, “Let’s try something different this time.”*