Kent Beck, Cynthia Andres (2004) *extreme programming explained - embrace change*. 2nd edition. p 66

Shared Code

Anyone on the team can improve any part of the system at any time. If something is wrong with the system and fixing it is it out of scope for what I'm doing right now, I should go ahead and fix it.

One objection I've heard is that if one person is responsible for a piece of code, then everyone will act irresponsibly. They will make expedient changes, leaving a mess for the next person who has to touch the code. The risk of this happening is why I've listed Shared Code as a corollary practice. Until the team has developed a sense of collective responsibility, no one is responsible and quality will deteriorate. People will make changes without regard for the team-wide consequences.

There are other models of teamwork besides "every man for him-self." The team members can collectively assume responsibility not just demonstrate their commitment to quality to each other and helps them normalize their expectations for what constitutes quality.

Continuous integration is another important prerequisite for collective ownership. A two-hour programming session can touch many parts of the system if there are many opportunities for improvement. Two pairs making many, widespread changes increase the chance of expensive-to-resolve incompatible changes. If the team is making lots of changes, it may want to reduce the interval between integrations to keep the cost of integration down.