Agile Evolution to Legacy Software

The older the code base and the more mission critical the application, the more difficult it is to maintain or enhance it. One primary difficulty is just understanding the code base which may not be well-designed or well-written. For this reason, the author promotes the idea that when working with legacy software, Discovery and Transformation should be the focus over Design and Development. Discovery combines the stories obtained from experienced developers and customers with knowledge gained by analyzing the code and associated documentation and test cases. Transformations of the code should be done in systematically in small pieces, testing each change before moving to the next.

The author also describes some agile methods that prove useful in software maintenance:

- Methaphors/ use cases/ cards are all useful in conveying the big picture of the system and having that big picture, the essence of the code, live on through generations of developers. A story is easier to remember than documentation.
- Problem tracking systems help prioritize what parts of the system to update
- Pair development makes maintaining legacy code much easier and reliable especially if the code is unfamiliar and/or mission critical
- Continuous integration allows for more frequent regression testing, which is a key factor in effectively maintaining a legacy system (or any system for that matter)
- Of course refactoring is necessary to remove defects and replace complex pieces with code that is more understandable