updraft

AIM Methodology

Analyze your applications

Describe and inventory your code

- Document application components and code size
- Measure code complexity and analyze identifiers
- Uncover replicated, dead or unreachable code

Symbolically interpret code

- Map source code relationships and interdependencies
- Visualize data flow, code flow and data lineage
- Analyze dead code, external databases accessed and other called programs

Understand what your code does

- Extract and analyze business rules
- Analyze features

Improve your applications

Enhance code quality and maintainability

- Reduce complexity: remove dead code, reduce replication and remove GOTO statements
- Simplify naming of identifiers and methods
- Conform to coding standards

Increase performance

- Systematically improve algorithms
- Replace a file used for temporary storage with an in-memory database

Expand scalability

- Refactor for a service-oriented architecture using microservices
- Refactor for data microservices

Migrate to alternative platforms

- **Identify a target language** — translate COBOL to Java, for example
- **Determine an execution environment** — move from IBM mainframe with CICS to Linux with JBoss
- **Database** — adapt code to use PostgreSQL versus DB2
- **User interface** — move from CICS with BMS to Angular
- **Cloud** — refactor to operate in a cloud framework; dockerize or migrate to AWS/Azure
- **Optimize file handling** — move from flat files to a database and change code to handle ASCII code instead of EBCDIC, for example