1 - Complex Environment

Parallel and heterogenous hardware
Compilers must be multi-target, collaborate, and be specialized.

2 - Source-to-Source

Many source-to-source successful compilers. Flexible transformation systems for heterogeneous computing:
- parallelism detection algorithm,
- variable privatization,
- communication generation,
- etc.

3 - Model for Code Transformations

Sequencing of code transformations is the compiler core.
Formal point of view for interactions between passes: several transformation composition rules.

4 - Based on a Scripting Language

on the shoulders of giants
No DSL, does not reinvent the wheel, build over a high level language with a rich ecosystem which widens the set of possibilities.

5 - Abstractions

Program, File, Function, Loop

Classes: high level program representation
Methods: compose transformations into more complex ones
Inheritance: compose scheme for heterogeneous targets

6 - Control Structures

Conditionals: manage switches, choose different compilation schemes.
For loops: iterate over the callgraph, loop nests.
Exceptions: recover from compilation failure, impossible transformations, etc.
While loops: look for fixed point for sequence of passes.

Related Work