

Thesis – Fortran 77 Ongeschikt voor Moderne Software-Architecturen

Thesis - Fortran 77 Ongeschikt voor Moderne Software-Architecturen

en waarom Fortran 95 de minimale toekomstvaste standaard vormt

Abstract

Deze thesis onderzoekt de geschiktheid van Fortran 77 binnen moderne software-architecturen. Hoewel Fortran 77 historisch een cruciale rol speelde in wetenschappelijke en numerieke toepassingen, voldoet de taal niet langer aan de eisen van hedendaagse software-ontwikkeling: modulariteit, type-veiligheid, interoperabiliteit, geheugenbeheer en onderhoudbaarheid. Fortran 95 wordt gepresenteerd als de minimale moderne standaard die dezelfde performance en eenvoud behoudt, maar wél de noodzakelijke taalconstructies biedt voor veilige, schaalbare en interoperabele systemen.

1. Inleiding

Fortran 77 werd in 1978 gestandaardiseerd en domineerde decennialang de numerieke softwarewereld. De taal is echter ontworpen voor een tijdperk waarin software monolithisch was, geheugen schaars was en interoperabiliteit geen vereiste vormde. In 2026 zijn deze aannames volledig achterhaald. Moderne software-architecturen vereisen modulaire componenten, veilige interfaces, dynamisch geheugenbeheer, interoperabiliteit met C, C++, COBOL, Python en DLL-ecosystemen, en onderhoudbaarheid door teams. Deze thesis toont aan dat Fortran 77 structureel tekortschiet en dat Fortran 95 de logische, minimale opvolger is.

2. Historische Context

Fortran 77 introduceerde verbeteringen t.o.v. Fortran 66, zoals block IF, CHARACTER-types en PARAMETER-statements. Toch bleef het taalontwerp beperkt door fixed-format code, COMMON blocks, het ontbreken van modules, geen dynamische arrays en geen expliciete interface-controle. Deze beperkingen waren acceptabel in 1978, maar zijn onhoudbaar in moderne software-ecosystemen.

3. Technische Analyse van Fortran 77-Beperkingen

3.1 Gebrek aan Modulariteit

Fortran 77 kent geen modules. Alle gedeelde data moet via COMMON blocks of INCLUDE-files worden beheerd. Dit leidt tot naamconflicten, fragiele afhankelijkheden en moeilijk onderhoud.

3.2 Geen Type-veiligheid

Fortran 77 heeft geen intent-specificatie, geen interface-controle en impliciete typing (I-N integer, rest real). Dit veroorzaakt stille fouten die pas tijdens runtime zichtbaar worden.

3.3 Geen Dynamisch Geheugen

Alle arrays zijn statisch, compile-time bepaald en niet uitbreidbaar. Moderne workloads vereisen dynamische datastructuren.

3.4 Geen Interoperabiliteit

Fortran 77 heeft geen bind(C), geen iso_c_binding en geen voorspelbare symbol-namen. Daardoor is F77 onbruikbaar voor DLL-exports, C-ABI, COBOL-interfacing en moderne toolchains.

3.5 Geen Structs of Derived Types

Struct-achtige data is onmogelijk zonder hacks, wat interoperabiliteit met C-structs onmogelijk maakt.

4. Waarom Fortran 95 de Minimale Moderne Standaard is

4.1 Modules

Fortran 95 introduceert namespaces, interface-controle, expliciete typing en veilige afhankelijkheden.

4.2 Intent-specificatie

Intent-specificatie voorkomt een groot deel van klassieke Fortran-bugs.

4.3 Allocatable Arrays

Dynamisch, veilig en automatisch beheerd.

4.4 Volledige C-Interoperabiliteit

Via `iso_c_binding` en `bind(C)` biedt Fortran 95 stabiele symbol-namen, voorspelbare ABI en directe mapping naar C, COBOL en ML64.

4.5 Performance Identiek aan Fortran 77

Fortran 95 introduceert geen runtime-overhead. De performance is gelijkwaardig, maar de veiligheid is veel hoger.

5. Vergelijking Fortran 77 vs. Fortran 95

Eigenschap	Fortran 77	Fortran 95	Conclusie
Modulariteit	Geen	Modules	F95 wint
Type-veiligheid	Geen	intent + interfaces	F95 wint
Dynamische arrays	Geen	Allocatable	F95 wint
Interoperabiliteit	Geen	C-ABI	F95 wint
Structs	Geen	Derived types	F95 wint
Performance	Hoog	Hoog	Gelijk
Onderhoudbaarheid	Slecht	Goed	F95 wint

6. Interoperabiliteit in Moderne Architecturen

Moderne systemen bestaan uit C-kernels, Fortran-numerieke routines, COBOL-front-ends, DLL-architecturen en ML64-fastpaths. Fortran 77 kan hier niet in functioneren. Fortran 95 voldoet aan alle eisen zonder de complexiteit van latere standaarden.

7. Case Study: DLL-Architectuur (COBOL + C + Fortran + ML64)

In een moderne DLL-architectuur definieert C de ABI, levert Fortran de numerieke kernels, roept COBOL de DLL aan en levert ML64 fastpaths. Fortran 77 faalt op symbol-stabiliteit, pointer-passing, struct-mapping, C-ABI en error-handling. Fortran 95 voldoet aan alle eisen.

8. Conclusie

De stelling is technisch correct:

Fortran 77 is ongeschikt voor moderne software-architecturen. Fortran 95 biedt dezelfde

performance en eenvoud, maar met de noodzakelijke taalconstructies voor veiligheid, modulariteit en interoperabiliteit.

Fortran 95 is de minimale moderne standaard voor numerieke software die moet integreren met C, COBOL, DLL's en assembly.

9. Aanbevelingen

1. Nieuwe projecten nooit in Fortran 77 starten.
2. Legacy F77-code migreren naar Fortran 95-modules.
3. Alle externe interfaces via C-ABI (bind(C)).
4. Gebruik allocatable arrays intern, maar niet in de ABI.
5. Gebruik ML64-fastpaths voor performance-kritische kernels.