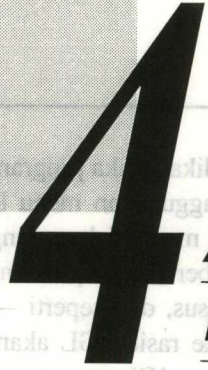


Alasan utama dari pemindahan ke bahasa generasi keempat adalah untuk meningkatkan/mempertahankan produktivitas pembuatan aplikasi. Suatu pengalaman dari tujuan ini adalah sangat bervariasi. Variasi ini ada pada type system yang dibuat, dengan individual dimasukkan dan teknik management proyek digunakan. Dalam beberapa hal pemindahan menggunakan 4GL telah memberikan sedikit atau tidak ada perbaikan/peningkatan produktivitas karena teknik management yang kurang sistematisnya diterapkan atau karena pemakaian 4GL telah membuat kotor oleh karena (lack) metodologi design.

Sangat umum menemukan program 4GL untuk system kompleks yang moderat dengan mempunyai seperti atau seperseduh jumlah baris code yang sama dengan yang dimiliki program COBOL. Rata-rata program COBOL menuliskan sekitar 30 baris code per hari (yaitu, jumlah total baris COBOL dalam system akhir yang dibagi dengan jumlah total program per hari secara tipikal, kurang lebih 30, termasuk debugging dan penulisan kembali). Program 4GL secara tipikal melakukan hal ini lebih baik, sering mencapai 40 dan juga umum mencapai 100 baris code ketika diukur dengan cara yang sama. Hal ini sering tidak berarti menggunakan pengurangan yang sama, namun demikian, karena 4GL memang menggunakan teknik semacam pengisian dalam panel dalam

COBOL-KE-RASIO 4GL



**AKIBAT DARI 4GL
PADA PRODUKTIFITAS DP**

COBOL-ke-rasio 4GL bervariasi menurut alam aplikasi. Cobol pada pokoknya adalah pembuatan laporan, menggg...
layar data...
baik diband...
kecil. Penulisan program untuk routine pemantuan caru, 4GL mempunyai sedikit akibat pada produktivitas; dengan menggunakan editor diagram aksi (Bar-08) dengan bahasa yang terstruktur dengan baik seperti C yang melakukan "produktivitas perbaikan.

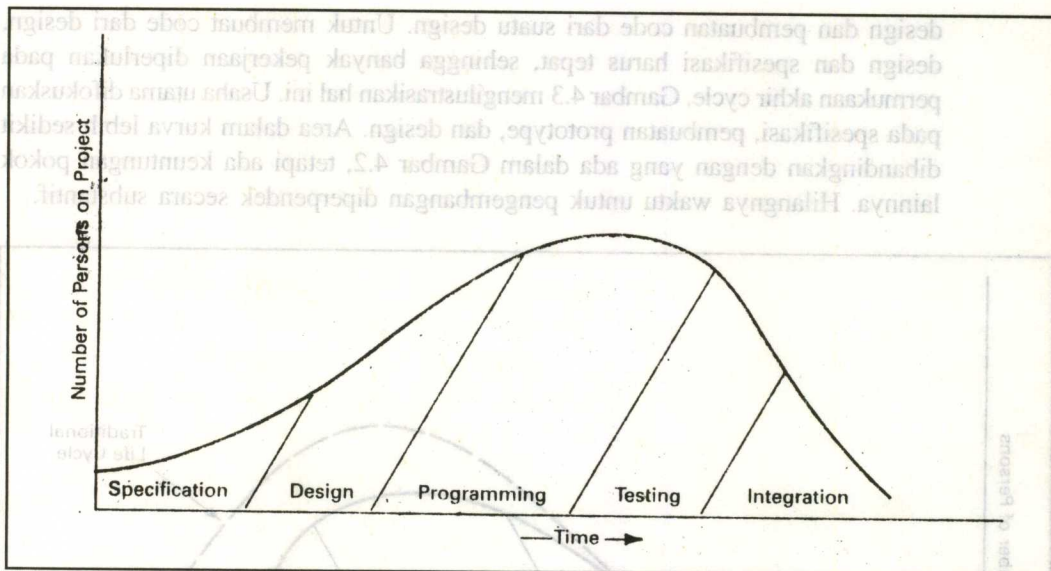
Pengenalan

Alasan utama dari pemindahan ke bahasa generasi keempat adalah untuk meningkatkan/memperbaiki produktifitas pembuatan aplikasi. Suatu pengalaman dari tujuan ini adalah sangat bervariasi. Variasi ini ada pada type system yang dibuat, dengan individual dimasukkan dan teknik management proyek digunakan. Dalam beberapa hal perpindahan menggunakan 4GL telah memberikan sedikit atau tidak ada perbaikan/peningkatan produktifitas karena teknik management yang kurang semestinya diterapkan atau karena pemakai 4GL telah membuat kotor oleh karena (lack) metodologi design.

Sangat umum menemukan program 4GL untuk system kompleks yang moderat dengan mempunyai seperlima atau sepesepuluh jumlah baris code yang sama dengan yang dimiliki program COBOL. Rata-rata programmer COBOL menuliskan sekitar 20 baris code per hari (yaitu, jumlah total baris COBOL dalam system akhir yang dibagi dengan jumlah total programmer per hari secara tipikal, kurang lebih 20, termasuk debugging dan penulisan kembali). Programmer 4GL secara tipikal melakukan hal ini lebih baik, sering mencapai 40 dan juga umum mencapai 100 bari code ketika diukur dengan cara yang sama. Hal ini sering tidak praktis menggunakan pengukuran yang sama, namun demikian, karena 4GL mungkin menggunakan teknik semacam pengisian dalam panel dalam layar.

COBOL-KE-RASIO 4GL

COBOL-ke-rasio 4GL bervariasi menurut alam aplikasi. Jika program aplikasi berisi pada pokoknya adalah pembuatan laporan, menggunakan menu layar atau layar data-entry, dan accessing basis data, 4GL akan mengerjakan dengan lebih baik dibandingkan dengan COBOL. Jika suatu aplikasi berisi pada pokoknya adalah logika — routine yang disarangkan, loop, struktur kasus, dan seperti — laporan singkat, layar, dan penggunaan basis data, COBOL-ke rasio 4GL akan menjadi kecil. Penulisan program untuk routine permainan catur, 4GL mempunyai sedikit akibat pada produktifitas; dengan menggunakan editor diagram aksi (Bab-08) dengan bahasa yang terstruktur dengan baik seperti C “yang melakukan” produktifitas perbaikan.



Gambar 4.1 Usage of people during the typical 3GL development life cycle

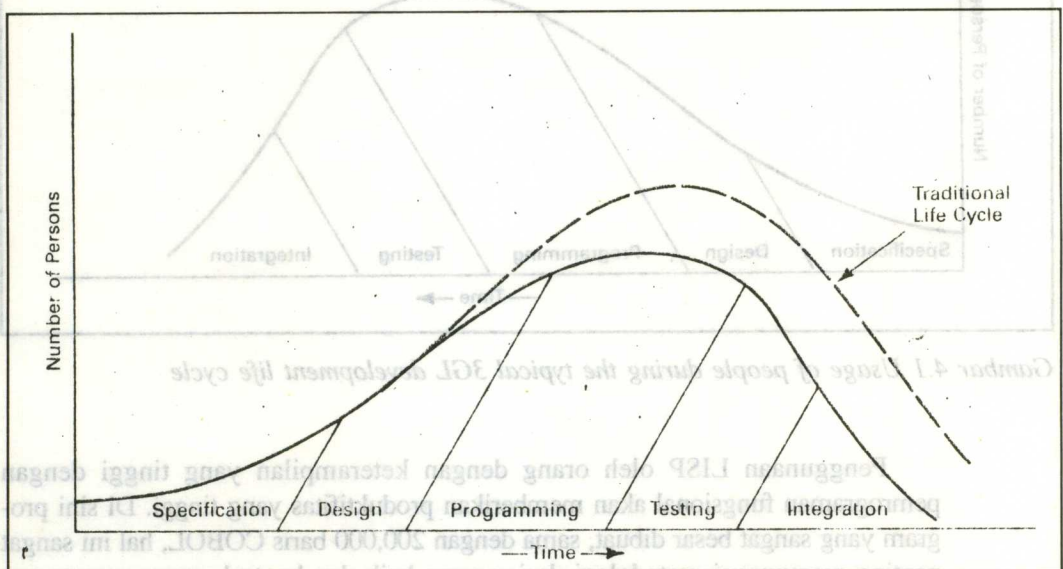
Penggunaan LISP oleh orang dengan keterampilan yang tinggi dengan pemrograman fungsional akan memberikan produktifitas yang tinggi. Di sini program yang sangat besar dibuat, sama dengan 200,000 baris COBOL, hal ini sangat penting mempunyai metodologi design yang baik dan kontrol management yang diambil dari penggunaan 4GL. Hal ini terlalu sering hilang. Metodologi design dan management untuk 4GL mungkin secara lengkap berbeda dengan pengembangan tradisional COBOL atau PL/I. Ini harus mencakup prototype, design iterative, modeling data yang ter-komputerisasi, dan struktur penambahan komputer.

Pembuatan system secara tradisional dilakukan dengan system pengembangan life cycle, diilustrasikan dalam Gambar 3.3. Pembuatan orang-orang dalam tradisional life cycle ditunjukkan dalam Gambar 4.1.

Penggunaan bahasa "pemrograman" yang berbeda akan mempengaruhi hanya bagian program dari life cycle (termasuk penge-test-an dan perawatan). Akibatnya mungkin diilustrasikan dalam Gambar 4.2. Perbaikan ini sangat berharga, tetapi sesuatu yang lebih dari bahasa "pemrograman" masih diperlukan. Jika suatu bahasa membuat laporan, layar, dialog, dan accesses basis data, perbaikan akan lebih besar, tetapi banyak waktu yang tetap diperlukan untuk spesifikasi dan design.

Beberapa peralatan generasi keempat berkonsentrasi pada spesifikasi dan tahap

design dan pembuatan code dari suatu design. Untuk membuat code dari design, design dan spesifikasi harus tepat, sehingga banyak pekerjaan diperlukan pada permukaan akhir cycle. Gambar 4.3 mengilustrasikan hal ini. Usaha utama difokuskan pada spesifikasi, pembuatan prototype, dan design. Area dalam kurva lebih sedikit dibandingkan dengan yang ada dalam Gambar 4.2, tetapi ada keuntungan pokok lainnya. Hilangnya waktu untuk pengembangan diperpendek secara substantif.



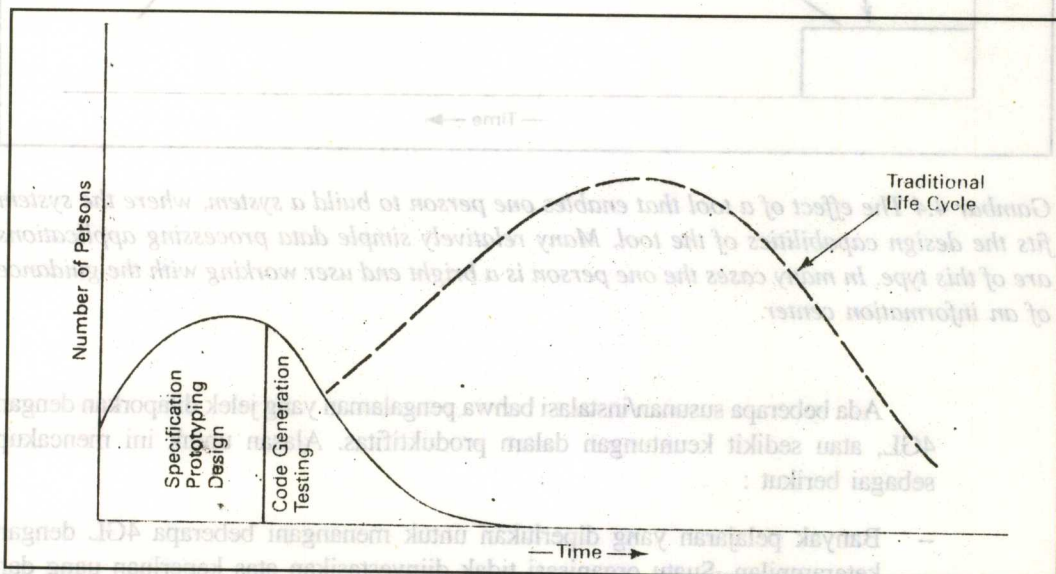
Gambar 4.2 Usage of people with a 4GL that affects only the programming, testing (and maintenance) phases

Tabel 4.1 Programs installed per person-month

Program Type	Using COBOL	Using ADF
Inventory locator	0.3675	20.44
Order billing	0.1365	35.11
Order processing	0.3220	11.11
Average	0.2753	22.22

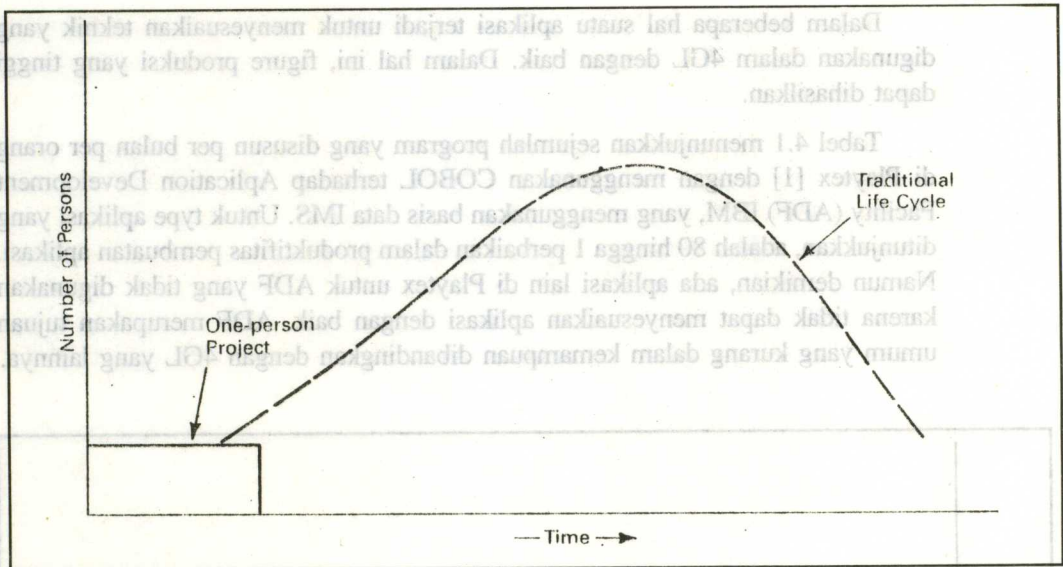
Dalam beberapa hal suatu aplikasi terjadi untuk menyesuaikan teknik yang digunakan dalam 4GL dengan baik. Dalam hal ini, figure produksi yang tinggi dapat dihasilkan.

Tabel 4.1 menunjukkan sejumlah program yang disusun per bulan per orang di Playtex [1] dengan menggunakan COBOL terhadap Application Development Facility (ADF) IBM, yang menggunakan basis data IMS. Untuk type aplikasi yang ditunjukkan, adalah 80 hingga 1 perbaikan dalam produktifitas pembuatan aplikasi. Namun demikian, ada aplikasi lain di Playtex untuk ADF yang tidak digunakan karena tidak dapat menyesuaikan aplikasi dengan baik. ADF merupakan tujuan umum yang kurang dalam kemampuan dibandingkan dengan 4GL yang lainnya.



Gambar 4.3 Usage of a fourth-generation tool that concentrates on specification and design, and generates code from the design.

Jika 80 hingga 1 perbaikan ditemukan dalam beberapa hal yang didokumentasikan, maka ada perkecualian bukannya aturan. Perbaikan 10 hingga 1 atas COBOL, namun demikian, secara tipikal dengan aplikasi DP komersial yang sederhana.



Gambar 4.4 The effect of a tool that enables one person to build a system, where the system fits the design capabilities of the tool. Many relatively simple data processing applications are of this type. In many cases the one person is a bright end user working with the guidance of an information center.

Ada beberapa susunan/instalasi bahwa pengalaman yang jelek dilaporkan dengan 4GL, atau sedikit keuntungan dalam produktifitas. Alasan untuk ini mencakup sebagai berikut :

- Banyak pelajaran yang diperlukan untuk menangani beberapa 4GL dengan keterampilan. Suatu organisasi tidak diinvestasikan atas keperluan uang dan waktu dalam pembuatan team dengan keterampilan dan praktek yang memadai.
- Beberapa pembuat aplikasi terbatas dalam apa yang mereka dapat perbuat. Ketika diterapkan pada system yang tidak semestinya, mereka dapat menyebabkan suatu problem atau gagal mendapatkan hasil yang diminta.
- Untuk mendapatkan pengurangan pokok dalam pengembangan waktu, perubahan pokok dalam teknik dan kontrol management sangat diperlukan. Kadang-kadang kontrol yang semestinya untuk COBOL digunakan dengan pembuat aplikasi, dengan merusak banyak kemampuan untuk pengembangan yang cepat.

