NAMA : Vira Wahyuni Idhayanti

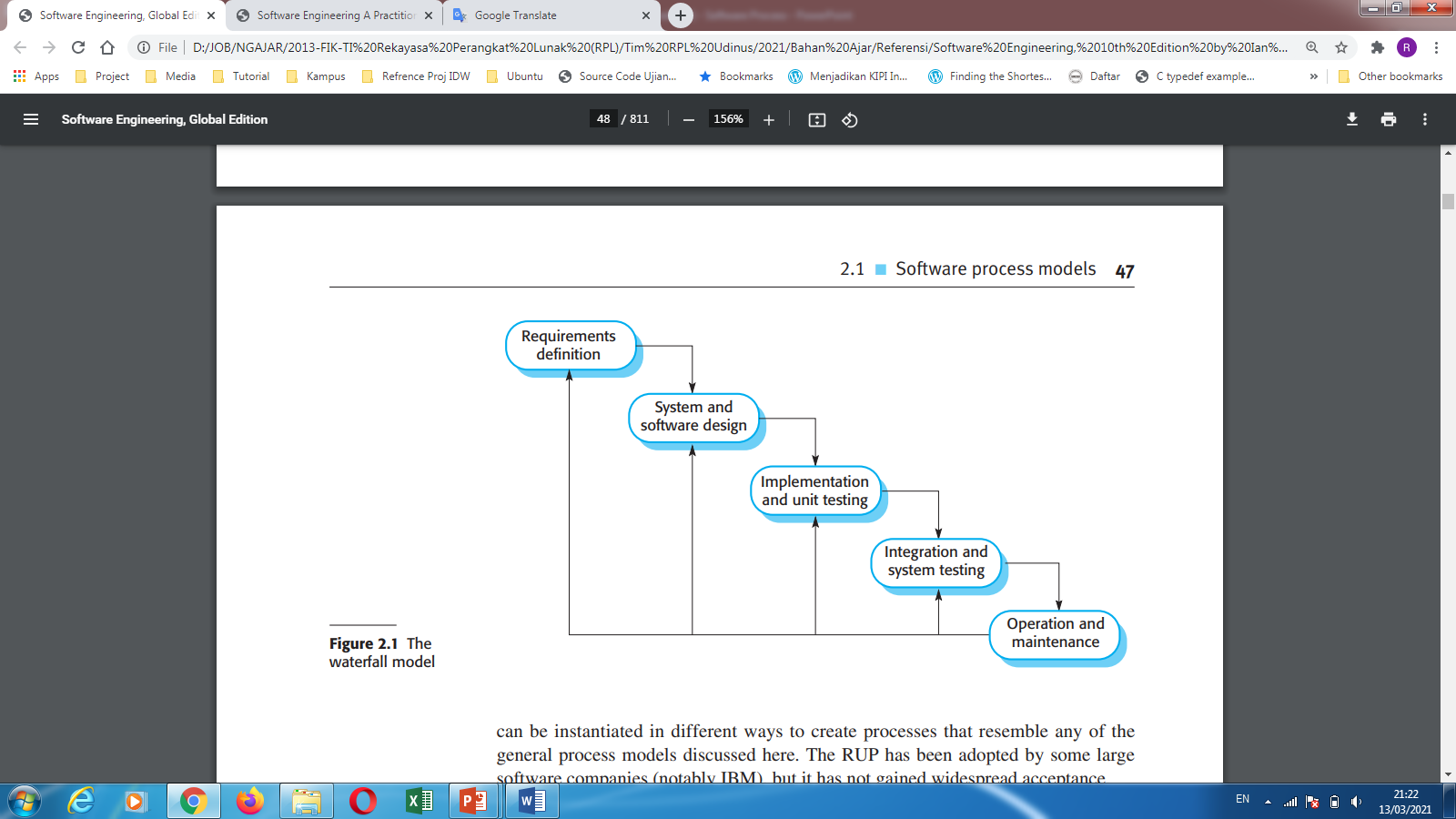
NIM : A11.2019.12119

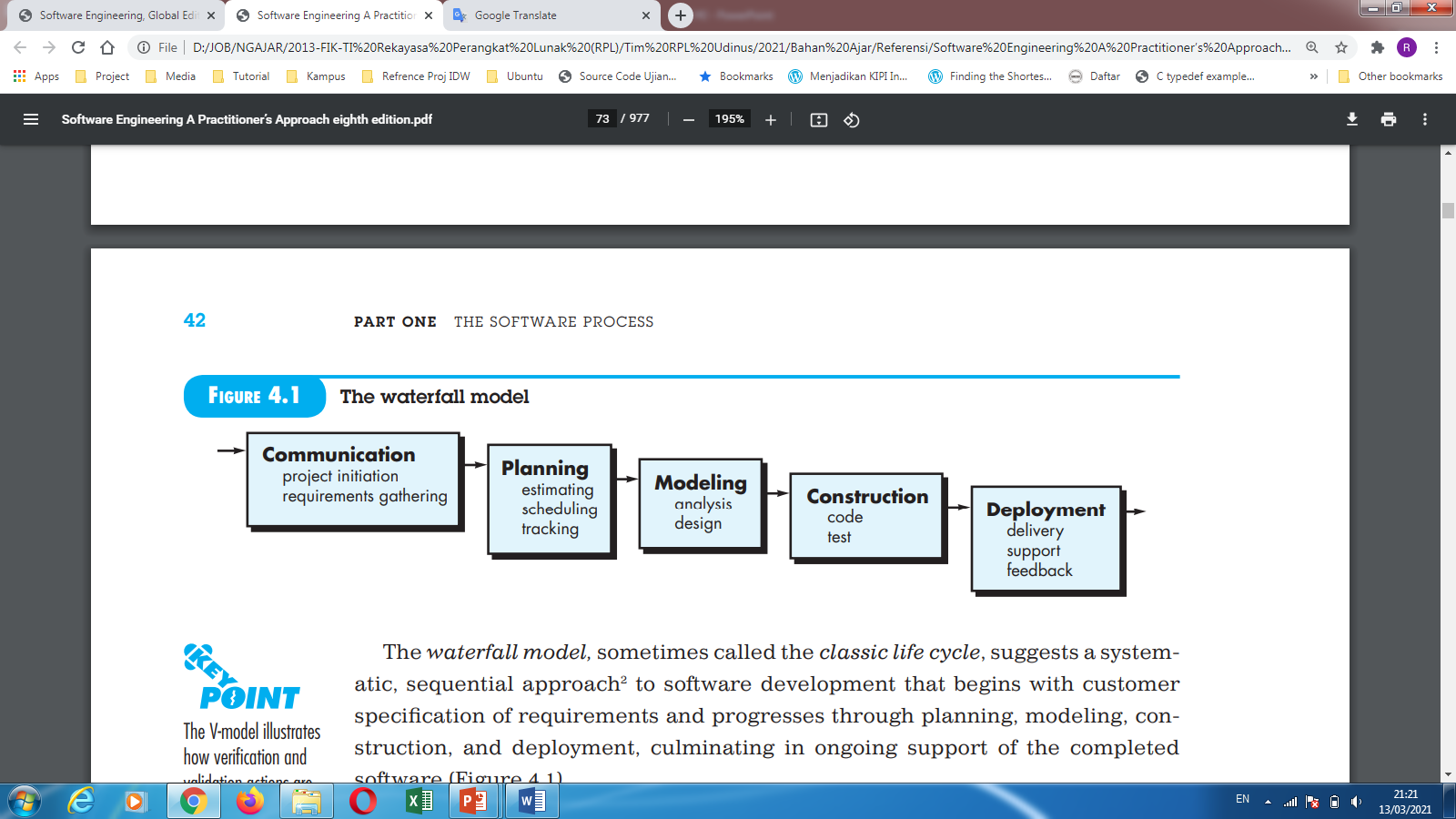
KELOMPOK : A11.4606

MATA KULIAH : Proyek Perangkat Lunak

Model Rekayasa Perangkat Lunak

**Waterfall Model**





* Waterfall Model disebut juga dengan *classic life cycle* merupakan pendekatan sistematis dan sekuensial untuk pengembangan perangkat lunak yang dimulai dengan spesifikasi kebutuhan dari pelanggan dan berkembang melalui perencanaan, pemodelan, konstruksi, dan *deployment*, yang berpuncak pada dukungan berkelanjutan untuk penyelesaian perangkat lunak
* Waterfall Model adalah contoh proses berbasis rencana
* Merencanakan dan menjadwalkan semua aktivitas proses sebelum memulai pengembangan perangkat lunak.
* Hasil dari setiap tahapan dalam model waterfall adalah satu atau lebih dokumen yang disetujui (“ditandatangani”)
* Fase berikut tidak boleh dimulai hingga fase sebelumnya selesai.

**Permasalahan penerapan model Waterfall:**

* Proyek nyata jarang mengikuti aliran berurutan
* Seringkali sulit bagi pelanggan untuk menyatakan semua persyaratan secara eksplisit
* Pelanggan tidak bersabar menunggu hasl program hingga akhir jangka waktu proyek

**Kapan menggunakan Waterfall Model:**

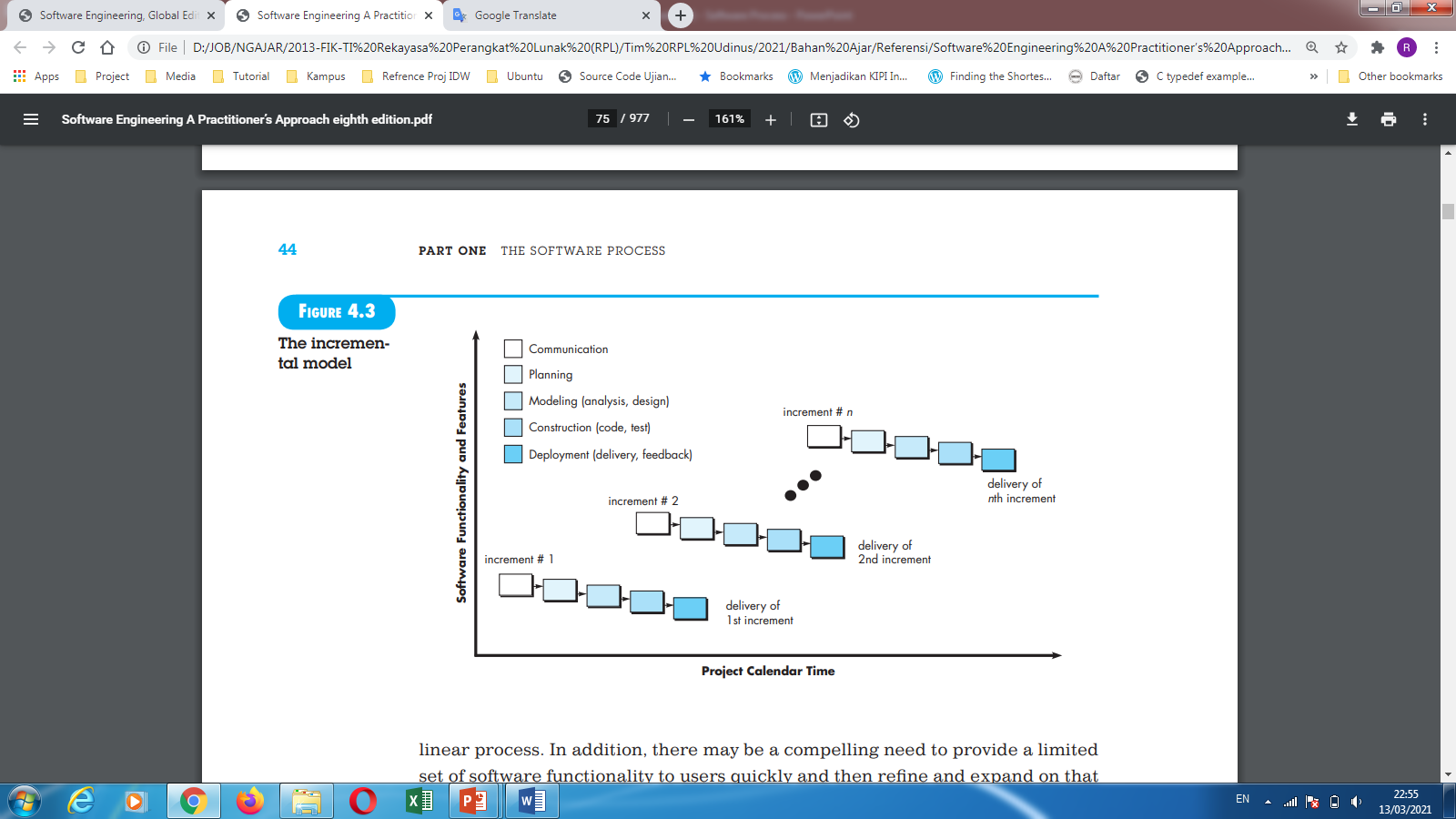
* Ketika persyaratan dipahami dengan baik dan jelas (tidak ada persyaratan yang ambigu)
* Saat perubahan cukup terbatas selama proses desain
* Ketika teknologi dipahami dan tersedia sumber daya yang cukup dengan keahlian yang dibutuhkan.

**Waterfall Model sesuai untuk sistem:**

* **Embedded systems** dimana perangkat lunak harus berinteraksi dengan sistem perangkat keras. Karena perangkat keras tidak fleksibel, biasanya tidak mungkin untuk menunda keputusan tentang fungsionalitas perangkat lunak sampai perangkat lunak itu diterapkan.
* **Critical systems** yang memerlukan analisis keselamatan dan keamanan ekstensif dari spesifikasi dan desain perangkat lunak. Dalam sistem ini, spesifikasi dan dokumen desain harus lengkap sehingga dapat dilakukan analisis. Masalah terkait keselamatan dalam spesifikasi dan desain biasanya sangat mahal untuk diperbaiki pada tahap implementasi.
* **Large software systems** yang merupakan bagian dari sistem rekayasa yang lebih luas yang dikembangkan oleh beberapa perusahaan mitra. Perangkat keras dalam sistem dapat dikembangkan dengan menggunakan model yang serupa, dan perusahaan merasa lebih mudah untuk menggunakan model umum untuk perangkat keras dan perangkat lunak. Lebih lanjut, jika beberapa perusahaan terlibat, spesifikasi lengkap mungkin diperlukan untuk memungkinkan pengembangan subsistem yang berbeda secara independen.

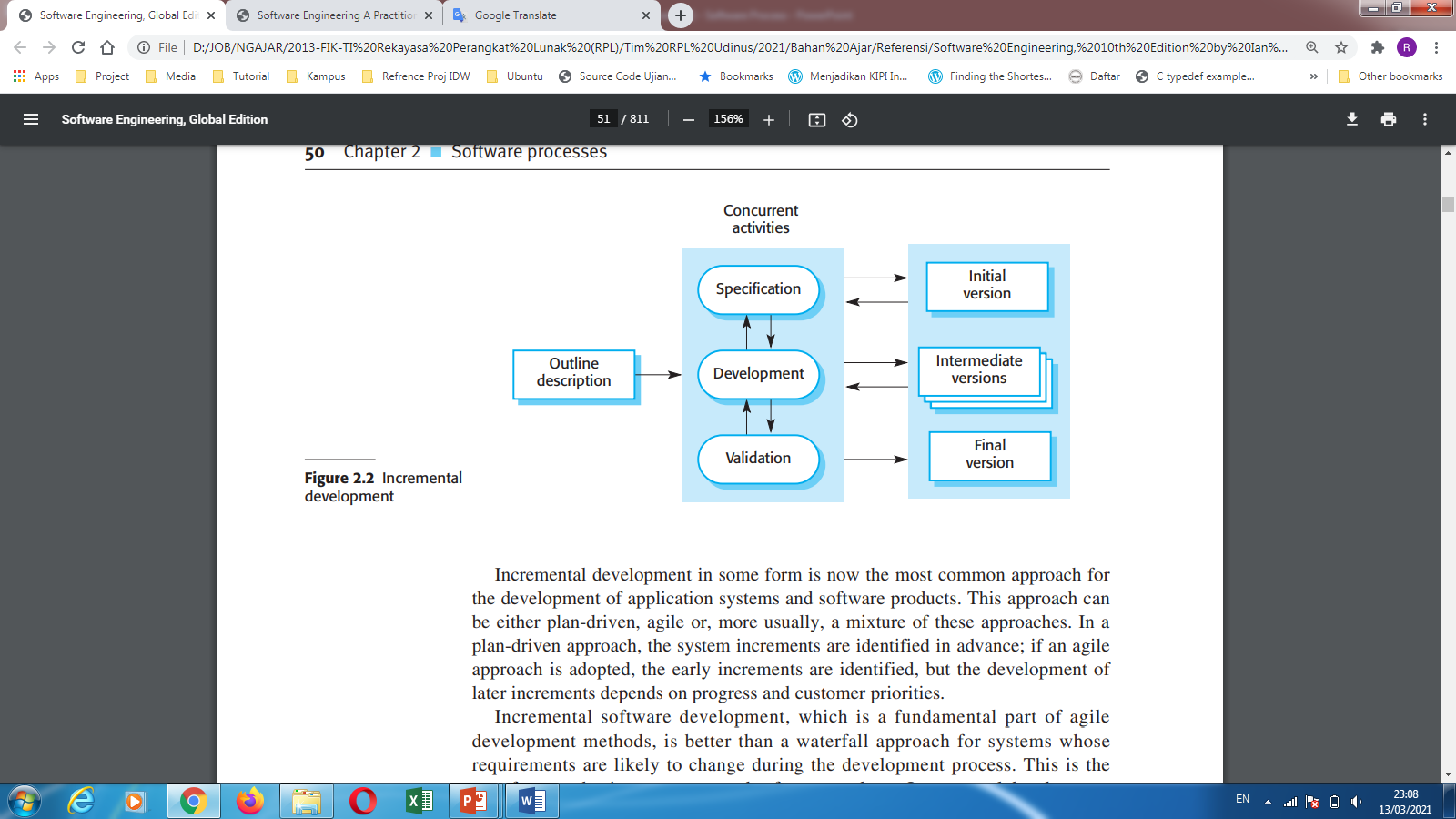
**Increment Model**

* Incremental Model merupakan gabungan antara model linier sekuensial dan prototyping.
* Setiap linier sekuen menghasilkan produk yang *deliverables* (dapat dikirim)
* Increment pertama merupakan produk inti (*core*), yang mengandung persyaratan / kebutuhan dasar.
* Penambahan dilakukan pada increment berikutnya



Delivery of increment:

* **A core product** 🡪 persyaratan dasar ditangani namun banyak fitur tambahan tidak dikirimkan
* **Intermediate product (#1)** 🡪 membahas modifikasi produk inti serta fitur dan fungsionalitas tambahan.
* **Intermediate product (#n)** 🡪 proses ini diulangi untuk lebih memenuhi kebutuhan pelanggan
* **Final product** 🡪 produk lengkap diproduksi dan dikirim



* Incremental development didasarkan pada gagasan mengembangkan implementasi awal, mendapatkan umpan balik dari pengguna
* Mengembangkan perangkat lunak melalui beberapa versi hingga sistem yang dibutuhkan lengkap
* *Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities*
* Bagian fundamental dari pendekatan *agile* yang mencerminkan cara memecahkan masalah
* Sebuah solusi dikembangkan melalui serangkaian langkah dan *backtracking* yang memungkinkan pengembang melakukan evaluasi atau mengambil tindakan korektif ketika ditemukan kesalahan
* Lebih murah dan lebih mudah untuk membuat perubahan dalam perangkat lunak yang sedang dikembangkan.
* Pelanggan dapat mengevaluasi sistem pada tahap yang relatif awal dalam pengembangan untuk melihat apakah sistem tersebut memberikan apa yang dibutuhkan.

**Keuntungan Increment Model**

* Biaya penerapan perubahan atas *requirement* berkurang
* Jumlah analisis dan dokumentasi yang harus dilakukan jauh lebih sedikit daripada yang dibutuhkan dengan waterfall model
* Lebih mudah mendapatkan umpan balik pelanggan atas pekerjaan pengembangan yang telah dilakukan
* Pelanggan dapat mengomentari demonstrasi perangkat lunak dan melihat seberapa banyak yang telah diterapkan
* Pengiriman dan penerapan awal perangkat lunak (*deployment*) mungkinkan, meskipun semua fungsionalitas belum disertakan
* Pelanggan dapat menggunakan dan mendapatkan nilai dari perangkat lunak lebih awal daripada model Waterfall

**Masalah pendekatan Increment** memiliki dua masalah:

* Prosesnya tidak terlihat, sangat sulit bagi manajer untuk mengukur kemajuannya.
* Struktur sistem cenderung menurun seiring bertambahnya penambahan baru karena fungsionalitas baru ditambahkan dengan cara apa pun yang memungkinkan
* Increment tidak cocok untuk sistem yang besar, kompleks, dan berumur panjang, dimana tim yang berbeda mengembangkan bagian sistem yang berbeda
* Sistem yang besar membutuhkan kerangka kerja atau arsitektur yang stabil, dan tanggung jawab tim yang berbeda yang bekerja pada bagian-bagian sistem yang perlu didefinisikan dengan jelas sehubungan dengan arsitektur tersebut.
* Harus direncanakan sebelumnya daripada dikembangkan secara bertahap (Increment)

**Kapan menggunakan Increment Model:**

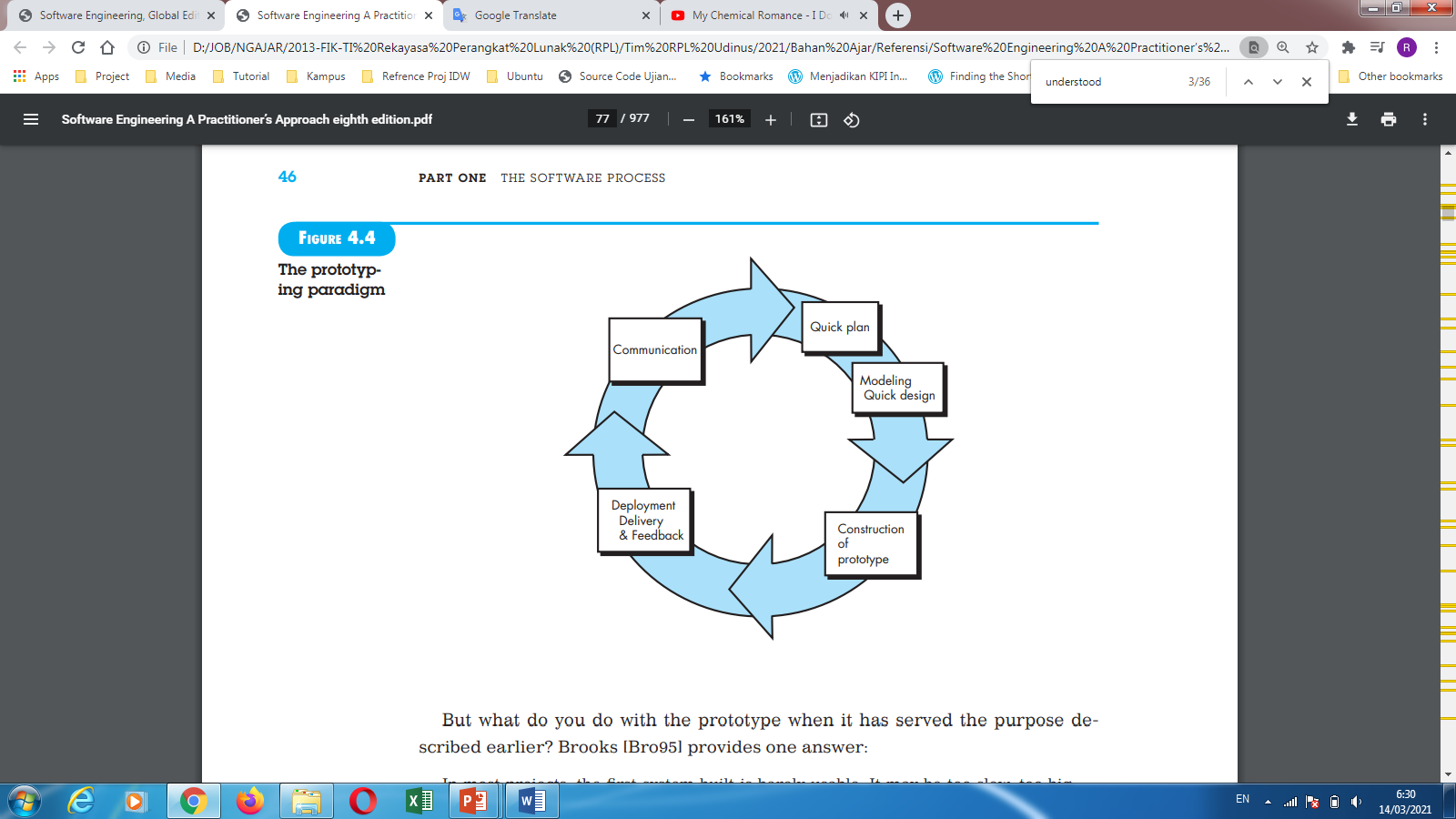
* Saat mengembangkan *non-large system* dengan tim lokal yang bekerja di lokasi yang sama.
* Ketika sistem tidak memiliki atau kurang dalam hal yang berkaitan dengan keselamatan atau keamanan sistem
* Digunakan sebagai platform untuk eksperimen dengan persyaratan dan desain sistem.

**Prototyping Model**

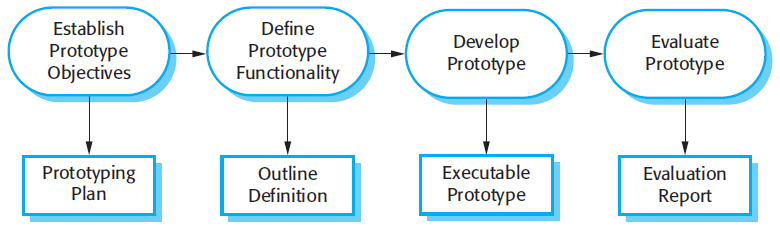
* Versi awal dari sistem perangkat lunak yang digunakan untuk mendemonstrasikan konsep, mencoba opsi desain, dan mencari tahu lebih lanjut tentang masalah dan kemungkinan solusinya.
* Paradigma pembuatan prototype membantu Anda dan pemangku kepentingan lainnya untuk lebih memahami apa yang akan dibangun ketika persyaratan tidak jelas

**Prototyping dapat digunakan untuk:**

* Membantu perolehan dan validasi persyaratan (dalam proses rekayasa persyaratan)
* Mengeksplorasi opsi dan mengembangkan desain UI (dalam proses desain)
* Memeriksa kelayakan desain yang diusulkan;Untuk menjalankan pengujian back-to-back (dalam proses pengujian).



* Communication, menemui pemangku kepentingan, tentukan semua tujuan, garis besar dimana definisi lebih lanjut adalah wajib
* Quick Plan, iterasi prototipe direncanakan dengan cepat
* Modeling Quick Design, berfokus pada representasi dari aspek sistem yang terlihat.
* Construction of Prototype
* Deployment Delivery and Feedback, evaluasi oleh pemangku kepentingan dengan memberikan umpan balik yang digunakan untuk lebih menyempurnakan persyaratan



* Tujuan pembuatan prototipe harus dibuat eksplisit sejak awal proses.
* Buat garis besar (outline), putuskan apa yang akan dimasukkan dan sistem prototipe apa yang tidak digunakan
* Pengembangan, membangun prototipe yang dapat dieksekusi di mana beberapa fungsionalitas keluar dari prototype
* Evaluasi, disesuaikan untuk memenuhi persyaratan dan memberikan pemahaman apa yang perlu dilakukan.

**Keuntungan Prototyping :**

* Improved system usability
* Kecocokan yang lebih dekat dengan kebutuhan nyata pengguna.
* Improved design quality
* Improved maintainability
* Usaha pengembangan berkurang

**Kekurangan Prototyping :**

* Tidak mungkin/sulit untuk menetapkan prototype yang memenuhi persyaratan non-fungsional (keamanan, ketahanan, keandalan)
* Sebagian besar tidak berdokumen
* Sistem akan sulit dan mahal perawatannya
* Standar kualitas organisasi biasanya santai