
No-Code, Low-Code,
dan Inovasi Cepat
Oleh: Rudy C Tarumingkeng
No Code
Low Code

Oleh:
Prof Ir Rudy C Tarumingkeng, PhD
Guru Besar Manajemen NUP: 9903252922
Rektor, Universitas Cenderawasih, Papua (1978-1988, dan
Rektor, Kampus AGRO Manokwari sekarang Universitas Papua Manokwari)
Coordinator, CIDA/DIKTI SFU Burnaby BC Canada 1988-1991
Rektor, Universitas Kristen Krida Wacana, Jakarta (1991-2000)
Ketua Dewan Guru Besar, IPB-University, Bogor (2005-2006)
AI - Data Analyst, dan Ketua Senat Akademik, IBM-ASMI, Jakarta 2024-

---

rudyct75@gmail.com
15 Maret 2026

NO-CODE, LOW-CODE, DAN INOVASI CEPAT

No-Code, Low-Code, dan Inovasi Cepat

Pendahuluan

Dalam banyak organisasi modern, tantangan terbesar bukan lagi sekadar menemukan ide, melainkan mengubah ide menjadi solusi yang dapat dipakai dalam waktu yang cukup cepat. Unit bisnis ingin aplikasi internal untuk mempercepat layanan. Divisi operasional membutuhkan alur kerja yang lebih efisien. Tim pemasaran ingin eksperimen digital berjalan dalam hitungan hari, bukan bulan. Sementara itu, tim TI sering menghadapi tumpukan permintaan, keterbatasan talenta, integrasi sistem lama, tuntutan keamanan, dan kebutuhan kepatuhan yang terus meningkat. Di tengah tekanan semacam ini, pendekatan *no-code* dan *low-code* muncul sebagai jawaban praktis: bagaimana mempercepat pengembangan solusi digital tanpa selalu bergantung pada proses pemrograman tradisional yang panjang dan mahal. Microsoft mendefinisikan *low-code development* sebagai metode membuat aplikasi dengan lebih sedikit kode dibanding pendekatan *code-first*, menggunakan alat visual, *drag-and-drop*, dan otomasi; IBM menjelaskan *low-code* sebagai pendekatan *rapid application development* yang menghasilkan kode secara otomatis melalui blok visual, sedangkan *no-code* memungkinkan pengguna membuat aplikasi dan mengotomasi proses bisnis tanpa menulis kode sama sekali. (Microsoft)

Kehadiran *no-code* dan *low-code* tidak dapat dipahami hanya sebagai tren teknis. Keduanya merupakan bagian dari pergeseran yang lebih besar dalam transformasi digital: dari model inovasi yang tersentralisasi di unit TI ke model inovasi yang lebih terdistribusi, di mana pengguna bisnis, analis, operator, dan profesional non-programmer mulai ikut membangun solusi digital. Microsoft menyebut para pengguna seperti

ini sebagai *citizen developers*, yaitu orang-orang di dalam organisasi yang tidak selalu memiliki keahlian pemrograman mendalam tetapi mampu memakai platform visual untuk membangun aplikasi yang mendukung kebutuhan kerja. IBM juga menekankan bahwa *low-code* dan *no-code* mendemokratisasi pengembangan aplikasi dengan memberi ruang bagi pengguna lini bisnis untuk ikut membangun solusi. (Microsoft)

Namun, percepatan inovasi tidak pernah bebas biaya. Semakin mudah pembuatan aplikasi, semakin besar pula risiko munculnya *shadow IT*, duplikasi data, lemahnya kontrol versi, celah keamanan, dan solusi yang cepat dibuat tetapi sulit dipelihara. Microsoft secara eksplisit memperingatkan bahwa pengembangan oleh *citizen developers* dapat dengan mudah berubah menjadi *shadow IT* bila berlangsung tanpa pengawasan TI dan keamanan, sedangkan McKinsey mencatat bahwa penggunaan *low-code/no-code* oleh nondeveloper dapat dibatasi oleh kebutuhan akan pengembang berpengalaman untuk memantau dan melakukan *debugging* aplikasi; McKinsey juga menyoroti tantangan *monitoring*, kontrol versi, kualitas, keamanan, kekayaan intelektual, dan tanggung jawab hukum dalam lingkungan pengembangan yang semakin terdorong oleh otomasi dan AI. Bahkan Gartner, dalam tren keamanan siber 2026, menyebut bahwa platform *no-code/low-code* bersama *vibe coding* dapat memperluas proliferasi agen AI yang tidak terkelola, kode yang tidak aman, dan potensi pelanggaran kepatuhan regulasi. (Microsoft)

Karena itu, diskusi tentang *no-code*, *low-code*, dan inovasi cepat harus ditempatkan dalam kerangka yang lebih luas. Pertanyaannya bukan hanya apakah platform ini dapat mempercepat pembuatan aplikasi, melainkan juga: inovasi seperti apa yang dipercepat, oleh siapa, untuk kebutuhan apa, dengan tata kelola bagaimana, dan dengan konsekuensi jangka panjang apa bagi organisasi. Esai ini membahas ketiga tema tersebut secara akademik dan naratif. Pertama, dijelaskan pengertian *no-code* dan *low-code* serta perbedaannya. Kedua, dibahas mengapa

keduanya berkaitan erat dengan inovasi cepat. Ketiga, dianalisis manfaat strategisnya bagi organisasi. Keempat, ditelaah keterbatasan dan risikonya. Kelima, dibahas kebutuhan tata kelola agar percepatan inovasi tidak berujung pada kekacauan digital. Keenam, dibahas hubungan baru antara *no-code/low-code* dan AI. Argumen utamanya adalah bahwa *no-code* dan *low-code* dapat menjadi mesin inovasi cepat yang sangat kuat, tetapi hanya bila organisasi memadukan kecepatan dengan arsitektur, pelatihan, tata kelola, dan disiplin strategis yang memadai. (Microsoft)

1. Memahami No-Code dan Low-Code

Secara sederhana, *no-code* adalah pendekatan pengembangan perangkat lunak yang memungkinkan pengguna membuat aplikasi dan mengotomasi proses bisnis tanpa menulis kode. IBM menekankan bahwa *no-code* menggunakan antarmuka visual dan alat *drag-and-drop* sehingga pengguna nonteknis dapat membangun perangkat lunak fungsional dengan sedikit atau tanpa pengetahuan pemrograman. Microsoft menambahkan bahwa pendekatan ini memungkinkan praktis siapa pun di organisasi membangun aplikasi yang sesuai dengan infrastruktur dan alur kerja yang ada, dengan pengetahuan kode nol, khususnya untuk proyek web dan seluler yang relatif sederhana. (ibm.com)

Sementara itu, *low-code* berada di wilayah tengah antara *code-first* tradisional dan *no-code*. IBM menyebutnya sebagai pendekatan *rapid application development* yang mengandalkan otomatisasi kode melalui blok bangunan visual seperti *drag-and-drop* dan menu tarik-turun, tetapi tetap memungkinkan pengguna menambahkan kode di atas kode yang dihasilkan secara otomatis. Microsoft menjelaskan bahwa *low-code* adalah metode pengembangan dengan lebih sedikit pengkodean daripada pengembangan tradisional, sehingga orang dengan pengalaman coding dasar—bukan hanya pengembang profesional—dapat membuat aplikasi lebih cepat dan lebih mudah. (ibm.com)

Perbedaan konseptual antara keduanya penting. *No-code* umumnya cocok untuk kebutuhan yang lebih terstandar, seperti formulir, alur persetujuan, aplikasi survei, otomasi proses sederhana, atau aplikasi internal yang tidak memerlukan kustomisasi mendalam. Sebaliknya, *low-code* memberi ruang lebih besar bagi integrasi, kustomisasi, pengembangan logika yang lebih kompleks, dan kolaborasi antara pengguna bisnis dan pengembang profesional. Microsoft menegaskan bahwa *low-code* tidak akan menggantikan coding tradisional; untuk proyek yang sangat kompleks atau sepenuhnya terkustomisasi, pendekatan *code-first* tetap lebih tepat, sedangkan dalam banyak kasus kombinasi *low-code* dan *code-first* justru paling efektif. (Microsoft)

Di titik ini, kita dapat melihat bahwa *no-code* dan *low-code* bukan dua dunia yang benar-benar terpisah, melainkan spektrum. Bahkan Microsoft Power Fx menggambarkan kontinuitas tersebut dengan cukup jelas: ia disebut sebagai bahasa *low-code* yang dapat membawa pengguna dari wilayah *no-code* menuju *pro-code* tanpa jurang belajar yang tajam, sehingga tim dengan tingkat keterampilan berbeda dapat berkolaborasi dan menghemat waktu serta biaya. Ini menunjukkan bahwa masa depan pengembangan digital tidak lagi harus dibelah secara biner antara "programmer" dan "bukan programmer", tetapi semakin berbentuk kontinuum kapabilitas. (Microsoft Learn)

2. Mengapa No-Code dan Low-Code Berkaitan dengan Inovasi Cepat

Kata kunci utama dalam *no-code* dan *low-code* adalah percepatan. Microsoft menyebut manfaat utama *low-code* sebagai kecepatan dan penghematan biaya, karena platform semacam itu dapat dipakai untuk merancang dan membangun aplikasi, alur kerja, atau proses dengan cepat serta membantu organisasi melewati *development backlog* lebih cepat. Pada sisi *no-code*, Microsoft menekankan bahwa ide dapat dibawa dari awal hingga selesai dalam waktu lebih singkat dan dengan sumber daya lebih sedikit dibanding bila dikerjakan sepenuhnya dengan kode atau dialihdayakan ke pengembang profesional. IBM juga menyatakan

bahwa *no-code* memungkinkan tim, misalnya tim pemasaran, menyusun aplikasi survei dalam hitungan jam, bukan minggu. (Microsoft)

Kecepatan ini muncul karena beberapa sebab. Pertama, banyak pekerjaan teknis yang sebelumnya harus ditulis dari nol kini sudah dikemas dalam komponen siap pakai: formulir, konektor, alur persetujuan, logika dasar, notifikasi, dan integrasi standar. Kedua, *prototyping* menjadi jauh lebih cepat karena pengguna dapat melihat hasil visual lebih awal dan langsung mengujinya di konteks kerja nyata. Ketiga, proses pengembangan menjadi lebih dekat ke pengguna akhir; orang yang memahami proses bisnis tidak harus menunggu lama menjelaskan kebutuhan kepada pengembang, lalu menunggu terjemahannya ke dalam kode. Mereka dapat langsung membangun atau setidaknya mengonfigurasi solusi itu sendiri. (Microsoft)

Dalam perspektif inovasi, percepatan ini sangat penting. Inovasi organisasi sering gagal bukan karena tidak ada ide, tetapi karena siklus dari ide ke implementasi terlalu lama. Jika sebuah unit bisnis harus menunggu berbulan-bulan untuk aplikasi sederhana, maka momentum inovasi hilang. *No-code/low-code* mempersingkat jarak antara identifikasi masalah dan pembuatan solusi. Itulah sebabnya Microsoft menyatakan bahwa siapa pun di organisasi dapat berperan dalam mempercepat proses daripada menunggu orang yang memiliki keahlian coding punya waktu untuk mengerjakannya. Platform *low-code* bahkan disebut dapat meningkatkan produktivitas pengembang dengan membebaskan mereka untuk fokus pada pekerjaan yang memang membutuhkan coding ekstensif. (Microsoft)

Inovasi cepat juga berkaitan dengan kemampuan bereksperimen. Dalam lingkungan bisnis yang berubah cepat, organisasi perlu menguji gagasan baru dengan biaya kegagalan yang rendah. *No-code* dan *low-code* mendukung logika ini karena memungkinkan pembuatan *minimum viable solution* secara lebih cepat. Sebuah tim HR, misalnya, dapat menguji alur *onboarding* digital sederhana sebelum menginvestasikan

proyek TI besar. Tim layanan pelanggan dapat mencoba aplikasi tiket internal baru sebelum memutuskan pembelian sistem besar. Tim keuangan dapat membangun otomasi persetujuan pengeluaran untuk melihat dampaknya terlebih dahulu. Dalam semua contoh ini, platform visual menurunkan ambang masuk bagi eksperimen digital. (Microsoft)

3. Demokratisasi Pengembangan dan Munculnya Citizen Developer

Salah satu implikasi terpenting dari *no-code/low-code* adalah demokratisasi pengembangan perangkat lunak. Microsoft menyebut bahwa platform ini membuat pembangunan aplikasi lebih mudah diakses oleh orang dengan pengalaman coding yang lebih sedikit, yang sering disebut *citizen developers*. IBM juga menggambarkan *citizen developers* sebagai pengguna bisnis dengan keahlian formal pemrograman yang minim, seperti analis bisnis atau manajer proyek, yang kini dapat terlibat dalam pengembangan aplikasi. (Microsoft)

Demokratisasi ini memiliki dimensi manajerial yang besar. Selama bertahun-tahun, pengembangan perangkat lunak di banyak organisasi menjadi fungsi yang sangat tersentralisasi. Setiap perubahan proses harus diterjemahkan dulu ke dalam spesifikasi, masuk ke antrian TI, diprioritaskan, diuji, dan baru kemudian dirilis. Model ini sering efektif untuk sistem besar dan kritikal, tetapi tidak selalu cocok untuk kebutuhan unit kerja yang cepat berubah. *Citizen development* menawarkan cara baru: pemilik proses ikut menjadi pembuat solusi. Orang HR dapat mengotomasi formulir tertentu. Analis keuangan dapat membangun dashboard kecil. Tim operasional dapat membuat aplikasi inspeksi lapangan. Inovasi menjadi lebih dekat ke akar masalah. (Microsoft)

Akan tetapi, demokratisasi tidak identik dengan anarki. Microsoft secara terbuka menekankan perlunya *low-code governance*, yaitu cara organisasi membimbing pengembang profesional dan *citizen developers* dalam membangun aplikasi kustom dengan platform *low-code*. Menurut Microsoft, governance diperlukan agar para pembuat solusi mematuhi

kebijakan, prosedur keamanan, dan standar organisasi; kerangka governance dapat menentukan apa yang boleh dibangun, siapa yang harus meninjau dan menyetujui aplikasi, serta bagaimana praktik terbaik TI diikuti. (Microsoft)

Dari sini tampak suatu prinsip penting: *citizen developer* bukan pengganti pengembang profesional, melainkan pelengkap. Microsoft menyatakan bahwa pengembang profesional tetap memegang peran karena pekerjaan pengembangan tradisional tetap membutuhkan *code review* dan pemeliharaan, sementara *citizen developers* lebih banyak melihat peluang perbaikan proses di tim mereka sendiri. Dengan kata lain, *no-code/low-code* yang sehat membentuk kolaborasi baru antara domain expert dan technical expert. (Microsoft)

4. No-Code, Low-Code, dan Produktivitas Organisasi

Salah satu alasan utama organisasi tertarik pada *no-code/low-code* adalah produktivitas. Microsoft menyebut bahwa platform *low-code* dapat menghemat waktu, meningkatkan produktivitas pengembang, mengurangi biaya, menambah fleksibilitas, dan membantu mengisi kesenjangan talenta. IBM juga menyatakan bahwa *no-code* dapat meningkatkan efisiensi, mengurangi kesalahan, serta menghasilkan penghematan biaya yang signifikan dengan memanfaatkan staf internal untuk membuat atau memodifikasi aplikasi tertentu. (Microsoft)

Produktivitas ini dapat dilihat pada beberapa level. Pada level individu, pengguna tidak perlu selalu menunggu bantuan teknis untuk menyelesaikan masalah digital sederhana. Pada level tim, proses manual dapat diotomasi dengan lebih cepat. Pada level organisasi, backlog TI dapat dikurangi karena tidak semua permintaan harus masuk ke jalur rekayasa perangkat lunak penuh. Pada level strategis, pengembang profesional dapat dipindahkan dari pekerjaan berulang ke pekerjaan yang lebih bernilai tinggi, seperti arsitektur, keamanan, integrasi kompleks, dan sistem inti. Microsoft secara eksplisit menyatakan bahwa *low-code* membantu pengembang profesional mempercepat

pengembangan sehingga waktu mereka dapat dipakai untuk proyek yang memerlukan coding yang lebih ekstensif. (Microsoft)

Produktivitas juga terkait dengan modernisasi aplikasi. Microsoft menyatakan bahwa kapabilitas *low-code* Power Platform memungkinkan organisasi membangun dan menerapkan aplikasi modern lebih cepat dan lebih hemat biaya, sambil mengintegrasikan sistem dan sumber data yang ada, serta menambahkan AI untuk meningkatkan pengalaman pengguna dan otomatisasi proses. Hal ini menunjukkan bahwa *low-code* bukan hanya alat membuat aplikasi sederhana, tetapi juga bagian dari strategi modernisasi proses dan sistem. (Microsoft Learn)

Jika dikaitkan dengan AI, logika produktivitas ini menjadi lebih kuat lagi. McKinsey melaporkan bahwa pengembang perangkat lunak dapat menyelesaikan tugas hingga dua kali lebih cepat dengan alat generative AI, dan bahwa teknologi tersebut membantu memulai draf awal kode, menghadapi tantangan baru, serta mempercepat pencarian solusi. Walaupun temuan McKinsey berbicara tentang pengembangan perangkat lunak secara lebih luas, implikasinya sangat relevan bagi dunia *low-code/no-code*: ketika pembangunan solusi visual bertemu dengan AI berbasis *prompting*, hambatan awal inovasi menjadi semakin rendah. (McKinsey & Company)

5. Ruang Penerapan: Dari Workflow Sederhana sampai Integrasi dan Agen AI

Sering ada kesalahpahaman bahwa *no-code/low-code* hanya cocok untuk aplikasi kecil dan sepele. Kenyataannya, ruang penerapannya cukup luas, meskipun tetap tidak tanpa batas. Microsoft menyebut contoh *low-code* dalam aplikasi pengalaman pelanggan, aplikasi lini bisnis seperti manajemen pengeluaran, penganggaran, *onboarding*, tiket TI, serta otomasi proses berulang seperti input data pelanggan. IBM menambahkan bahwa *no-code* sangat berguna untuk otomasi alur kerja lintas fungsi seperti HR, keuangan, dan operasi. (Microsoft)

Salah satu area yang semakin penting adalah integrasi. IBM mendefinisikan *low-code integration* sebagai metode menghubungkan aplikasi, platform, dan *data pipelines* dengan konektor siap pakai dan alat visual, sehingga baik pengembang profesional maupun pengguna bisnis dapat menghubungkan sistem, membangun dan mengotomasi alur kerja, serta menyinkronkan data. Ini sangat relevan dalam organisasi yang memiliki banyak aplikasi terpisah dan ingin mengurangi ketergantungan pada pengembangan integrasi kustom yang mahal. (ibm.com)

Perkembangan terbaru juga menunjukkan konvergensi antara *no-code/low-code* dan AI. IBM telah memperkenalkan pendekatan *no-code/low-code* untuk mengembangkan agen dengan watsonx Orchestrate, dan Gartner memperingatkan bahwa platform *no-code/low-code* dapat memperluas penggunaan agen AI yang tidak terkelola jika tidak diawasi dengan baik. Pada sisi Microsoft, Power Fx menunjukkan bahwa bahasa *low-code* kini dirancang semakin ramah manusia, mirip formula spreadsheet, sehingga tim yang beragam dapat bekerja bersama. Artinya, inovasi cepat masa kini tidak lagi sekadar berarti "membuat aplikasi lebih cepat", tetapi juga "membuat otomatisasi cerdas lebih cepat", termasuk agen, alur keputusan, dan integrasi yang sebelumnya hanya bisa dikerjakan oleh tim teknis yang lebih besar. (@ibmdeveloper)

6. Keterbatasan: No-Code dan Low-Code Bukan Obat untuk Semua

Walaupun menjanjikan percepatan, *no-code* dan *low-code* bukan solusi universal. Microsoft dengan jelas menyatakan bahwa *low-code development* tidak akan menggantikan coding tradisional dan bahwa proyek yang kompleks atau sepenuhnya terkustomisasi tetap lebih cocok dikerjakan dengan pendekatan *code-first*. Bahkan pada aplikasi *low-code*, pengembang—baik profesional maupun *citizen developers*—tetap perlu memahami bagaimana aplikasi dipakai untuk mendukung misi

organisasi, prioritas bisnis, dan integrasi dengan alur kerja yang ada. (Microsoft)

McKinsey memberi peringatan yang lebih tajam. Dalam laporan tren teknologi 2024, McKinsey menyebut bahwa pertumbuhan penggunaan alat *low-code* dan *no-code* oleh nondeveloper dapat dibatasi karena pengembang berpengalaman tetap dibutuhkan untuk memantau dan melakukan *debugging* aplikasi. Laporan itu juga menyoroti bahwa *monitoring* menyeluruh dan kontrol versi menjadi lebih sulit bila ada perubahan dan peningkatan dari banyak vendor yang tidak terkoordinasi. Dengan kata lain, percepatan pembangunan tidak otomatis berarti percepatan pengelolaan. Dalam banyak kasus, yang cepat dibuat justru bisa mahal dipelihara bila arsitekturnya buruk.

Keterbatasan lain adalah risiko kualitas dan keamanan. Microsoft sendiri menempatkan keamanan, skalabilitas, dan governance sebagai pertimbangan utama dalam memilih platform *low-code*. McKinsey menambahkan bahwa kualitas dan keamanan tetap menjadi perhatian, terutama ketika kode yang dihasilkan secara otomatis atau dibantu AI tidak diperbarui dengan standar terbaru atau tidak dilatih pada kode yang bersih dan cepat. Ini berarti organisasi tidak boleh mengasumsikan bahwa solusi visual otomatis lebih aman hanya karena lebih mudah dibangun. (Microsoft)

Selain itu, ada batasan organisasi yang bersifat manusiawi. Tidak semua pengguna bisnis ingin atau mampu menjadi *citizen developer*. Sebagian mungkin antusias, sebagian lain tidak punya waktu, minat, atau kemampuan analitis yang cukup. Microsoft sendiri mengakui bahwa siapa pun di organisasi dapat membangun aplikasi bila memiliki waktu, minat, dan setidaknya sedikit kecakapan teknis. Artinya, demokratisasi tetap bergantung pada kesiapan manusia, bukan sekadar ketersediaan alat. (Microsoft)

7. Inovasi Cepat dan Bahaya Shadow IT

Semakin cepat inovasi didorong ke unit-unit bisnis, semakin besar peluang munculnya *shadow IT*. Microsoft menjelaskan bahwa *citizen development* dapat dengan mudah keluar jalur menjadi penggunaan perangkat keras atau perangkat lunak tanpa pengetahuan atau persetujuan tim TI atau keamanan, sehingga memunculkan risiko keamanan, kebocoran data, dan pelanggaran kepatuhan. Inilah salah satu paradoks besar *no-code/low-code*: alat diciptakan untuk mempercepat inovasi, tetapi bila tidak diatur justru dapat menghasilkan fragmentasi digital. (Microsoft)

Bayangkan sebuah perusahaan dengan lima divisi yang masing-masing membangun aplikasi kecil sendiri untuk kebutuhan lokal. Dalam jangka pendek, semua tampak efisien. Setiap divisi merasa lebih cepat bergerak. Namun enam bulan kemudian muncul pertanyaan: data pelanggan tersimpan di mana, siapa yang memiliki hak akses, mengapa ada tiga versi formulir yang berbeda, bagaimana aplikasi itu terhubung ke sistem pusat, dan siapa yang bertanggung jawab jika pembuatnya pindah kerja? Di titik ini, "inovasi cepat" mulai berubah menjadi "kompleksitas cepat".

Karena itu, inovasi cepat harus dibedakan dari inovasi liar. Organisasi yang matang tidak akan melarang semua inisiatif *citizen development*, tetapi juga tidak akan membiarkannya berjalan tanpa rel. Microsoft menyarankan bahwa governance perlu mendefinisikan aturan bagi *citizen developers*, menetapkan syarat keamanan yang ketat, menentukan siapa yang memenuhi syarat mengikuti program, melatih mereka dalam keamanan data dan pengembangan berkelanjutan, serta menugaskan departemen TI untuk memberi persetujuan dan pengawasan atas sumber daya yang digunakan. (Microsoft)

8. Governance: Syarat agar Kecepatan Tidak Merusak Organisasi

Jika ada satu konsep yang harus selalu menyertai pembicaraan tentang *low-code/no-code*, konsep itu adalah governance. Microsoft mendefinisikan *low-code governance* sebagai cara organisasi membimbing pengembang profesional dan *citizen developers* dalam

menggunakan platform *low-code*, sambil menjaga keamanan, kepatuhan, dan nilai bisnis. Microsoft juga menekankan bahwa governance yang baik memengaruhi bagaimana tujuan organisasi ditetapkan dan dicapai, bagaimana risiko dipantau dan ditangani, dan bagaimana kinerja dioptimalkan. (Microsoft)

Secara praktis, governance mencakup beberapa hal. Pertama, klasifikasi solusi: aplikasi seperti apa yang boleh dibangun oleh *citizen developers*, dan aplikasi seperti apa yang harus naik ke tim rekayasa penuh. Kedua, kontrol data: siapa yang boleh membuat basis data, sumber data apa yang boleh dipakai, dan bagaimana kebijakan retensi, berbagi, dan penggunaan data dijalankan. Ketiga, kontrol akses dan identitas: siapa yang boleh mengakses, mengubah, menerbitkan, dan menghapus aplikasi. Keempat, alur persetujuan: aplikasi mana yang perlu ditinjau oleh TI, keamanan, hukum, atau pemilik proses. Kelima, dokumentasi dan kepemilikan: siapa pemilik aplikasi dan bagaimana transisi dilakukan jika pemiliknya pindah. Beberapa unsur ini dijelaskan cukup langsung oleh Microsoft, termasuk pentingnya kebijakan manajemen data untuk mencegah duplikasi dan paparan data yang tidak perlu. (Microsoft)

Di sinilah terlihat bahwa governance bukan penghambat inovasi. Governance justru syarat bagi inovasi cepat yang berkelanjutan. Inovasi yang tidak didukung struktur akan mudah menghasilkan solusi lokal yang tidak dapat diskalakan, tidak dapat diaudit, dan tidak dapat dipercaya. Sebaliknya, governance yang terlalu birokratis juga berbahaya karena membunuh kecepatan yang justru menjadi alasan utama adopsi *no-code/low-code*. Tantangan pemimpin adalah menemukan titik keseimbangan: kontrol yang cukup kuat untuk melindungi organisasi, tetapi cukup ringan untuk menjaga ruang eksperimen. (Microsoft)

9. Hubungan Baru antara Low-Code, No-Code, dan AI

Dalam fase terkini transformasi digital, *no-code* dan *low-code* tidak lagi berdiri sendiri. Keduanya semakin bertemu dengan AI, terutama AI generatif dan agen digital. Microsoft Power Fx menggambarkan satu

arah perkembangan penting: bahasa *low-code* yang ramah manusia, seperti formula Excel, dapat menjembatani pembuat nonteknis dan pengembang profesional dalam satu spektrum kolaborasi. Sementara itu, McKinsey menunjukkan bahwa alat generatif berbasis *prompting* dapat membantu pengembang memulai draf, memahami kerangka baru, serta menyelesaikan tugas lebih cepat—hingga dua kali pada beberapa konteks. (Microsoft Learn)

Kombinasi ini berpotensi melahirkan gelombang baru inovasi cepat. Jika dahulu pengguna bisnis hanya bisa menyeret komponen visual, kini mereka mulai dapat membangun logika, formula, alur, dan bahkan agen dengan dukungan bahasa alami. Hal ini sangat menarik untuk organisasi yang ingin mempercepat digitalisasi tanpa menunggu siklus proyek besar. Akan tetapi, justru karena semakin mudah, risikonya pun membesar. Gartner memperingatkan bahwa *no-code/low-code* dan *vibe coding* memperluas proliferasi agen AI yang tidak terkelola, kode yang tidak aman, dan potensi pelanggaran kepatuhan. Dalam konteks ini, governance menjadi semakin kritis, bukan semakin opsional. (Gartner)

AI juga mengubah arti "keahlian" dalam dunia *low-code/no-code*. Pengguna bukan lagi hanya perlu tahu cara menyeret komponen, tetapi juga cara memberi konteks yang benar, memeriksa hasil, memahami data yang dipakai, dan mengenali batas kemampuan alat. McKinsey menekankan bahwa alat generatif hanya sebaik keterampilan insinyur yang menggunakannya; pengawasan manusia penting untuk memeriksa bug, memberi konteks organisasi, dan menangani kebutuhan pengkodean yang kompleks. Prinsip yang sama berlaku untuk *citizen development*: alat bisa semakin pintar, tetapi akal organisasi tidak boleh diturunkan. (McKinsey & Company)

10. Strategi Organisasi: Kapan Menggunakan No-Code, Low-Code, atau Code-First

Agar *no-code/low-code* benar-benar menjadi mesin inovasi cepat, organisasi perlu memilih pendekatan sesuai jenis masalah. Secara umum,

no-code paling cocok untuk otomasi sederhana, formulir, aplikasi internal ringan, alur persetujuan, survei, dan solusi yang relatif seragam. *Low-code* lebih tepat untuk aplikasi bisnis yang butuh integrasi, penyesuaian moderat, dan kolaborasi antara pengguna bisnis dan pengembang profesional. Pendekatan *code-first* tetap diperlukan untuk sistem inti, aplikasi berskala besar, logika sangat kompleks, kebutuhan keamanan sangat tinggi, dan solusi yang menuntut kustomisasi penuh. Microsoft sendiri menegaskan bahwa kombinasi *low-code* dan *code-first* sering kali merupakan jalan terbaik. (Microsoft)

Strategi ini dapat dibayangkan sebagai model portofolio. Bukan semua masalah harus diselesaikan dengan satu jenis alat. Sebuah organisasi yang cerdas akan memetakan kebutuhannya: mana solusi cepat untuk efisiensi lokal, mana solusi integratif untuk lintas unit, mana sistem strategis yang harus dibangun lebih formal. Pendekatan semacam ini juga membantu menghindari dua kekeliruan umum. Kekeliruan pertama adalah "meng-*low-code*-kan semua hal", sehingga platform visual dipaksa menangani beban yang sebenarnya tidak cocok. Kekeliruan kedua adalah "meng-*enterprise*-kan semua hal", sehingga kebutuhan kecil harus melewati prosedur proyek raksasa. Inovasi cepat menuntut diskriminasi yang tepat, bukan sekadar antusiasme teknologi.

Secara manajerial, strategi ini perlu diterjemahkan ke struktur. Organisasi memerlukan pusat keunggulan (*center of excellence*) atau fungsi koordinasi yang menetapkan standar, menyeleksi platform, menyiapkan pelatihan, dan mendampingi unit kerja. Microsoft menekankan bahwa keberhasilan platform *low-code* bergantung pada kebijakan dan strategi yang terdefinisi dengan baik, termasuk identifikasi siapa yang cocok untuk program, pelatihan dalam keamanan dan perencanaan aplikasi, serta keterlibatan pemimpin TI dalam strategi dan mitigasi risiko. (Microsoft)

11. Narasi Kasus: Ketika No-Code/Low-Code Mempercepat, dan Ketika Ia Menyesatkan

Mari kita bayangkan dua organisasi. Organisasi pertama adalah universitas yang ingin mempercepat layanan akademik. Selama ini, permintaan surat, persetujuan topik, dan pelaporan kegiatan mahasiswa dilakukan manual atau lewat email. Tim TI kecil dan sibuk. Fakultas kemudian memakai platform *low-code* untuk membuat portal permintaan layanan, alur persetujuan digital, dan notifikasi otomatis. Karena pemilik proses terlibat langsung, kebutuhan diterjemahkan dengan cepat. Dalam tiga minggu, sistem sederhana sudah dipakai. Waktu layanan turun drastis, beban administrasi menyusut, dan dosen serta mahasiswa merasa proses lebih jelas. Ini contoh ketika *low-code* benar-benar mempercepat inovasi karena masalahnya jelas, ruang lingkupnya cukup sempit, dan integrasi dasarnya dapat dikelola.

Sekarang organisasi kedua: sebuah perusahaan distribusi membiarkan setiap cabang membangun aplikasi sendiri untuk inventaris, pengiriman, dan klaim. Awalnya semua tampak cepat. Namun beberapa bulan kemudian, data stok antar cabang tidak sinkron, definisi “barang rusak” berbeda, akses pegawai lemah, dan ada aplikasi yang menyimpan data pelanggan tanpa kontrol. Cabang-cabang merasa “inovatif”, tetapi kantor pusat kehilangan visibilitas. Ketika audit datang, organisasi menyadari bahwa mereka tidak memiliki peta aplikasi yang jelas. Ini contoh ketika percepatan pengembangan tanpa governance menghasilkan fragmentasi dan risiko.

Dua narasi ini menegaskan bahwa *no-code* dan *low-code* bukan otomatis baik atau buruk. Nilainya bergantung pada *fit* antara kebutuhan, kapabilitas, data, dan kontrol organisasi. Itulah sebabnya Microsoft menempatkan governance sebagai syarat keberhasilan, dan McKinsey menekankan pentingnya pengawasan pengembang berpengalaman, kontrol versi, dan perhatian terhadap kualitas serta keamanan. (Microsoft)

12. Implikasi bagi Kepemimpinan dan Manajemen

Bagi pemimpin organisasi, *no-code/low-code* menuntut perubahan cara pandang. Pemimpin tidak cukup hanya bertanya, "platform apa yang kita beli?" Mereka harus bertanya, "inovasi seperti apa yang ingin kita percepat, masalah backlog apa yang ingin kita kurangi, data apa yang akan disentuh, dan mekanisme pengendalian apa yang kita siapkan?" Dalam pengertian ini, *no-code/low-code* adalah isu strategi dan tata kelola, bukan semata isu aplikasi. Microsoft bahkan menegaskan bahwa governance yang baik membantu manajemen membuat keputusan yang selaras dengan tujuan organisasi, menetapkan tanggung jawab antara TI dan pengembang *low-code*, serta mempercepat waktu ke pasar. (Microsoft)

Pemimpin juga harus mampu mengelola perubahan peran. Ketika *citizen developers* diberdayakan, TI tidak lagi berfungsi hanya sebagai pembangun tunggal, tetapi lebih sebagai arsitek, penjaga standar, pelatih, dan pengawas. Sebaliknya, unit bisnis tidak boleh hanya menjadi peminta solusi, tetapi juga harus bertanggung jawab atas kualitas proses yang mereka digitalkan. Hubungan ini memerlukan kejelasan mandat. Jika tidak, konflik antara "kebebasan inovasi" dan "kontrol korporat" akan terus muncul.

Selain itu, pemimpin harus menyadari bahwa inovasi cepat bukan hanya soal kecepatan menghasilkan sesuatu, tetapi juga kecepatan belajar. Keunggulan utama *no-code/low-code* mungkin bukan pada aplikasi yang selesai dibangun, tetapi pada kemampuan organisasi untuk bereksperimen, melihat umpan balik, memperbaiki proses, dan mengulangi siklus dengan biaya lebih rendah. Dalam hal ini, *no-code/low-code* paling berharga ketika dipakai sebagai mesin pembelajaran organisasi, bukan sekadar pabrik aplikasi mini. (Microsoft)

Kesimpulan

No-code dan *low-code* menandai perubahan penting dalam cara organisasi membangun solusi digital. *No-code* memungkinkan pengguna nonteknis membangun aplikasi dan otomasi tanpa menulis kode,

sedangkan *low-code* mempercepat pengembangan melalui alat visual sambil tetap memberi ruang untuk kustomisasi dan penambahan kode. Keduanya berkontribusi pada inovasi cepat karena menurunkan hambatan teknis, mempercepat prototyping, mendekatkan pembangunan solusi ke pemilik proses, dan membantu organisasi mengatasi backlog pengembangan. Microsoft, IBM, dan berbagai sumber lain menunjukkan manfaat yang konsisten: kecepatan, penghematan biaya, peningkatan produktivitas, pengisian kesenjangan talenta, dan fleksibilitas yang lebih tinggi. (Microsoft)

Namun, justru karena ia mempercepat, *no-code/low-code* juga memperbesar risiko bila diterapkan tanpa disiplin. *Shadow IT*, kebocoran data, masalah kepatuhan, lemahnya kontrol versi, integrasi yang kacau, dan kualitas aplikasi yang tidak stabil merupakan konsekuensi yang nyata. Microsoft menegaskan perlunya governance untuk mengarahkan pengembang profesional dan *citizen developers*, sedangkan McKinsey dan Gartner mengingatkan bahwa kualitas, keamanan, pengawasan, serta risiko AI menjadi semakin krusial dalam lingkungan pengembangan yang makin terotomasi. (Microsoft)

Karena itu, organisasi yang ingin memanfaatkan *no-code/low-code* sebagai mesin inovasi cepat harus menghindari dua jebakan sekaligus: romantisme teknologi dan konservatisme berlebihan. Romantisme teknologi membuat organisasi percaya bahwa semua masalah bisa diselesaikan dengan platform visual. Konservatisme berlebihan membuat organisasi menolak peluang percepatan hanya karena takut kehilangan kontrol. Jalan yang lebih sehat adalah jalan tengah yang strategis: gunakan *no-code* untuk kebutuhan ringan dan cepat, *low-code* untuk aplikasi bisnis yang lebih kaya integrasi, dan *code-first* untuk sistem yang memang membutuhkan rekayasa mendalam; bangun governance, pelatihan, arsitektur data, dan pengawasan TI yang memadai; serta perlakukan *citizen development* bukan sebagai pemberontakan terhadap TI, melainkan sebagai bentuk kolaborasi baru antara pengetahuan bisnis dan kemampuan digital. Dalam kerangka seperti inilah *no-code*, *low-*

code, dan inovasi cepat dapat benar-benar menjadi kekuatan transformatif bagi organisasi. (Microsoft)

Berikut **Glosarium** dan **Daftar Pustaka (APA 7)** untuk topik **"No-Code, Low-Code, dan Inovasi Cepat."**

Glosarium

No-code
Pendekatan pengembangan perangkat lunak yang memungkinkan pengguna membuat aplikasi dan mengotomasi proses bisnis tanpa menulis kode, biasanya melalui antarmuka visual dan komponen *drag-and-drop*. (ibm.com)

Low-code
Pendekatan pengembangan aplikasi yang mengurangi kebutuhan *hand-coding* dengan memakai blok visual, komponen siap pakai, dan otomatisasi, tetapi tetap memberi ruang untuk menambahkan kode bila diperlukan. (ibm.com)

Citizen developer
Pengguna bisnis atau pegawai nonprogrammer profesional yang memakai platform *low-code/no-code* untuk membangun aplikasi, otomasi, atau solusi digital bagi kebutuhan kerja mereka. (Microsoft)

Visual development
Metode pengembangan yang mengandalkan antarmuka grafis, elemen visual, dan konfigurasi komponen alih-alih penulisan kode dari nol. (ibm.com)

Drag-and-drop interface
Antarmuka yang memungkinkan pengguna membangun aplikasi dengan menyeret dan meletakkan komponen visual, sehingga proses desain menjadi lebih cepat dan lebih mudah diakses oleh pengguna nonteknis. (Microsoft)

Rapid application development (RAD)
Pendekatan pengembangan yang menekankan kecepatan pembuatan prototipe, iterasi cepat, dan percepatan *delivery* aplikasi. IBM menjelaskan *low-code* sebagai salah satu bentuk pendekatan RAD. (ibm.com)

Low-code governance
Kerangka aturan, peran, dan pengawasan yang memastikan pengembang profesional maupun *citizen developers* mematuhi kebijakan, prosedur keamanan, dan standar organisasi saat membangun solusi *low-code*. (Microsoft)

Shadow IT
Perangkat lunak, aplikasi, atau infrastruktur yang dibangun atau digunakan di luar pengetahuan, persetujuan, atau standar resmi tim TI dan keamanan organisasi. Dalam konteks *citizen development*, ini menjadi risiko utama bila tidak ada governance. (Microsoft)

Workflow automation
Otomasi alur kerja atau proses bisnis berulang, misalnya persetujuan, notifikasi, formulir, dan perpindahan data antar sistem, yang menjadi salah satu penggunaan paling umum platform *no-code/low-code*. (ibm.com)

Low-code integration
Metode menghubungkan aplikasi, platform, dan *data pipelines* menggunakan konektor siap pakai dan alat visual, sehingga integrasi tidak sepenuhnya bergantung pada kode kustom yang ekstensif. (ibm.com)

Application modernization
Upaya memperbarui aplikasi lama atau proses digital yang usang agar memakai arsitektur, antarmuka, integrasi, dan kapabilitas modern yang lebih cepat, fleksibel, dan mudah dikembangkan. Platform *low-code* sering dipakai untuk mendukung proses ini. (Microsoft Marketing Assets)

Code-first / pro-code development
Pendekatan pengembangan tradisional yang mengandalkan penulisan kode secara langsung dan biasanya dipilih untuk aplikasi yang sangat kompleks, sangat terkustomisasi, atau kritis. (Microsoft)

Reusable components
Komponen, template, konektor, atau fungsi yang sudah tersedia dan dapat digunakan ulang untuk mempercepat pembangunan aplikasi serta mengurangi pekerjaan berulang. (Microsoft)

Power Fx
Bahasa *low-code* pada Microsoft Power Platform yang dirancang menyerupai formula spreadsheet, sehingga dapat dipakai oleh pembuat aplikasi dengan berbagai tingkat kemampuan teknis. (Microsoft Learn)

AI-assisted development
Pengembangan perangkat lunak yang dibantu AI untuk mempercepat pembuatan kode, logika, atau solusi digital, termasuk dalam konteks *low-code* dan prototyping berbasis bahasa alami. (McKinsey & Company)

Unmanaged AI agent proliferation
Pertumbuhan agen AI yang tidak terkelola dengan baik dalam organisasi, yang dapat memperluas permukaan serangan, menghasilkan kode tidak aman, dan menimbulkan risiko kepatuhan—terutama ketika digabungkan dengan *no-code/low-code* dan praktik pengembangan yang terlalu longgar. (Gartner)

Daftar Pustaka (APA 7)

Gartner. (2026, February 5). *Gartner identifies the top cybersecurity trends for 2026*. Gartner Newsroom. (Gartner)

IBM. (n.d.). *Low-code vs. no-code: What's the difference?* IBM Think. (ibm.com)

IBM. (n.d.). *What is low-code?* IBM Think. (ibm.com)

IBM. (n.d.). *What is no code?* IBM Think. (ibm.com)

IBM. (2026, February 3). *What is low-code integration?* IBM Think. (ibm.com)

McKinsey & Company. (2023, June 27). *Unleashing developer productivity with generative AI*. McKinsey & Company. (McKinsey & Company)

McKinsey & Company. (2024). *Technology trends outlook 2024*. McKinsey & Company. (McKinsey & Company)

McKinsey & Company. (2024, May 31). *How generative AI could accelerate software product time to market*. McKinsey & Company. (McKinsey & Company)

McKinsey & Company. (2025, July 22). *The top trends in tech*. McKinsey & Company. (McKinsey & Company)

Microsoft. (n.d.). *Low-code vs. no-code app development*. Microsoft Power Apps. (Microsoft)

Microsoft. (n.d.). *How low-code platforms solve IT challenges*. Microsoft Power Apps. (Microsoft)

Microsoft. (n.d.). *What is low-code governance and why it is necessary*. Microsoft Power Apps. (Microsoft)

Microsoft. (2025, June 4). *Microsoft Power Fx overview*. Microsoft Learn. (Microsoft Learn)

Copilot for this article - Chatgpt 5.2 Thinking. Access date: 15 Maret 2026 Prompting on Writer's account (Rudy C Tarumingkeng)

https://chatgpt.com/c/69b6138d-6df0-839d-b6f5-8878599c9c14