Context Representation and Operational Coherence in System Theoretic Process Design

Abstract

Sociotechnical systems must adapt their behavior as operational conditions, goals, authority, and available resources change. An action that is appropriate in one situation may be unacceptable in another, even when equipment states appear identical. This article develops a context-oriented account of System-Theoretic Process Design, or STPD, in which the implementation preserves the distinctions needed to select and constrain situation-dependent behavior. Building on context-aware computing, human-centred design, human systems integration, and the systems-theoretic safety tradition, the proposed approach connects context variables to operational coherence, Canonical Scenario Representation, and Controller Multi-Service Interactions. It distinguishes actual process conditions from controller representations and from the contextual role that information plays in a decision. Historical restrictions, distributed interpretation, synchronization, transitions, and uncertainty are treated as operational design concerns. A gas-vessel example illustrates how observations acquire meaning through physical expectations, operating scenarios, and retained obligations. The contribution is a design synthesis rather than an empirically validated method: it organizes established concerns into a traceable relationship between operational situations, information requirements, interaction constraints, and verification. Its practical aim is to make the assumptions governing behavior explicit and maintainable throughout operation.

Keywords: System-Theoretic Process Design; operational context; operational coherence; STAMP; STPA; human systems integration; Canonical Scenario Representation; Controller Multi-Service Interaction.

1 Situation dependent behavior as an engineering problem

Modern sociotechnical systems must support safety, productive work, and effective human interaction while conditions change. Their success depends on more than the correct execution of individual functions. A system must also recognize when a previously appropriate response has ceased to be appropriate. Startup, normal operation, maintenance, degraded operation, emergency response, and recovery impose different expectations on the same equipment and on the people responsible for it. Seamless operation, in the sense developed here, means sustaining operational objectives through these changes, including justified restrictions or interruptions when continued operation would violate an applicable constraint.

Consider a command to open a valve. The actuator may be healthy, the command correctly addressed, and the requested movement physically possible. Nevertheless, the movement may be required during one activity, permissible during another, and prohibited while personnel are working on isolated equipment. The action acquires its operational meaning through the circumstances in which it is issued

and executed. An engineering account of correct behavior must therefore encompass the situation, the relevant operational rule, and the evidence that the rule applies.

Experience with known failures remains essential, but recorded errors do not exhaust the combinations of conditions that a system may encounter. New difficulties can arise from familiar elements combined in an unfamiliar way: a delayed command during a handover, a recovered measurement after a protective shutdown, or equipment that is technically operable but has not been released from maintenance. Prospective analysis must examine such distinctions without assuming that every challenge will resemble an earlier incident. This is an argument for extending the questions asked during design, rather than treating all existing engineering as retrospective.

Context-oriented STPD addresses the information and control arrangements needed for situation-dependent behavior. Its guiding question is what the system must establish about the operational situation before an interaction can be interpreted and acted upon appropriately. This article develops that question as a continuous engineering argument. It presents CSR and CMSI as constructs of the proposed STPD account, while identifying the established ideas on which that account depends. Claims about improved safety, productivity, or reduced complexity remain design objectives requiring evaluation.

2 Context in prior work

Dey (2001) made situation-relevant information central to context-aware computing and connected its use to software support through the Context Toolkit. His treatment widens attention beyond the physical surroundings of a device to information relevant to the entities involved in an interaction. Winograd (2001) further emphasizes that contextual relevance arises through interpretation and use. A datum does not become context merely because it is available. These perspectives support a selective engineering approach: the significance of a measurement, identity, or activity depends on the decision it informs.

Human-centred design makes circumstances of use part of evaluation. ISO 9241-11 relates usability to specified users, goals, and conditions of use, encompassing tasks, resources, and environmental circumstances (ISO, 2018). ISO 9241-210 places understanding users, tasks, and environments within an iterative human-centred design process (ISO, 2019). These standards support attention to the human and organizational setting; they do not prescribe the STPD constructs developed here. Their implication for this discussion is that effectiveness cannot be assessed independently of who is acting, toward what purpose, and under what conditions.

Disdier, Masson, Jankovic, and Boy (2024) provide a particularly close foundation in human systems integration, or HSI. Their review of 44 context formulations leads to an operational account relating a focus, its situations, and event-driven history. They characterize context as specific, curated, holistic, transient, entangled, and persistent. In engineering terms, relevant information depends on the activity under consideration, is necessarily selective, and can involve relationships beyond individual subsystems. It changes during operation, both influences and is influenced by activity, and may retain significance after the originating event. STPD adopts these concerns while directing attention toward the information required to govern particular interactions.

This representation-oriented project also has limits. Dourish (2004) challenges approaches that treat context as a fully predefined collection of independent attributes and stresses its emergence through activity. For STPD, that critique cautions against equating the implemented representation with the whole lived situation. Designers can represent selected distinctions on which operational constraints depend without claiming to capture everything participants understand. Observation of work, scenario review, and revision of assumptions remain necessary when practice exposes distinctions that the original model missed.

Situation-awareness research provides a related human-factors perspective. Endsley (1995) connects perception of relevant elements, understanding of their significance, and anticipation of their development to dynamic decision making. An STPD context representation can support those activities, but storing an accurate scenario label does not establish that an operator understands the situation. Interfaces, workload, training, and opportunities to resolve ambiguity influence whether represented information becomes usable understanding. Context engineering must therefore address both computational representation and the human work through which it is interpreted.

3 The systems theoretic foundation and the proposed STPD contribution

STAMP, the System-Theoretic Accident Model and Processes, treats safety through constraints on system behavior and the control structures that enforce them. Leveson (2004, 2012) explains why interactions, software, organizational arrangements, and inadequate control can produce losses even when individual components operate as intended. This perspective supplies the foundation for considering operational behavior across technical and organizational boundaries. The relevant unit of concern is not limited to an isolated component or an individual mistake.

System-Theoretic Process Analysis, or STPA, applies this foundation through analysis of losses, hazards, control structures, unsafe control actions, and causal scenarios. It already considers the circumstances in which an action is unsafe, including omission, provision, timing or ordering, and inappropriate duration. The STPA Handbook also addresses design and requirements development and applications beyond safety (Leveson and Thomas, 2018). STPD should therefore not be positioned as introducing context into an otherwise context-free method, or as introducing design into a tradition confined to retrospective analysis.

The proposed contribution is more specific. STPD organizes situation-dependent constraints around an operational representation and the interactions that rely on it. It asks how the contextual distinctions identified during analysis become established, retained, distributed, and used in the implemented sociotechnical system. CSR provides the scenario abstraction; CMSI provides the interaction perspective. Their combination makes it possible to trace an operational distinction to the controllers that need it, the behavior it governs, and the conditions under which it remains valid.

This emphasis is compatible with STPA and can use its findings as inputs. An unsafe control action may expose a missing distinction between maintenance and production, or between a requested handover and an accepted handover. STPD can then organize the representation and interaction requirements

associated with that distinction. Conversely, reviewing a scenario transition or a shared context dependency may reveal additional circumstances for hazard analysis. The two activities can inform each other without assuming that the proposed terminology replaces the existing method.

4 Context representation and operational coherence

Operational coherence concerns the relationship between actual behavior and the behavior appropriate to the applicable operational circumstances. A representation supplies a reference for deciding which constraints apply, but agreement with that representation alone is insufficient. A system can behave consistently with a mistaken belief. Coherence assessment must therefore examine both the adequacy of the representation relative to actual conditions and the behavior produced using it.

The central STPD argument is one of behavioral discrimination. When two operational situations demand different handling of an interaction, the design must preserve the distinction somewhere in the mechanism governing that interaction. If a command is evaluated using indistinguishable information in both situations, that information cannot justify selecting different responses. The designer must add or improve the relevant evidence, retain a consequential part of history, constrain the operation, or choose a response acceptable across the unresolved possibilities. Context can be embodied in software, physical interlocks, procedures, and human responsibilities; it need not reside in a single central data structure.

Context completeness and context correctness describe complementary concerns. Completeness is relative to the operational scope and the distinctions needed by its decisions. It does not require an exhaustive account of the world. Correctness concerns whether the represented information is sufficiently accurate, current, and meaningful for its intended use. A model can include the right variable but hold an obsolete value, or it can hold accurate measurements while omitting an authorization condition that determines whether action is permissible.

Neither property guarantees successful control by itself. The applicable constraints can be wrong or conflicting, the decision mechanism can misuse valid information, and the commanded response can fail to occur. The coherence argument therefore establishes a design obligation concerning context, not a universal assurance of safety. Its practical value is to expose a context gap when the implemented arrangement cannot distinguish situations for which its intended rules differ.

The aim is a sufficient and maintainable representation, rather than the smallest possible model at any cost. Additional information can increase complexity without improving discrimination, but aggressive simplification can conceal necessary distinctions. Each contextual element should be justified by its effect on a decision or interaction. Sufficiency remains bounded by the hazards, objectives, assumptions, and scenarios actually examined and must be reconsidered when the operating scope changes.

5 State process models and the contextual role of information

The distinction between state, process model, and context is functional. A state description concerns the actual condition of the process or environment within the chosen modeling boundary. Pressure, valve

position, equipment availability, and location can serve this descriptive role. Here, state is used in an engineering sense without implying that every observation is a state variable in a formal dynamic model.

A controller's process model concerns its representation of the process and the conditions relevant to control. The representation may be encoded in software or maintained through human understanding. Leveson's systems-theoretic treatment emphasizes the importance of process-model adequacy and feedback (Leveson, 2012). Actual valve position and the controller's represented valve position must remain analytically distinct: delayed feedback can leave the controller treating an open valve as closed. Internal consistency does not remove the discrepancy.

Information serves a contextual role when it changes the interpretation, admissibility, or required handling of an interaction. Actual pressure describes a process condition; represented pressure belongs to a controller's process model; pressure becomes contextually relevant when a decision depends on it. These are related roles rather than mutually exclusive classes. The same information can matter greatly to one interaction and have no bearing on another.

Contextual relevance extends beyond equipment. An operation may depend on mission phase, protection status, personnel clearance, communication capability, available redundancy, or authority. Spatial context can concern position relative to a restricted region rather than coordinates alone. Temporal context can concern the interval since an earlier action. Organizational context can distinguish equipment that works from equipment formally released for use. Human context should be limited to conditions actually required by the operational rule, such as an assigned qualified role; the framework does not presume that every human characteristic is measurable or should be monitored.

These dimensions often act together. Maintenance mode does not by itself establish that personnel are present, that isolation is complete, or that restoration has been authorized. A scenario must preserve the combinations that affect behavior. A broad label that conceals consequential variation can be as inadequate as an omitted variable.

6 Establishing context through evidence and operational interpretation

STPD distinguishes how contextual information is obtained from the role it plays. Measured information comes from observation or sensing. Derived information follows from observations and an established relationship. Inferred information depends on an assessment of evidence that may not uniquely determine the answer. Virtual context variables express operational distinctions introduced because physical observations alone do not adequately carry their meaning. These descriptions can overlap: a virtual leak assessment may itself be inferred.

The gas-vessel example illustrates the difference. A pressure decrease in a nominally sealed vessel does not by itself establish leakage, because temperature changes can also affect pressure. A suitable physical model, using measured pressure and temperature together with vessel properties and justified assumptions, can support an estimate of gas inventory. That inventory estimate concerns a physical quantity and is a derived physical variable. Its calculation does not automatically make it a virtual operational state.

A leak assessment has a different role. A distinction between normal conditions, suspected leakage, and confirmed leakage summarizes the operational interpretation of measurements, expected flows, valve configuration, model validity, and the available corroboration. No individual sensor directly measures the operational status of suspicion. Introducing such a variable can make the required response explicit, provided the design also establishes how the assessment is reached, who may change it, and what uncertainty remains.

Interpretation depends on the operating scenario. A reduction in estimated inventory may be consistent with authorized process consumption or maintenance depressurization. The same reduction during expected isolation may justify investigation or protective action. Startup can introduce transient conditions that challenge the assessment model. None of these labels automatically explains away a discrepancy: the expected flows, physical assumptions, and evidence must support the interpretation. This example illustrates the need for context and a reference model; it does not specify a validated leak detector.

Context-variable discovery can begin by comparing plausible situations in which the same interaction needs different handling. If routine operation permits remote opening but work on isolated equipment prohibits it, isolation and release status become candidate dependencies. If a delayed command loses validity during a handover, command age, authority, and association with the originating operation may matter. Designers then establish the meaning, source, ownership, update conditions, and operational consequence of each candidate. Availability in an existing database is neither sufficient justification for inclusion nor a reason to omit a missing distinction.

7 Historical and distributed context

History matters when its consequences outlast the observations that initiated them. A closed valve may have reached that position through routine shutdown or through automatic protective isolation. Identical present measurements can then require different restoration behavior. A retained protection or recovery status can carry the relevant consequence without requiring every subsequent decision to reconstruct the complete event history. Such persistence is consistent with the historical dimension identified by Disdier et al. (2024).

Persistence must be governed as carefully as activation. The design should establish which evidence permits release, who has authority to approve it, and how the condition survives a restart, communication interruption, or shift handover when necessary. A new scenario label must not silently erase a maintenance isolation or an unresolved concern about sensor integrity. Indefinite retention can also impede necessary work, so safe release and correction of obsolete restrictions belong in the design.

Context is distributed because controllers see different parts of an operation. A sensor service observes physical quantities, a maintenance team holds release information, and a supervisory controller may hold the current mission or production objective. Each participant can maintain a locally reasonable representation while the joint interaction depends on information held elsewhere. Local context supports

a participant's own decisions; shared interaction context comprises the distinctions that participants must interpret compatibly across a boundary.

Compatibility does not require identical internal models or universal access to every observation. Participants need sufficient agreement on the information governing their joint behavior. Equally, agreement does not establish truth: two services can repeat the same stale observation. Context assurance must address both semantic compatibility among participants and the quality of their connection to actual operational conditions.

The communication mechanism alone cannot supply this assurance. A received message may confirm delivery without confirming that its meaning was understood or its prerequisite established. The term available, for example, can mean powered, mechanically functional, unallocated, or authorized for service. Where these meanings affect action, the interaction must distinguish them. Object identity, operation identity, and the scope of an acknowledgement deserve the same attention as the transmitted value.

8 Scenarios and Canonical Scenario Representation

Scenarios organize combinations of context into operationally recognizable situations with associated expectations and constraints. In the proposed STPD treatment, they serve as operational design constructs as well as aids to development-time analysis. They help group circumstances that permit the same handling, while retaining differences that require another response. This grouping can make reasoning more manageable, although a reduction in complexity must be demonstrated for the application rather than assumed.

Canonical Scenario Representation, or CSR, is the proposed abstraction connecting those situations to the interactions governed by them. It makes the relationship between evidence, scenario assessment, and applicable behavior visible. The representation need not contain every raw observation, but it must preserve the distinctions on which the selected constraints depend. A scenario name is therefore an entry point to an operational interpretation, not a substitute for its prerequisites.

The source framework distinguishes external and internal context. External context concerns relevant circumstances outside the selected controller boundary, including other actors, environment, demands, and resources. Internal context concerns the operational interpretation and response state maintained within that boundary, such as a protection condition, a recovery stage, or a permission. The distinction depends on the boundary: one controller's internal protection status may be external information for another. CSR connects these views without confusing an interpretation with the condition interpreted.

Recognition and validation must remain linked. Context evidence helps determine which scenario applies, while the candidate scenario determines which expectations and additional evidence matter. To avoid circular confirmation, the implementation must allow observations to challenge the current scenario and must detect when its assumptions no longer hold. If several scenarios remain plausible, the representation should preserve that ambiguity and identify behavior appropriate to the unresolved possibilities.

Concurrent activities also constrain the abstraction. A facility may be producing while one subsystem remains isolated for maintenance. A single global normal-operation label would be insufficient if it erased the local restriction. CSR therefore needs an explicit scope and a way to retain overlapping obligations. Scenario granularity should follow the interactions and constraints under examination rather than force every participant into an unnecessarily uniform global state.

9 Controller Multi Service Interactions and synchronization

Controller Multi-Service Interaction, or CMSI, provides the second proposed STPD structure. It directs analysis to operational dependencies involving controllers and services, including information, control actions, feedback, and other exchanges. The network of CMSIs shows where a locally valid decision depends on another participant's information, interpretation, or commitment. It complements CSR by identifying who must establish and maintain the context required under each scenario.

In the gas-vessel example, sensing, process-state reporting, leak assessment, operator alerting, and protective isolation can involve several services and controllers. Their joint behavior depends on more than accurate measurements. An assessment may rely on an expectation of isolation, while a maintenance participant knows that controlled depressurization is in progress. A protective command may depend on authority or equipment status that the assessment service does not itself maintain. CMSI analysis makes those dependencies explicit rather than assuming that all participants already share the same situation.

Reviewing each CMSI across the relevant CSRs exposes which context must be available, what it means, and how missing or incompatible information changes handling. The same assessment exchange can carry different obligations during production, maintenance, and protective operation. This combined review also helps identify unnecessary distribution: a participant needs the distinctions relevant to its responsibility, not every detail of the full system model.

The source framework describes a move from coordination toward context synchronization. This is best understood as making a particular coordination requirement explicit. Coordination can already include shared interpretation; synchronization here emphasizes keeping decision-relevant context sufficiently compatible as conditions change. It is not a requirement for every controller to update simultaneously, nor is it equivalent to synchronizing clocks. The practical requirement concerns what each participant may assume when it commits to an action.

Distributed-systems research provides relevant foundations without establishing this STPD construct. Lamport (1978) distinguishes causal event ordering from an assumed universal ordering of all events. For context management, this highlights why reception order or a timestamp alone may not settle whether an authorization precedes a command or remains applicable to it. The application must also establish the semantic relationship between events. Depending on the consequences and timing constraints, explicit operation identifiers, context versions, acknowledgements, or revalidation can help preserve that relationship. These are design options to be justified, rather than a universally prescribed protocol.

10 Transitions and uncertainty

Transitions deserve analysis as operational intervals in their own right. During a mode change or transfer of authority, participants can temporarily apply different assumptions. A command valid when issued may arrive after its permission has expired; an acknowledgement may refer to an earlier request; a queued action may survive the withdrawal of its enabling condition. Testing only stable initial and final scenarios misses these intermediate obligations.

Transition design should establish entry conditions, retained restrictions, responsibility during the interval, and evidence of completion. It should also determine how pending work is handled. Cancellation, renewed validation, explicit acceptance, or temporary restriction may be appropriate. Restriction is not automatically safe: withholding an urgent protective action can itself create a hazard. The response must follow the process consequences, available alternatives, and time in which the relevant action remains effective.

Context uncertainty takes several forms. A value can be unavailable, estimated, disputed, or too old for its intended use. These conditions should not be collapsed into an ordinary operating value. A controller also needs to distinguish the assessed condition from information about its provenance, age, confidence, and consistency. High confidence in an inference cannot substitute for an authorization that must be explicitly granted by another responsible party.

Freshness depends on events as well as elapsed time. A recent release report may cease to apply immediately after new maintenance work begins. Conversely, a protective restriction can remain valid for a long period until its release conditions are met. Update requirements should therefore follow the lifecycle of the contextual fact. Restoring a communication link does not establish that all participants have recovered the current context; outstanding decisions and retained obligations may require reconciliation.

Fallback behavior should be specified for the uncertainty that actually matters. The system may seek corroboration, limit an operation, use an alternative service, or transfer a decision to a person with appropriate evidence and authority. The fallback itself requires analysis, including how normal operation resumes. A blanket instruction to stop, continue, or ask an operator leaves the central context question unresolved unless the circumstances making that response appropriate are established.

11 Design implications and evaluation

An identified context dependency creates an implementation obligation. Requirements should connect the operational distinction to the interaction constraint it supports and identify how the information will be obtained and maintained. The responsible source or conflict-resolution process, recipients, meaning, validity conditions, and release authority must be explicit where consequential. Timing requirements should follow process dynamics and the consequences of delay rather than a convenient uniform update rate.

The resulting traceability links a situation requiring different behavior to its contextual evidence, the relevant CSR and CMSI, the enforcing mechanism, and the verification case. A requirement such as

maintaining current context is too general to establish compliance. A useful requirement identifies the decision affected and the evidence needed before it proceeds, including the response when that evidence is absent or inconsistent. This makes context engineering reviewable across software, procedures, interfaces, and organizational responsibilities.

Human participation must be designed with the same specificity. If an operator is responsible for validating a scenario or releasing a restriction, the interface must explain the basis of the assessment, unresolved uncertainty, and consequences of the proposed action. The person needs adequate time and authority to decide. Evaluation with representative users and tasks should examine whether the information supports appropriate action in the circumstances of use, consistent with the human-centred orientation of ISO (2018, 2019). More displayed data is not evidence of better understanding.

Verification should deliberately compare situations that look similar but require different handling. The same command can be exercised during routine shutdown and protective isolation, before and after a handover, or with operational availability established but maintenance release absent. Tests should include stale observations, conflicting reports, reordered or duplicated messages, restart, concurrent activities, and restoration after degraded operation where relevant. The purpose is to demonstrate that the distinctions identified in analysis produce the intended behavior, including persistence and justified release of constraints.

AI-assisted analysis could support discovery by comparing engineering artifacts, scenarios, physical expectations, and interaction descriptions to propose overlooked distinctions. Such suggestions remain hypotheses. Engineers must check their meaning, observability, reliability, and connection to enforceable constraints. Automated generation of more variables or scenarios is not evidence of coverage, and an inferred contextual assessment should retain the limitations of the observations and models on which it depends.

Empirical evaluation remains necessary. A useful study would compare the dependencies and requirements identified through STPD with those produced by an existing STPA or HSI process on the same application. It could examine consequential omissions found, effort required, representation maintenance, transition failures detected, and operator interpretation. Operational trials would additionally need to evaluate the effects of restrictions and false assessments on both safety and productive work. Until such evidence is available, CSR and CMSI should be treated as a proposed organization of design reasoning, without claims of proven superiority or exhaustive coverage.

12 Conclusion

Situation-dependent behavior requires a system to preserve the distinctions on which its operational rules depend. Context-oriented STPD turns that observation into a design focus: identify the information required for an interaction, establish its meaning and validity, and maintain it across the controllers and transitions that determine behavior. Operational coherence requires attention to actual circumstances as well as to consistency within the represented scenario.

The proposed framework brings together context variables, CSR, and CMSI to connect situations to interaction constraints. Its treatment of state and process-model roles preserves the difference between the world and a controller's beliefs; its treatment of history and uncertainty prevents a current observation or scenario label from erasing unresolved obligations. Context synchronization then concerns the compatible interpretation needed for joint action, with sufficient evidence that the interpretation remains applicable.

The underlying importance of context, feedback, human activity, and distributed control is established prior art. STPD's contribution is the proposed synthesis of these concerns into an operational design structure. Its value will depend on whether that structure helps engineers expose consequential assumptions, implement the necessary distinctions, and verify behavior under changing conditions. Context representation is thus part of the mechanism governing operation, and its adequacy must be demonstrated at the interactions where it matters.

References

Dey, A. K. (2001). Understanding and using context. Personal and Ubiquitous Computing, 5, 4–7. https://doi.org/10.1007/s007790170019

Disdier, A., Masson, D., Jankovic, M., and Boy, G.-A. (2024). Defining and characterizing the operational context for human systems integration. Systems Engineering, 27(6), 1103–1115. https://doi.org/10.1002/sys.21775

Dourish, P. (2004). What we talk about when we talk about context. Personal and Ubiquitous Computing, 8, 19–30. https://doi.org/10.1007/s00779-003-0253-8

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors, 37(1), 32–64. https://doi.org/10.1518/001872095779049543

International Organization for Standardization. (2018). Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. ISO 9241-11:2018. https://www.iso.org/standard/63500.html

International Organization for Standardization. (2019). Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems. ISO 9241-210:2019. https://www.iso.org/standard/77520.html

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7), 558–565. https://doi.org/10.1145/359545.359563

Leveson, N. G. (2004). A new accident model for engineering safer systems. Safety Science, 42(4), 237–270. https://doi.org/10.1016/S0925-7535(03)00047-X

Leveson, N. G. (2012). Engineering a safer world: Systems thinking applied to safety. MIT Press. https://doi.org/10.7551/mitpress/8179.001.0001

Leveson, N. G., and Thomas, J. P. (2018). STPA handbook. Massachusetts Institute of Technology. https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

Winograd, T. (2001). Architectures for context. Human–Computer Interaction, 16(2–4), 401–419. https://doi.org/10.1207/S15327051HCI16234_18